

A Study of 3D Representations for Human–Scene Interaction: From Classical to Neural

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Mohamad Aymen Mir, M.Sc.
aus Bristol/ Vereinigtes Königreich

Tübingen
2026

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der
Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:

20.07.2026

Dekan:

Prof. Dr. Thilo Stehle

1. Berichterstatter:

Prof. Dr. Gerard Pons-Moll

2. Berichterstatter:

Prof. Dr. Peter Gehler

ماما رتہ ابو،

توہنڈن ساریوے قوربانین خاطر، توہنڈس سارے مدلس خاطر،
رتہ ساریوے کھورہ زیادہ پکان روزہ نکس تہہ طاقس خاطر،
پیلہ میانی پنی ہمت مے تڑاوتھ رگے۔

ABSTRACT

Creating virtual humans that look realistic and behave naturally in 3D environments is a central challenge in computer vision and graphics. At its core lies a fundamental question: how should we represent 3D humans and their environments to enable realistic interaction? Classical mesh and point cloud representations provide the geometric machinery for physics, collision detection, and path planning—but achieving photorealism requires complex material models, global illumination, and high polygon counts. 3D Gaussian Splatting offers photorealistic rendering at real-time rates from images alone—but provides no explicit surfaces, distance fields, or navigable structures. Can these missing geometric capabilities be recovered entirely from Gaussian fields? This thesis investigates this question in two parts: the first leverages the geometric strengths of classical representations for human appearance, capture, and motion synthesis, while the second demonstrates that the same capabilities can be extracted directly from raw Gaussian fields, enabling photorealistic human–scene interaction without classical geometry.

The broader question of how to represent 3D humans is especially pressing as we enter the era of video diffusion models, which can generate photorealistic animations of humans from 2D data alone, challenging the very relevance of 3D representations. While the challenge is valid, we contend that 3D representations will remain essential for simulators, game engines, and digital twins, where geometry-consistent rendering, multi-view coherence, and explicit control over motion, lighting, and physics are required—capabilities that 2D generative models fundamentally cannot provide. But we also contend that for 3D representations to remain competitive, they must close the photorealism gap: the answer is not to abandon 3D, but to evolve it. Neural 3D representations can deliver both geometric control and photorealism. This thesis is our contribution toward that goal.

In the first part, we use classical representations to address foundational problems. We introduce Pix2Surf, a method to transfer textures of clothing images to 3D garments worn on top of SMPL [Loper et al., 2015] in real time, learning dense correspondences from garment silhouettes to UV maps using shape information alone. We then introduce HPS (Human POSEitioning System), a method to recover the full 3D pose of a human registered with a 3D scan of the surrounding environment using body-mounted sensors. HPS fuses camera-based self-localization with IMU-based body tracking, enabling capture across environments spanning 300–2500 m^2 . Next, we introduce a method for synthesizing animator-guided human motion across 3D scenes by composing short-term motions in a canonical coordinate frame, generating long sequences of diverse actions without scene-specific training data.

In the second part, we introduce 3D Gaussian Splatting as a unified representation for photorealistic human–scene interaction. We present InteractSplat, an end-to-end pipeline that reconstructs separate controllable human and object models from multi-view video and animates them together in diverse 3DGS environments, enabling

long-horizon sequences where avatars navigate scenes, pick up objects, and set them down at new locations. We then present SALA, the first method to synthesize human interactions in diverse 3D environments using 3DGS as the sole underlying representation, extracting navigable structures from raw Gaussian fields and introducing differentiable contact refinement in Gaussian space. To improve the realism of composited avatars, we introduce RAGA, a ray-traced shadow casting formulation that computes physically plausible shadows entirely in Gaussian space at interactive rates. Finally, we present AHOY, which exploits the fact that 3DGS can be reconstructed from images alone—without the dense multi-view capture or depth sensors that mesh-based avatars require—to build complete, animatable avatars from in-the-wild YouTube video despite heavy occlusion. Using video diffusion priors, AHOY unlocks YouTube-scale footage as a practical source of 3D human assets that can be animated and composited into 3DGS scenes.

Together, the contributions of this thesis demonstrate that 3DGS can serve as a complete replacement for classical representations in the human–scene animation pipeline—from navigation and locomotion to contact modeling and shadow casting—while delivering superior visual fidelity.

ZUSAMMENFASSUNG

Die Erstellung virtueller Menschen, die realistisch aussehen und sich natürlich in 3D-Umgebungen verhalten, ist eine zentrale Herausforderung in Computer Vision und Computergrafik. Im Kern liegt eine grundlegende Frage: Wie sollten wir 3D-Menschen und ihre Umgebungen repräsentieren, um realistische Interaktion zu ermöglichen? Klassische Mesh- und Punktwolken-Repräsentationen liefern das geometrische Instrumentarium für Physik, Kollisionserkennung und Pfadplanung – doch das Erreichen von Fotorealismus erfordert komplexe Materialmodelle, globale Beleuchtung und hohe Polygonzahlen. 3D Gaussian Splatting bietet fotorealistisches Rendering in Echtzeit allein aus Bildern – liefert jedoch keine expliziten Oberflächen, Distanzfelder oder navigierbaren Strukturen. Können diese fehlenden geometrischen Fähigkeiten vollständig aus Gaußschen Feldern gewonnen werden? Diese Arbeit untersucht diese Frage in zwei Teilen: Der erste nutzt die geometrischen Stärken klassischer Repräsentationen für das Aussehen von Menschen, deren Erfassung und Bewegungssynthese, während der zweite zeigt, dass dieselben Fähigkeiten direkt aus rohen Gaußschen Feldern extrahiert werden können, was fotorealistische Mensch-Szene-Interaktion ohne klassische Geometrie ermöglicht.

Die übergeordnete Frage, wie 3D-Menschen repräsentiert werden sollten, ist besonders drängend, da wir in die Ära der Videodiffusionsmodelle eintreten, die fotorealistische Animationen von Menschen allein aus 2D-Daten erzeugen können und damit die Relevanz von 3D-Repräsentationen grundsätzlich in Frage stellen. Obwohl diese Herausforderung berechtigt ist, vertreten wir die Auffassung, dass 3D-Repräsentationen für Simulatoren, Spiele-Engines und digitale Zwillinge unverzichtbar bleiben werden, wo geometriekonsistentes Rendering, Mehrfachansichtskohärenz und explizite Kontrolle über Bewegung, Beleuchtung und Physik erforderlich sind – Fähigkeiten, die 2D-generative Modelle grundsätzlich nicht bieten können. Wir vertreten aber auch die Auffassung, dass 3D-Repräsentationen, um wettbewerbsfähig zu bleiben, die Fotorealismus-Lücke schließen müssen: Die Antwort liegt nicht darin, 3D aufzugeben, sondern es weiterzuentwickeln. Neuronale 3D-Repräsentationen können sowohl geometrische Kontrolle als auch Fotorealismus liefern. Diese Arbeit ist unser Beitrag zu diesem Ziel.

Im ersten Teil verwenden wir klassische Repräsentationen zur Lösung grundlegender Probleme. Wir stellen Pix2Surf vor, eine Methode, um Texturen von Kleidungsbildern in Echtzeit auf 3D-Kleidungsstücke zu übertragen, die über SMPL [Loper et al., 2015] getragen werden, wobei dichte Entsprechungen von Kleidungssilhouetten zu UV-Karten allein mithilfe von Forminformationen gelernt werden. Anschließend stellen wir HPS (Human POSeitioning System) vor, eine Methode zur Wiederherstellung der vollständigen 3D-Pose eines Menschen, die mit einem 3D-Scan der umgebenden Umgebung mithilfe am Körper befestigter Sensoren registriert wird. HPS verbindet kamerabasierte Selbstlokalisierung mit IMU-basierter Körperverfolgung und ermöglicht die Erfassung in Umgebungen von 300–2500 m^2 . Dann stellen

wir eine Methode zur Synthese von Animator-geführter menschlicher Bewegung in 3D-Szenen vor, die Kurzzeitbewegungen in einem kanonischen Koordinatensystem zusammensetzt und lange Sequenzen verschiedener Aktionen ohne szenenspezifische Trainingsdaten erzeugt.

Im zweiten Teil führen wir 3D Gaussian Splatting als einheitliche Repräsentation für fotorealistische Mensch-Szene-Interaktion ein. Wir präsentieren InteractSplat, eine End-to-End-Pipeline, die separate steuerbare Mensch- und Objektmodelle aus Mehrfachansichtsvideo rekonstruiert und sie gemeinsam in verschiedenen 3DGS-Umgebungen animiert, um Langzeitsequenzen zu ermöglichen, in denen Avatare Szenen navigieren, Objekte aufnehmen und an neuen Orten abstellen. Anschließend präsentieren wir SALA, die erste Methode zur Synthese menschlicher Interaktionen in verschiedenen 3D-Umgebungen unter Verwendung von 3DGS als einzige zugrunde liegende Repräsentation, die navigierbare Strukturen aus rohen Gaußschen Feldern extrahiert und differenzierbare Kontaktverfeinerung im Gaußschen Raum einführt. Um den Realismus zusammengesetzter Avatare zu verbessern, stellen wir RAGA vor, eine Formulierung für raytraced Schattenwurf, die physikalisch plausible Schatten vollständig im Gaußschen Raum bei interaktiven Bildraten berechnet. Schließlich präsentieren wir AHOY, das die Tatsache nutzt, dass 3DGS allein aus Bildern rekonstruiert werden kann – ohne die dichte Mehrfachansichtserfassung oder Tiefensensoren, die Mesh-basierte Avatare erfordern – um vollständige, animierbare Avatare aus YouTube-Videos in freier Wildbahn trotz starker Verdeckung zu erstellen. Mithilfe von Videodiffusionspriors erschließt AHOY YouTube-skaliges Videomaterial als praktische Quelle für 3D-Mensch-Assets, die animiert und in 3DGS-Szenen eingefügt werden können.

Zusammengenommen zeigen die Beiträge dieser Arbeit, dass 3DGS als vollständiger Ersatz für klassische Repräsentationen in der Mensch-Szene-Animationspipeline dienen kann – von Navigation und Fortbewegung über Kontaktmodellierung bis hin zum Schattenwurf – und dabei eine überlegene visuelle Qualität liefert.

ABBREVIATIONS

2D	<i>Two Dimensional</i>
3D	<i>Three Dimensional</i>
3DGS	<i>3D Gaussian Splatting</i>
AABB	<i>Axis-Aligned Bounding Box</i>
AR	<i>Augmented Reality</i>
BRDF	<i>Bidirectional Reflectance Distribution Function</i>
BVH	<i>Bounding Volume Hierarchy</i>
CNN	<i>Convolutional Neural Network</i>
CUDA	<i>Compute Unified Device Architecture</i>
DDIM	<i>Denoising Diffusion Implicit Model</i>
DDPM	<i>Denoising Diffusion Probabilistic Model</i>
DoF	<i>Degree of Freedom</i>
EWA	<i>Elliptical Weighted Average</i>
FLAME	<i>Faces Learned with an Articulated Model and Expressions</i>
FPS	<i>Frames Per Second</i>
GPU	<i>Graphics Processing Unit</i>
GT	<i>Ground Truth</i>
ICP	<i>Iterative Closest Point</i>
IK	<i>Inverse Kinematics</i>
IMU	<i>Inertial Measurement Unit</i>
IoU	<i>Intersection over Union</i>

KL	<i>Kullback–Leibler (divergence)</i>
LBS	<i>Linear Blend Skinning</i>
LiDAR	<i>Light Detection and Ranging</i>
LoRA	<i>Low-Rank Adaptation</i>
LPIPS	<i>Learned Perceptual Image Patch Similarity</i>
LSTM	<i>Long Short-Term Memory</i>
MANO	<i>Hand Model with Articulated and Non-rigid Deformations</i>
MDP	<i>Markov Decision Process</i>
MLP	<i>Multi-Layer Perceptron</i>
NeRF	<i>Neural Radiance Field</i>
NN	<i>Neural Network</i>
ODE	<i>Ordinary Differential Equation</i>
PCA	<i>Principal Component Analysis</i>
PPO	<i>Proximal Policy Optimization</i>
PSNR	<i>Peak Signal-to-Noise Ratio</i>
RANSAC	<i>Random Sample Consensus</i>
RGB	<i>Red Green Blue</i>
RGB-D	<i>Red Green Blue plus Depth</i>
RL	<i>Reinforcement Learning</i>
RNN	<i>Recurrent Neural Network</i>
SDF	<i>Signed Distance Field</i>

SDS	<i>Score Distillation Sampling</i>
SfM	<i>Structure from Motion</i>
SH	<i>Spherical Harmonics</i>
SLAM	<i>Simultaneous Localization and Mapping</i>
SMPL	<i>Skinned Multi-Person Linear</i>
SO(3)	<i>Special Orthogonal Group of dimension 3</i>
SSIM	<i>Structural Similarity Index Measure</i>
TPS	<i>Thin Plate Spline</i>
UV	<i>Texture Coordinates</i>
VAE	<i>Variational Autoencoder</i>
VLM	<i>Vision-Language Model</i>
VR	<i>Virtual Reality</i>

NOTATION

a	<i>A scalar</i>
$\mathbf{a}$	<i>A vector (bold upright for Roman letters)</i>
$\boldsymbol{\theta}$	<i>A vector (bold italic for Greek letters)</i>
$\mathbf{A}$	<i>A matrix (bold upright for Roman letters)</i>
$\boldsymbol{\Sigma}$	<i>A matrix (bold italic for Greek letters)</i>
$\mathbf{A}^\top$	<i>Transpose of a matrix</i>
$\mathcal{A}$	<i>A set (calligraphic)</i>
$a \in \mathcal{A}$	<i>An element a of a set $\mathcal{A}$</i>
$\langle \mathbf{a}, \mathbf{b} \rangle$	<i>Dot product between two vectors</i>
$\mathbf{a} \times \mathbf{b}$	<i>Cross product between two vectors</i>
$\mathbf{I}_n$	<i>An identity matrix with n columns and rows</i>
$\ \mathbf{a}\ _p$	<i>L_p norm of a vector</i>
$\mathbb{R}$	<i>The set of real numbers</i>
$\mathbb{Z}$	<i>The set of integers</i>
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	<i>Normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$</i>
$\mathcal{L}$	<i>Loss function</i>
$\mathbb{E}[\cdot]$	<i>Expectation operator</i>
$\odot$	<i>Hadamard (element-wise) product</i>
$:=$	<i>Defined as</i>

$\hat{\mathbf{a}}$	<i>Wedge operator to create a skew-symmetric matrix corresponding to a vector</i>
$\mathbf{A}^\vee$	<i>Vee operator to obtain a vector corresponding to a skew-symmetric matrix</i>
$\exp(\mathbf{A})$	<i>The exponential operator to map a skew-symmetric matrix to a rotation matrix</i>
$\log(\mathbf{A})$	<i>The log operator for mapping a rotation matrix to its corresponding skew-symmetric matrix</i>

TABLE OF CONTENTS

1	Introduction	1
1.1	Representations for 3D Humans	2
1.2	Classical Representations for Humans in Scenes	4
1.3	Neural Representations for Humans in Scenes	4
1.4	Thesis Statement	5
1.5	Organization	6
1.6	Publications	7
2	Related Work	9
2.1	Texturing 3D Humans	9
2.2	Body-Mounted Sensors	11
2.3	Motion Synthesis and Animation	12
2.4	Human-Scene and Human-Object Interaction	13
2.5	Neural Rendering for Humans and Scenes	15
2.6	Inverse Rendering, Shadows, and Relighting	16
2.7	Monocular Human Reconstruction and Generative Priors	17
3	Background	21
3.1	Linear Blend Skinning	21
3.2	SMPL: A Skinned Multi-Person Linear Model	23
3.3	3D Gaussian Splatting	25
3.4	Meshes vs. 3D Gaussian Splatting	30
I	Humans in Scenes with Classical Representations	33
4	Pix2Surf: Learning to Transfer Texture from Clothing Image to 3D Humans	35
4.1	Introduction	36
4.2	Method	37
4.2.1	Preliminary: Parametric Garment Model	37
4.2.2	Non-Rigid Garment Fitting to Retail Images	38
4.2.3	Learning Automatic Texture Transfer	40
4.3	Implementation Details	42
4.4	Results	45
4.5	Experiments	48
4.5.1	Dataset	48
4.5.2	Shape Context Matching and Image-to-Image Translation Baselines	49
4.5.3	Towards Photorealistic Rendering	49
4.6	Conclusion	52

5	Human POSEitioning System (HPS): 3D Human Pose Estimation and Self-localization in Large Scenes from Body-Mounted Sensors	53
5.1	Introduction	54
5.2	Method	55
5.2.1	3D Scene Reconstruction and RGB Image Database	56
5.2.2	Camera Self-localization	56
5.2.3	IMU-Based Pose Estimation	59
5.2.4	IMU-Camera synchronization	59
5.2.5	Joint Optimization	59
5.2.6	Initialization	62
5.2.7	Coordinate Frame Alignment	63
5.3	Implementation Details	64
5.3.1	Hardware Setup	64
5.3.2	Camera Localization Implementation	64
5.3.3	Contact Detection	65
5.3.4	Scene Voxelization	66
5.3.5	Optimization Hyperparameters	66
5.3.6	Batched Optimization	67
5.4	Dataset	67
5.5	Experiments	67
5.5.1	Obtaining Ground Truth Point Clouds	68
5.5.2	Quantitative Evaluation:	71
5.5.3	Qualitative Evaluation	71
5.6	Conclusion	73
6	CoHMo: Generating Continual Human Motion in Diverse 3D Scenes	75
6.1	Introduction	76
6.2	Method	77
6.2.1	From Action Keypoints to an Anchor Pose	78
6.2.2	Canonical Coordinate Frame	79
6.2.3	Path Planning	79
6.2.4	WalkNet	80
6.2.5	TransNet	81
6.2.6	Chained actions	82
6.3	Implementation Details	82
6.3.1	Generating Keypoints in a Scene	82
6.3.2	Architecture Details	85
6.3.3	IK Details	85
6.3.4	Progressive growing of the Mask for TransNet	86
6.3.5	Different values of M used in <i>TransNet</i>	87
6.4	Experiments	87
6.4.1	Implementation Details	87
6.4.2	Datasets	87
6.4.3	Evaluation Metrics	89
6.4.4	Comparison with Baselines	89
6.4.5	Ablation Studies	91

6.4.6	Qualitative Results	93
6.5	Conclusion	94
II	Humans in Scenes with Neural Scene Representations	95
7	InteractSplat: Gaussian Splatting for Human–Object Scene Animation	97
7.1	Introduction	98
7.2	Method	99
7.2.1	Reconstruction	99
7.2.2	Animation	104
7.3	Implementation Details	108
7.3.1	Network Architecture	108
7.3.2	Feature-Based Object Reconstruction	110
7.3.3	Training Details	111
7.4	Experiments	112
7.4.1	Reconstruction Evaluation	112
7.4.2	Animation Evaluation	117
7.4.3	Additional Qualitative Results	119
7.5	Conclusion	119
8	SALA: Scene-Aligned Animation of Human Avatars in Diverse 3D Environments with Gaussian Splatting	121
8.1	Introduction	122
8.2	Background: Gaussian Reconstruction	124
8.2.1	Linear Blend Skinning for Posing	124
8.2.2	3DGS Avatar Learning Pipeline	125
8.3	Method	127
8.3.1	From 3DGS to Navigable and Proxy Structures	128
8.3.2	Gaussian-Aligned Locomotion	129
8.3.3	Interaction Synthesis via SDF Proxy	132
8.3.4	Differentiable Contact Refinement in 3DGS	133
8.3.5	Contact Detection	135
8.4	Implementation Details	135
8.4.1	Coordinate Frame and Height Normalization	135
8.4.2	Hyperparameters	136
8.5	Experiments	137
8.5.1	Rendering Evaluation	137
8.5.2	Motion Quality and Ablations	140
8.5.3	Qualitative Results	142
8.5.4	Failure Modes of Baseline A	142
8.6	Conclusion	144
9	RAGA: Real Time Ray Traced Gaussian Shadow Casting for 3DGS Avatar-Scene Interaction	147
9.1	Introduction	148

9.2	Method	150
9.2.1	Ray–Gaussian Shadowing	150
9.2.2	Exact Ellipsoid Intersection and Entry/Exit	152
9.2.3	Truncated Line Integral (Closed Form)	152
9.2.4	Normalization by Maximum Line Integral	153
9.2.5	From Opacity to Shadow Transmittance	153
9.2.6	Acceleration via BVH over Conservative AABBs	153
9.2.7	Shadow Rendering	154
9.2.8	Avatar Proxy for Temporally Stable Shadows	154
9.3	Implementation Details	155
9.3.1	3DGS Avatar Model	155
9.3.2	Shadow Tracer Implementation	156
9.3.3	Runtime Analysis	156
9.4	Experiments	156
9.4.1	Experimental Setup	157
9.4.2	Shadow Quality Evaluation	159
9.4.3	Temporal Stability	160
9.4.4	Perceptual Study	160
9.4.5	Ablation Studies	160
9.4.6	Qualitative Results	161
9.5	Conclusion	161
9.6	Appendix: Proofs	163
9.6.1	Preliminaries and Notation	163
9.6.2	Proof that $q(\tau)$ is a quadratic and expressions for (A, B, C)	163
9.6.3	Proof of the χ^2 intersection interval and entry/exit times	164
9.6.4	Proof of the closed-form truncated line integral	165
9.6.5	Proof of the global maximum line integral $I_{\max}$	165
9.6.6	Justification of the normalized factor semantics	167
9.6.7	Equivalence between exponentiated attenuation and effective opacity	167
9.6.8	Optical-depth mapping $\kappa(\alpha) = -\ln(1 - \alpha)$	167
10	AHOY! Animatable Humans under Occlusion from YouTube Videos with Gaussian Splatting and Video Diffusion Priors	169
10.1	Introduction	170
10.2	Method	172
10.2.1	Coarse Avatar Construction	172
10.2.2	Hallucinated Supervision Generation	174
10.2.3	Learning the Full Avatar under Multi-View Inconsistencies	176
10.2.4	Animation in 3D Scenes	178
10.3	Implementation Details	178
10.3.1	Network Architecture	178
10.3.2	Training Procedure	179
10.4	Experiments	180
10.4.1	Experimental Setup	180
10.4.2	Animation on YouTube	181

10.4.3	Static Reconstruction on YouTube	182
10.4.4	BEHAVE Evaluation	183
10.4.5	Ablation Study	184
10.4.6	Animation in 3D Scenes	185
10.5	Conclusion	185
11	Conclusion and Future Work	187
11.1	Key Insights	188
11.1.1	Pix2Surf	188
11.1.2	HPS	188
11.1.3	CoHMo	188
11.1.4	InteractSplat	189
11.1.5	SALA	189
11.1.6	RAGA	189
11.1.7	AHOY	190
11.2	Future Work	190
11.2.1	Scaling Motion Synthesis with Video Diffusion Models	190
11.2.2	Bridging Neural and Classical Rendering for Human–Scene Interaction	191
11.2.3	Neural Rendering for Human–Scene Datasets	191
11.2.4	Simulators for Robotics and Autonomous Driving	191
11.2.5	Rethinking the Role of 3D Structure in Visualization	191

LIST OF FIGURES

1.1	2D human representations. a) Stick figures. b, c) Pictorial models.	2
1.2	3D human representations. a) Sphere-based model. b, c) Cylinder-based models.	2
1.3	SMPL [Loper et al., 2015]. A learned mesh-based human body model.	4
1.4	Thesis roadmap. Ch. 2 surveys related work and Ch. 3 provides necessary background. The seven contribution chapters are grouped into two parts: Part I (Chs. 4–6) addresses human appearance, capture, and motion synthesis with classical representations, while Part II (Chs. 7–10) introduces neural representations for photorealistic human–scene interaction. Ch. 11 concludes with insights and future work.	6
3.1	The SMPL body model pipeline. a) Mean template T with joint segmentation and blend weights W . b) Shape blend shapes $B_S(\beta)$ sculpt identity-specific body proportions; joint regressor J adapts joint positions. c) Pose blend shapes $B_P(\theta)$ add pose-dependent vertex corrections. d) Linear blend skinning W articulates the corrected mesh. Image source: [Loper et al., 2015].	25
3.2	Surface splatting with Gaussian reconstruction kernels. Each surface point defines a 2D local reference frame on the object surface (top left). A Gaussian reconstruction kernel is centered at each point and blended in a 2D parameter domain (bottom), producing a smooth rendered surface (top right). This process—projecting anisotropic Gaussian primitives from 3D object space to screen space and blending them via alpha compositing—is the core rendering mechanism of 3D Gaussian Splatting. Image source: Zwicker et al. [2001].	29
4.1	Real-time texture transfer to 3D garments. Our model Pix2Surf digitally maps the texture of online retail store clothing images to the 3D surface of virtual garment items, enabling 3D virtual try-on in real time.	35
4.2	Automatic texture generation for 3D garments. Given online retail store images, our method produces textures for pre-defined garment templates that can be used to virtually dress SMPL [Loper et al., 2015] in 3D.	37
4.3	Overview of method. We build a large paired dataset of 3D garments and online retail store images by non-rigidly aligning 3D garment templates with automatically computed image segmentations (I). This allows us to train our model Pix2Surf (II) to predict correspondences from the UV map of the garment surfaces to image pixel locations.	38
4.4	Custom UV maps. From top to bottom: pants, shorts, and shirt UV maps created in Blender. The UV map shapes are designed to resemble the corresponding garment silhouettes.	44

4.5	Textured garments obtained using our method. From the online retail store images (left), we create textures for three different garment categories (T-shirt, pants, shorts). We use the textured garments to virtually dress SMPL (right).	45
4.6	Textures mapped by Pix2Surf rendered atop SMPL.	46
4.7	Textures mapped by Pix2Surf rendered atop SMPL.	47
4.8	Rendering textured garments on varied SMPL bodies. Once textures are mapped to their corresponding texture maps, they can be rendered atop SMPL in different shapes and poses.	48
4.9	Comparison with SC matching and image-to-image translation baselines. SC matching produces textures with artefacts and Pix2Pix produces blurry results. Left: Pix2Pix, Middle: SC matching, Right: Pix2Surf.	50
4.10	Photorealistic virtual try-on. Given the garment images and a 3D avatar with texture, we show our automatically textured garments on top.	50
4.11	Comparison with VITON and CP-VTON. Image-based methods fail for extreme poses while our 3D approach handles them correctly.	51
5.1	HPS system overview. HPS jointly estimates the full 3D pose and location of a subject within large 3D scenes, using only body-mounted sensors. Left: subject wearing IMUs and a head-mounted camera. Right: using the camera, HPS localizes the human in a pre-built map of the scene.	53
5.2	Overview. Our method uses IMU data, RGB video from a head mounted camera, and a pre-scanned scene as input. We obtain approximate 3D body pose using IMU data, and head camera self-localization to localize the subject within the 3D scene. We then integrate the approximate pose, the camera position and orientation, along with the scene landscape information in a joint optimization framework to obtain the final location and pose estimates.	56
5.3	Camera self-localization. We match the head camera image keypoints with the keypoints from the prefiltered database with known 2D-3D scene correspondences. We then localize the camera on the scene by minimizing a reprojection error of the keypoints. <i>From top to bottom:</i> head camera image (query), top-3 matches from a dataset, depthmaps rendered from the same position to map database keypoints to 3D, synthetic view of the scene from the inferred camera position.	57
5.4	Real versus rendered images. Comparison of a real image from a database and a result of our rendering.	58
5.5	Comparison of IMU and camera trajectories. IMU trajectory (green) and camera self-localization (red). The yellow dot marks the start. Notice the red trajectory is free of drift but has outliers.	60
5.6	Time synchronization between camera and IMU. Measured time drift between camera and IMU clap detections after synchronization with the first clap.	60
5.7	Manually defined back and feet vertices.	65

5.8	Sitting detection. We detect sitting by measuring the distance between a person's root joint and a predefined vertex on the heel. If $d_{\text{sit}} < 0.6 d_{\text{stand}}$ and the frame velocity is below a threshold, we force the hips to be in contact with the scene.	66
5.9	Setup of our ground truth recording system. A) Sample of images captured by Kinects. B) Raw point-cloud obtained by unprojecting the depth (RGB colors mark points from 1st, 2nd and 3rd Kinects respectively. C) Point-cloud after applying background removal procedure.	68
5.10	Global body orientation improvement. Combining the IMU pose with position from camera localization (IMU+Cam (filtered)) results in unnatural motion—the global body orientation does not face the direction of movement. By contrast, HPS correctly estimates the global orientation.	69
5.11	Effect of integrating predicted 3D scene contacts. As a baseline we used camera localization results for localizing SMPL model. Red regions mark closest surface to feet, heels and toes are colored with light blue and blue when IMUs detect ground contact.	70
5.12	We show qualitative results of our method. Our method can localize and estimate the 3D pose of people performing activities as diverse as exercising, dancing, reading, sitting, eating, talking in a range of indoor and outdoor scenes, all <i>without</i> external cameras.	73
6.1	Animator-guided human motion synthesis in diverse 3D scenes. Our method synthesizes diverse actions such as sitting and grabbing in diverse 3D scenes.	75
6.2	Overview of our method. We generate human motion satisfying action keypoint constraints by dividing it into 3 stages: a <i>Walk Motion</i> , which animates the human as it walks between actions, a <i>Transition-In</i> , which blends the walking motion with the target pose computed from keypoints and a <i>Transition-Out</i> , which animates the human back to the walking pose. We use an autoregressive transformer, <i>WalkNet</i> , to synthesize the walking motion, and a masked-autoencoder transformer to generate the blending motion <i>TransNet</i> .	78
6.3	Transition synthesis in the canonical coordinate frame. Using a) the motion-anchor pose in the 3D scene (purple), b) we move the motion sequence into the canonical coordinate frame. c) There <i>TransNet</i> synthesizes transitions (blue) between the input motion and the pose placed at the origin (purple). d) Once the motion is synthesized, we move it back to the scene coordinate frame.	79
6.4	Scene map and path planning. Right: scene point cloud projected down to get a map of the scene. Left: A* trajectory with keypoints in black and tangents in red.	80

6.5	Walking motion synthesis via goal-centric canonicalization. a) Using keypoints and tangents along a path, we move motion from the scene coordinate frame into b) the goal-centric canonical coordinate frame, where c) <i>WalkNet</i> synthesizes motion that converges at the origin of the coordinate frame. d) Once the synthesized motion reaches the origin, we move it back to the scene coordinate frame, and iterate the process to reach the next waypoint.	81
6.6	Automatic keypoint placement in 3D scenes. Using language instruction and semantic segmentation, keypoints can be automatically placed in a 3D scene.	83
6.7	Keypoint generation pipeline. Given a pointcloud with semantic labels and an activity description, we generate keypoints that allow us to synthesize motion describing the activity. The red and green points correspond to the right and left feet respectively, the blue point corresponds to the hand and the purple point corresponds to the root.	83
6.8	Action labels from language descriptions. Examples of action labels generated from language descriptions using a pre-trained language model. The model is able to understand geometric constraints (tallest book) and generate instructions coherent with the language description.	84
6.9	Transformer architecture for WalkNet and TransNet.	85
6.10	Progressive growing of the mask. In the first epoch, <i>TransNet</i> receives the whole sequence as input. In the second, alternative frames are dropped as illustrated by the white stripes. In the subsequent epoch, the masked out region is increased by one more frame. This goes on until all the middle $\frac{M}{2} - 1$ frames are masked out.	86
6.11	User study interface.	91
6.12	Generalization across different scenes. Our method generates motion that generalizes across different scenes. Here we show motion generation in scenes from 4 different datasets: Replica [Straub et al., 2019], Matterport [Chang et al., 2017], HPS [Guzov* et al., 2021] and Scannet [Dai et al., 2017].	92
6.13	Diverse and controllable motion generation. The keypoint representation allows us to generate diverse and highly controllable motion. We show here examples of different grabbing, sitting and newly defined motions.	92
7.1	InteractSplat. Given multi-view RGB of human-object interactions, we reconstruct separate controllable 3DGS models of humans and objects and animate them in diverse 3D scenes—synthesizing long-horizon sequences where avatars navigate scenes, pick up objects, carry them, and set them down at new locations.	97

7.2	Reconstruction pipeline overview. Using position maps as input, we learn Gaussian parameters for Gaussians anchored to canonical human and object templates. Gaussian properties are predicted as 2D maps structured according to a UV unwrapping of the canonical templates. Each pixel corresponds to a single Gaussian. The canonical Gaussians are then posed using LBS for the human and a rigid transform for the object. We guide reconstruction with an occlusion-aware loss.	100
7.3	Occlusion patterns due to human presence.	101
7.4	Learning Gaussians with features.	101
7.5	Canonical object Gaussian maps. Each pixel corresponds to a canonical Gaussian. The maps are time-invariant, yielding high-fidelity textures. Arrows indicate pixel-to-Gaussian correspondence.	102
7.6	Animation pipeline overview. Given reconstructed human and object Gaussians, we animate them in diverse 3D scenes via dynamic 2D maps for collision-free navigation, retrieval-based motion synthesis with inbetweening, Gaussian placement and rendering, and contact refinement in Gaussian space.	104
7.7	StyleUNet architecture overview. The encoder downsamples position maps with residual injection at each scale. The decoder uses StyleGAN2 modulated convolutions with skip connections from the encoder. A mapping network transforms style vector $\mathbf{z}$ into $\mathbf{W}$ space to modulate all decoder layers. View-direction features are injected at 256×256 . An inverse Haar transform converts the wavelet output to 1024×1024 . Two parallel branches (front/back) produce the 1024×2048 output.	109
7.8	Qualitative results of our method. Reconstructed humans interacting with diverse objects in <i>novel human and object poses from three datasets: NeuralDome, BEHAVE, DNA-Rendering</i> .	110
7.9	Qualitative comparison with existing methods for animatable human reconstruction. Existing methods are unable to reconstruct animatable objects.	112
7.10	Ablation and modified baseline comparison.	114
7.11	Comparison with unmodified baselines on BEHAVE. Existing methods are unable to reconstruct animatable objects.	115
7.12	Comparison with DynamicGaussians. Left: Ours, Right: DynamicGaussians (segmented).	115
7.13	Gaussian maps vs. 3D feature space. (Zoom in.) Left: learning Gaussian properties as 2D Gaussian maps for dynamic objects under occlusion. Right: Gaussian properties in 3D feature space.	116
7.14	Gaussian map ablation. Naive 3DGS fails due to occlusion.	117
7.15	Without dynamic maps, the avatar penetrates placed objects.	117
7.16	Scene-level animation and interaction with dynamic objects.	118
7.17	Additional qualitative results. We show novel-view renderings of diverse human-object interaction sequences reconstructed by our method, demonstrating its ability to handle a wide variety of objects and interaction types.	119

- 8.1 **Animating humans in 3DGS scenes.** We introduce a unified Gaussian-based representation for humans and environments, enabling synthesis of photoreal human–scene interactions entirely within the Gaussian splatting framework. 121
- 8.2 **Method overview.** Given a 3DGS scene, we (1) extract navigable structures via PCA-based alignment and orthographic projection to build navigation maps and A* paths (Sec. 8.3.1); (2) synthesize locomotion along the path using an RL policy adapted to 3DGS (Sec. 8.3.2); (3) generate fine-grained interactions (e.g., sitting) via latent-space optimization using a differentiable SDF proxy derived from scene Gaussians (Sec. 8.3.3); and (4) refine contact between the 3DGS avatar and scene for correct placement (Sec. 8.3.4). All stages operate directly on Gaussians without meshes. 128
- 8.3 **Visual comparison with baselines.** Left pair (motion evaluation): Baseline A uses a VGGT [Wang et al., 2025a] mesh from monocular video, producing collisions due to artifacts; our motion synthesis method entirely relies on 3DGS and correctly generates plausible motion in 3D scenes. Right pair (rendering evaluation): Baseline B uses the highest-quality mesh pipeline (Replica [Straub et al., 2019] + RenderPeople [RenderPeople, 2025]); our 3DGS result achieves more photorealistic appearance. Humans prefer our renderings in both settings (Tab. 8.2). 138
- 8.4 **Effect of the egocentric walkability grid (Sec. 8.3.2).** **Left:** without the walkability map, the RL policy lacks local obstacle awareness and generates motion that passes through the desk. **Right:** with the walkability grid extracted from scene Gaussians, the avatar correctly avoids the obstacle. 139
- 8.5 **Effect of SDF proxy (Sec. 8.3.3).** **Left:** without the SDF proxy, interactions generated in 3D scenes penetrate through the 3DGS scene. **Right:** with our SDF proxy, the avatar’s hips rest on the chair and feet maintain floor contact. 141
- 8.6 **Free-viewpoint rendering of monocular videos with inserted animated humans.** Each scene is a DL3DV [Ling et al., 2024] video reconstructed as 3DGS; the human avatar is animated by our pipeline and composited in Gaussian space, enabling consistent rendering from novel viewpoints. **Left:** indoor showroom. **Right:** outdoor park. 142
- 8.7 **Qualitative results across diverse environments and avatar sources.** Scenes span the SuperSplat library and ScanNet++ [Yeshwanth et al., 2023]; avatars are reconstructed from multi-view captures ranging from 4 cameras (BEHAVE [Bhatnagar et al., 2022]) to 48 cameras (ActorSHQ [Işık et al., 2023]). Our method generalizes across reconstruction quality and capture conditions. 143
- 8.8 **Effect of opacity culling on obstacle maps.** (a) Naive orthographic projection of Gaussians leads to blocky structures. (b) Orthographic projection with opacity culling. (c) Path in filtered map. 143

8.9	VGGT mesh reconstruction from monocular video. The underlying mesh contains blocked paths and significant noisy structures.	144
8.10	Additional qualitative results across diverse 3DGS environments. Top: animated human navigating an indoor living room scene while avoiding furniture. Bottom: animated human navigating an outdoor courtyard, demonstrating generalization to outdoor environments with obstacles such as planters, benches, and chairs. Both results are rendered entirely within the Gaussian Splatting framework.	145
9.1	RAGA casts physically plausible shadows for animated 3DGS avatars composited into 3DGS scenes. Our ray-traced Gaussian shadow casting produces soft, temporally stable shadows entirely in Gaussian space, without any mesh reconstruction.	147
9.2	Method overview. Left: a shadow ray is cast from a scene Gaussian toward the light source through an animated 3DGS avatar. Right: comparison of ray-Gaussian intersection strategies.	151
9.3	Avatar proxy. We replace the pose-dependent anisotropic Gaussians (left) with a stable isotropic proxy (right) that eliminates temporal flickering in shadows.	154
9.4	Qualitative comparison on SuperSplat scenes. Pseudo-GT is obtained using a proxy floor mesh and posed SMPL mesh via classical shadow mapping. Our method produces soft, physically plausible shadows that closely match the pseudo-GT. Both 3DGRT [Moenne-Loccoz et al., 2024] and RaySplat [Byrski et al., 2025] lose shadow detail due to their implausible attenuation models.	158
9.5	Visual ablations. Without $I_{\max}$ normalization (left), large anisotropic Gaussians cast disproportionately dark shadows. The linear attenuation model (middle) produces harder shadow falloff than our full exponentiated model (right).	160
9.6	Runtime vs. avatar Gaussians.	161
9.7	Proxy ablation. $ S_t - S_{t-1} $ maps: bright = large change. The proxy stabilizes shadows.	161
9.8	Qualitative results. Our method casts plausible soft shadows for various 3DGS avatars animated alone and with objects across diverse scenes.	162
10.1	AHOY reconstructs animatable 3D avatars from occluded YouTube video. Given a monocular YouTube video with occlusion, AHOY reconstructs a complete, animatable 3D human avatar using video diffusion priors and 3D Gaussian Splatting, enabling photorealistic human animation within 3D scenes.	169

- 10.2 **Method overview. Block 1** (Sec. 10.2.1): We map observed textures from partially occluded video onto a canonical pose via DensePose UV correspondences, inpaint missing regions with FLUX, and generate multi-view canonical-pose images with multi-view diffusion to supervise a coarse 3DGS avatar using canonical Gaussian maps. **Block 2** (Sec. 10.2.2): We finetune a video diffusion model (Wan 2.2) with LoRA to capture the subject’s identity, render the coarse avatar under structured motion sequences, and refine these renderings via RF-Inversion through the identity-finetuned latent space to produce hallucinated supervision videos. **Block 3** (Sec. 10.2.3): The hallucinated videos supervise a full avatar with pose-dependent Gaussian maps, where per-frame poses and cameras are jointly optimized to absorb multi-view inconsistencies; a separate FLAME-based head path preserves facial identity. **Block 4** (Sec. 10.2.4): At inference, the avatar is driven by novel poses and composited into 3DGS scenes. 173
- 10.3 **Static reconstruction comparison (Zoom In).** Top: canonical-pose input. Bottom: occluded input. Our animatable avatar is posed to match the target view. 180
- 10.4 **Reconstruction under difficult fully visible poses.** Static methods struggle to reconstruct humans under difficult poses, while our method allows reposing of 3DGS avatars to match the target pose. 180
- 10.5 **Animation comparison (Zoom In).** Top: canonical-pose input. Bottom (below line): occluded input. Ours produces higher-fidelity avatars in both settings. 181
- 10.6 **Additional static reconstruction comparison.** All methods receive an unoccluded canonical-pose image. Our animatable avatar is posed to match the target view. 182
- 10.7 **Novel-view evaluation on BEHAVE.** We render each reconstruction from held-out BEHAVE side cameras and compare against the ground-truth side views. 183
- 10.8 **Visual ablations.** Each pair shows the ablated result (w/o) and the full pipeline. (A) Without hallucination, the coarse-only avatar lacks detail in unobserved regions. (B) Without RF-Inversion, hallucinated supervision retains rendering artifacts. (C) Without map/LBS decoupling, multi-view inconsistencies cause blurring. (D) Without the head/body split, facial identity is lost. 184
- 10.9 **Animation in 3DGS scenes.** Avatars from occluded YouTube video, animated and composited into 3DGS scenes from casual cell-phone video. 185

LIST OF TABLES

Tab. 5.1	Drift and camera outliers. 3D error (in cm) for the subject standing in A-pose after moving freely around the scene.	72
Tab. 5.2	Drift and camera outliers (dynamic). 3D error (in cm) for the subject walking, standing and leaning on the table, after moving around the scene. Error is measured from the dynamic ground truth point-cloud to the result (3D mesh in motion). Rows indicate distance traveled before evaluation.	72
Tab. 5.3	Foot contact evaluation. For frames when foot contact is detected, we report (in cm) Distance to surface: average distance between foot vertices and the scene. Foot Sliding: average distance on the surface plane between foot vertices in two successive frames. Numbers are computed for a 3 minute long walking sequence.	72
Tab. 6.1	Perceptual study comparison with baselines. Comparisons between our method and existing baselines using a perceptual study.	88
Tab. 6.2	Component-wise analysis. We compare our method with different baselines across three design components: using language based or manually specified keypoints, the walking motion and the transition motion.	88
Tab. 6.3	Foot-skate metric comparison with baselines. Comparisons between our method and existing baselines using the foot-skate metric.	89
Tab. 6.4	Transformer architecture ablation. Ablation study to evaluate the importance of the transformer architecture.	93
Tab. 6.5	Progressive mask growing ablation. Ablation study to evaluate <i>TransNet</i> with and without progressive mask growing.	93
Tab. 7.1	Quantitative results on DNA-Rendering. Evaluated on held-out images. We outperform existing baselines that reconstruct animatable 3D humans.	113
Tab. 7.2	Quantitative results on BEHAVE. Evaluated on held-out images with novel human and object poses.	113
Tab. 7.3	Quantitative results on the NeuralDome benchmark. Our method achieves the best performance across human, object, and full metrics.	114
Tab. 7.4	Ablation study on DNA-Rendering, BEHAVE, and NeuralDome. Each component in our method adds to the performance and our method cumulatively yields best results. Each line in the table corresponds to an experiment detailed in Sec. 7.4.1.4.	116
Tab. 7.5	Animation evaluation. User study shows the percentage of participants preferring our method over each ablation in pairwise comparisons.	118

Tab. 8.1	Hyperparameters for all pipeline stages. Top: RL locomotion reward weights (Eq. (8.14)). Middle: interaction latent-space optimization weights (Eq. (8.15)). Bottom: contact refinement weights (Sec. 8.3.4).	137
Tab. 8.2	Rendering preference (win rate, %). Higher = our 3DGS preferred.	139
Tab. 8.3	Ablation study on ScanNet++. Motion synthesized using 3DGS; metrics evaluated in mesh space.	139
Tab. 8.4	Motion comparison: monocular setting. Our 3DGS-based motion vs Baseline A (VGGT mesh); evaluated against VGGT meshes. Despite operating entirely on Gaussians without explicit mesh geometry, our method achieves substantially lower skating, penetration, and jerk while reaching goals more reliably.	142
Tab. 9.1	Capability comparison.	149
Tab. 9.2	Shadow quality vs. pseudo-GT on ScanNet++. Averaged over 5 scenes. Mesh Shadow is an oracle.	157
Tab. 9.3	Temporal consistency (TSC↓). Averaged over 5 sequences. Lower = more stable.	157
Tab. 9.4	Ablation studies. Evaluated against pseudo-GT on ScanNet++. Best in each group is bold .	159
Tab. 9.5	Avatar proxy ablation. Temporal stability (TSC↓). Lower = more stable.	159
Tab. 10.1	Static reconstruction on YouTube videos. We compare under occluded and canonical-pose input settings.	182
Tab. 10.2	Animation comparison on YouTube videos. We animate reconstructed avatars to novel poses and compare against held-out video frames.	183
Tab. 10.3	Quantitative evaluation on BEHAVE. Novel-view and novel-pose settings.	183
Tab. 10.4	Ablation study on BEHAVE. Each row removes one component from the full pipeline. <i>Novel View</i> : render from held-out side cameras. <i>Novel Pose</i> : animate to unseen ground-truth SMPL-X poses and render from held-out side cameras.	184

INTRODUCTION

*He who learns must suffer.
And even in our sleep, pain which cannot forget
falls drop by drop upon the heart,
until, in our own despair, against our will,
comes wisdom through the awful grace of God.*

Aeschylus, *Agamemnon*

Humans are the primary actors in the world around us. We sit on chairs, walk through rooms, pick up objects, and interact with our surroundings in ways that are effortless in reality but extraordinarily difficult to reproduce digitally. Creating virtual humans that look realistic *and* behave naturally in 3D environments is a central challenge in computer vision and graphics, with applications spanning gaming, film production, AR/VR, robotics simulation, and autonomous driving.

At the heart of this challenge lies a fundamental question: *how should we represent 3D humans and their environments to enable realistic interaction?* Classical mesh and point cloud representations provide the geometric machinery for physics, collision detection, and path planning—but achieving photorealism requires complex material models, global illumination, and high polygon counts. 3D Gaussian Splatting (3DGS) offers photorealistic rendering at real-time rates from images alone—but provides no explicit surfaces, distance fields, or navigable structures. Can these missing geometric capabilities be recovered entirely from Gaussian fields? This thesis investigates this question in two parts: the first leverages the geometric strengths of classical representations for human appearance, capture, and motion synthesis, while the second demonstrates that the same capabilities can be extracted directly from raw Gaussian fields, enabling photorealistic human–scene interaction without classical geometry.

The broader question of how to represent 3D humans is especially pressing as we enter the era of video diffusion models, which can generate photorealistic animations of humans from 2D data alone, challenging the very relevance of 3D representations. While the challenge is valid, we contend that 3D representations will remain essential for simulators, game engines, and digital twins, where geometry-consistent rendering, multi-view coherence, and explicit control over motion, lighting, and physics are required—capabilities that 2D generative models fundamentally cannot provide. But we also contend that for 3D representations to remain competitive, they must close the photorealism gap: the answer is not to abandon 3D, but to evolve it. Neural 3D representations can deliver both geometric control and photorealism. This thesis is our contribution toward that goal.

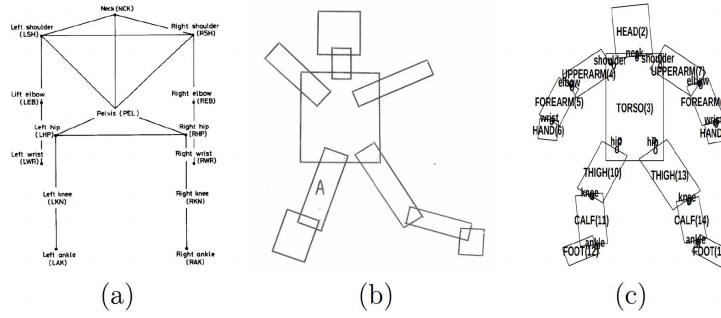


Figure 1.1: 2D human representations. a) Stick figures. b, c) Pictorial models.

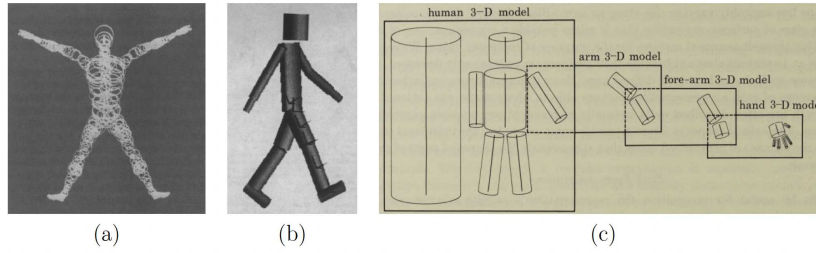


Figure 1.2: 3D human representations. a) Sphere-based model. b, c) Cylinder-based models.

1.1 REPRESENTATIONS FOR 3D HUMANS

Early approaches to modeling humans relied on 2D puppet-like representations [Hinton, 1976] or on connected rectangles to represent body parts [Ronfard et al., 2002] (Fig. 1.1). Similar ideas were used by Felzenszwalb and Huttenlocher [2005] for efficient 2D pose estimation from images, later extended with neural networks for multi-person 2D pose estimation [Insafutdinov et al., 2017, Pishchulin et al., 2016] and tracking [Iqbal et al., 2017].

In 3D, early articulated pose estimation relied on sphere primitives or hierarchical 3D cylinders [Marr and Nishihara, 1978, Hogg, 1983] (Fig. 1.2). These were combined with random forests and Gaussian kernels for machine-learning-based 3D pose estimation. Early neural methods for 3D pose and shape estimation used stick figures [Mehta et al., 2017, 2018] or voxel-based representations [Pavlakos et al., 2017, Varol et al., 2018].

A limitation of these early representations is that they cannot model intricate body details such as hands or clothing. Allen et al. [2006], Angelov et al. [2005] presented the first methods to address this, but their models require expensive post-processing and do not lend themselves to downstream tasks. Building on this, Loper et al. [2015], Pavlakos et al. [2019] (Fig. 1.3) introduced learned mesh-based body models that are both realistic and amenable to tasks such as 3D pose estimation [Kanazawa et al., 2018, Bogo et al., 2016, Omran et al., 2018], scan registration [Bogo et al., 2015], and shape estimation [Zhang et al., 2017]. These models also enabled the creation of

dressed virtual avatars from images [Alldieck et al., 2018b,a, 2019a] and the modeling of clothing as a separate layer [Bhatnagar et al., 2019, Pons-Moll et al., 2017].

Neural radiance fields (NeRF) [Mildenhall et al., 2020] introduced an alternative paradigm: learning a continuous volumetric representation from images, rendered via differentiable ray marching, to produce photorealistic novel views without explicit surface geometry. Methods such as HumanNeRF [Weng et al., 2022] and Animatable NeRF [Peng et al., 2021] learn canonical-space representations from video and articulate them via skeletal deformation fields, achieving high visual fidelity but at the cost of slow per-pixel ray marching that limits interactive use.

3D Gaussian Splatting (3DGS) [Kerbl et al., 2023] overcomes this bottleneck by representing scenes as collections of anisotropic 3D Gaussian primitives rendered via differentiable rasterization at real-time rates. This representation has been rapidly adopted for human avatar modeling: Animatable Gaussians [Li et al., 2024d], 3DGS-Avatar [Qian et al., 2024], and ASH [Pang et al., 2024] learn pose-dependent 3DGS avatars from multi-view video, combining SMPL-based articulation with photorealistic rendering. More recently, generative priors have enabled 3DGS avatar reconstruction from monocular or even single-image input—HUGS [Kocabas et al., 2024] disentangles humans from scenes in monocular video, while HumanSplat [Pan et al., 2024] and LHM [Qiu et al., 2025a] predict complete 3DGS avatars from a single image in seconds. This progression—from multi-view capture to monocular reconstruction—is central to the second part of this thesis.

Representing individual humans, however, is only half the challenge. For virtual humans to be useful in applications such as gaming, AR/VR, and simulation, they must be placed into 3D environments and interact with them convincingly—navigating scenes, manipulating objects, and casting shadows. This requires not just a human representation, but an integrated pipeline for human–scene interaction.

The difficulty is that the two dominant representation families offer complementary but seemingly incompatible strengths. Classical meshes and point clouds are built from explicit geometric primitives—vertices, edges, faces—that directly encode surfaces, distances, and spatial relationships. Decades of algorithms in computer graphics and robotics exploit this structure: navigation meshes for path planning, signed distance fields for collision queries, surface normals for lighting and shadow computation. Yet achieving photorealism with these representations demands complex material models, global illumination, and high polygon counts—a pipeline that is expensive, brittle, and difficult to scale to diverse real-world scenes. 3DGS, by contrast, achieves photorealism almost effortlessly from images alone, but its primitives are semi-transparent volumetric blobs with no notion of a surface, no inside or outside, and no well-defined distance metric. Even basic geometric queries—is this point above the floor? does this limb penetrate that object?—have no direct answer in Gaussian space. Bridging this gap, recovering the geometric capabilities that interaction demands from a representation designed purely for appearance, is the central technical challenge that motivates this thesis. The following two sections describe how classical and neural representations each represent human–scene interactions, and present the contributions of this thesis.



Figure 1.3: SMPL [Loper et al., 2015]. A learned mesh-based human body model.

1.2 CLASSICAL REPRESENTATIONS FOR HUMANS IN SCENES

Mesh-based body models like SMPL, combined with mesh and point cloud representations of scenes, form what we refer to as *classical representations* in this thesis. As outlined above, these representations excel at providing explicit geometry for downstream algorithms, but their rendering quality is fundamentally limited by the fidelity of the underlying surface approximation.

Despite this limitation, classical representations have enabled significant progress in modeling humans in scenes. The pioneering work of Hassan et al. [2019] first captured virtual humans along with their context, though restricted to a static camera with limited recording volume. The first part of this thesis builds on classical representations to address three foundational problems. We begin with human appearance—creating realistic-looking avatars is a prerequisite for any downstream application—before turning to capturing and synthesizing human behavior in 3D scenes:

- **Appearance** (Ch. 4): transferring realistic clothing textures from 2D images to 3D garments worn atop SMPL, enabling virtual try-on and diverse avatar generation.
- **Capture** (Ch. 5): recovering the full 3D pose of a human registered with a large pre-scanned 3D scene using only body-mounted sensors, removing the limitation of external cameras.
- **Motion synthesis** (Ch. 6): generating long-horizon human motion across diverse 3D scenes by composing short-term motions in a canonical coordinate frame, without requiring scene-specific training data.

1.3 NEURAL REPRESENTATIONS FOR HUMANS IN SCENES

As discussed in the previous section, 3DGS offers photorealistic rendering at real-time rates and can be reconstructed from images alone—even sparse or monocular video—opening the door to building 3D human assets from casual, in-the-wild footage at a scale unattainable with classical pipelines. However, adopting 3DGS as the sole 3D representation for human–scene interaction poses fundamental challenges. Gaussian fields provide none of the geometric quantities that classical animation pipelines rely on: there are no explicit surfaces to walk on, no signed distance fields for collision

detection, no navigable meshes for path planning, and no surface normals for shadow casting. The second part of this thesis addresses these challenges, demonstrating that all necessary structures can be extracted or approximated directly from raw Gaussian fields:

- **Reconstruction and animation** (Ch. 7): reconstructing separate controllable 3DGS models of humans and objects from multi-view video and animating them together in diverse 3DGS scenes, enabling long-horizon interaction sequences.
- **Scene-level animation** (Ch. 8): extracting navigable structures from raw Gaussian fields, adapting RL-based locomotion, and introducing differentiable contact refinement—all in Gaussian space—to synthesize human interactions in diverse 3DGS environments.
- **Shadow casting** (Ch. 9): computing physically plausible shadows entirely in Gaussian space using ray–Gaussian line integrals, producing temporally stable shadows at interactive rates without mesh reconstruction.
- **Avatar reconstruction from YouTube video** (Ch. 10): exploiting the image-only reconstruction capability of 3DGS to build complete, animatable avatars from in-the-wild YouTube video under heavy occlusion, using video diffusion priors to generate supervision for unobserved body regions. We demonstrate that the resulting avatars are robust enough to be animated with novel poses and composited into 3DGS scenes, unlocking YouTube-scale footage as a practical source of 3D human assets.

1.4 THESIS STATEMENT

This thesis demonstrates that the choice of 3D representation fundamentally shapes what is achievable in human–scene interaction. Classical mesh and point cloud representations provide the geometric machinery for robust capture and motion synthesis, while 3D Gaussian Splatting enables photorealistic rendering of human–scene interactions at a quality unattainable with classical pipelines. By developing methods across both representation families, we show that 3DGS can serve as a complete replacement for meshes in the animation pipeline—from navigation and locomotion to contact modeling and shadow casting—while delivering superior visual fidelity.

More broadly, this thesis argues that the classical geometry pipeline and photorealistic neural rendering need not remain separate worlds. The methods developed in Part II demonstrate that walkable surfaces, collision boundaries, contact points, and shadow-casting rays can all be derived from the same Gaussian field used for rendering—without ever converting to a mesh or computing an explicit signed distance field. This suggests a path toward a unified representation that serves both photorealistic rendering and animation of humans in scenes, collapsing a pipeline that has historically required multiple heterogeneous representations. Together, the contributions of this thesis advance the state of the art in creating virtual humans that

both look real and interact naturally with their 3D environments, which we hope takes us one step closer to *Real Virtual Humans*.

1.5 ORGANIZATION

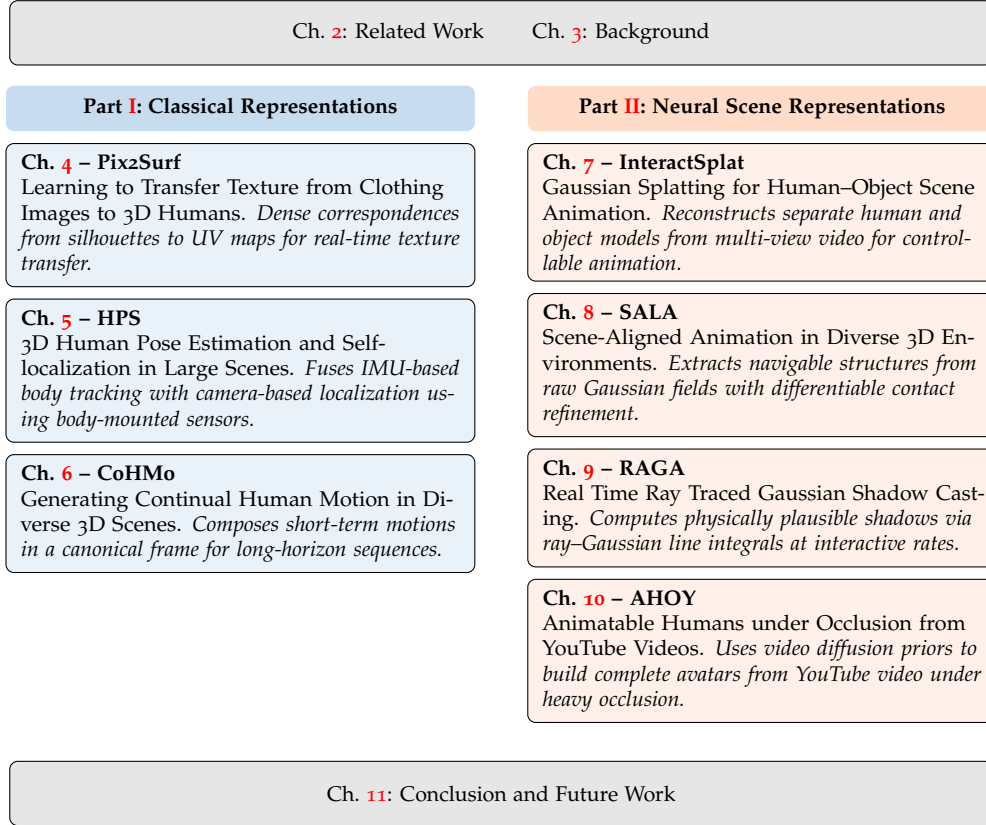


Figure 1.4: **Thesis roadmap.** Ch. 2 surveys related work and Ch. 3 provides necessary background. The seven contribution chapters are grouped into two parts: Part I (Chs. 4–6) addresses human appearance, capture, and motion synthesis with classical representations, while Part II (Chs. 7–10) introduces neural representations for photo-realistic human–scene interaction. Ch. 11 concludes with insights and future work.

1.6 PUBLICATIONS

Chs. 4, 5, 6, 9, and 10 have been published or accepted for publication at the following venues:

- Aymen Mir, Thiemo Alldieck, Gerard Pons-Moll: Learning to Transfer Texture from Clothing Images to 3D Humans, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020
- Vladimir Guzov* and Aymen Mir*, Torsten Sattler, Gerard Pons-Moll: Human POSEitioning System: Human Pose Estimation and Localization in Large 3D Scenes, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021
* = Equal Contribution
- Aymen Mir, Xavier Puig, Angjoo Kanazawa, Gerard Pons-Moll: Generating Continual Human Motion in Large 3D Scenes, *International Conference on 3D Vision (3DV)*, 2024
- Aymen Mir, Riza Alp Guler, Jian Wang, Peter Wonka, Bing Zhou, Gerard Pons-Moll: RAGA: Real Time Ray Traced Gaussian Shadow Casting for 3DGS Avatar-Scene Interaction, *European Conference on Computer Vision (ECCV)*, 2026
- Aymen Mir, Riza Alp Guler, Xiangjun Tang, Peter Wonka, Gerard Pons-Moll: AHOY! Animatable Humans under Occlusion from YouTube Videos with Gaussian Splatting and Video Diffusion Priors, *European Conference on Computer Vision (ECCV)*, 2026

Chs. 7 and 8 are based on preprints available online:

- Aymen Mir, Arthur Moreau, Helisa Dhamo, Zhensong Zhang, Gerard Pons-Moll, Eduardo Pérez-Pellitero: InteractSplat: Gaussian Splatting for Human–Object Scene Animation
- Aymen Mir, Jian Wang, Riza Alp Guler, Chuan Guo, Gerard Pons-Moll, Bing Zhou: SALA: Scene-Aligned Animation of Human Avatars in Diverse 3D Environments with Gaussian Splatting

The author also contributed to the following paper:

- Julian Chibane, Aymen Mir, Gerard Pons-Moll: Neural Unsigned Distance Fields for Implicit Function Learning, *Advances in Neural Information Processing Systems (NeurIPS)*, 2020

RELATED WORK

This chapter surveys the prior work most relevant to the contributions of this thesis. Foundational representations—Linear Blend Skinning, SMPL, and 3D Gaussian Splatting—are covered in Ch. 3. The topics below are organized to follow the two-part structure of the thesis. Secs. 2.1 and 2.2 focus on human appearance and body-mounted capture, which are the primary concerns of Part I. Secs. 2.3 and 2.4 cover motion synthesis and human–scene/object interaction, which straddle both parts of the thesis. Secs. 2.5–2.7 focus on neural rendering, inverse rendering, and monocular reconstruction with generative priors, which are the primary concerns of Part II.

2.1 TEXTURING 3D HUMANS

Physics-Based Garment Modeling. Synthesizing images of people in realistic clothing is a long-studied problem in the Computer Graphics community. Early works enable one to represent and parametrize fabric virtually in 3D [Weil, 1986, Ng and Grimsdale, 1996], and later realistic draping and animation have been achieved [Baraff and Witkin, 1998, House and Breen, 2000]. While these works require artist-designed garments, careful parameter selection, and computationally expensive physics-based simulation, fully-automated and faster methods have been introduced more recently. In contrast, these methods process sensor data such as images, depth maps, or 3D point clouds, and produce 3D reconstructions or photorealistic images with minimal interaction or even fully automatically.

2D Image Synthesis. These methods produce images of people holding a given pose or wearing a desired outfit. These works utilize recent advances in conditional image-to-image translation using generative adversarial networks (GANs) [Goodfellow et al., 2014, Isola et al., 2017]. Lassner *et al.* [Lassner et al., 2017] present a method to produce full-body images of people in clothing from semantic segmentation of the body and clothing. However, the produced subject and outfit are not controllable. Zhao *et al.* [Zhao et al., 2018] produce novel views of given fashion images. While the subject and outfit remain untouched, the pose can be constrained with a sparse set of camera angles. To gain more control over the output, a large number of works deal with synthesizing person images under a desired pose [Ma et al., 2017, 2018, Pumarola et al., 2018, Siarohin et al., 2018, Dong et al., 2018, Balakrishnan et al., 2018, Song et al., 2019, Shysheya et al., 2019]. Grigorev *et al.* [Grigorev et al., 2019] learn warpings between images and the SMPL texture; predicting correspondences instead of color values helps the network to generalize due to much lower variation in the warps than in the color space. In a related direction, Neverova *et al.* [Neverova et al.,

2018] utilize DensePose [Güler et al., 2018] to warp image pixels into the SMPL texture, but inpainting is performed before warping into the target pose. In Ch. 4, a similar correspondence-based approach is adopted, but the focus is on garments rather than full bodies, and correspondences are inferred directly on the clothing surface [Shotton et al., 2012, Pons-Moll et al., 2015, Güler et al., 2018, Neverova et al., 2018]. More related to the texture transfer problem addressed in Ch. 4, recent methods focus on exchanging a subject’s outfit while preserving [Raj et al., 2018, Zangir et al., 2018b] or changing [Han et al., 2018, Wang et al., 2018, Yu et al., 2019, Dong et al., 2019a, Zheng et al., 2019] the pose. Additionally, special case methods for text-based fashion image manipulation [Zhu et al., 2017] and image-based outfit improvement [Hsiao et al., 2019] have been presented. In contrast to Ch. 4, all these *virtual try-on* methods work in the image space and thus perform no explicit reasoning about the underlying 3D scene. This means they are not guaranteed to produce consistent output under varying poses and camera angles. Dong et al. [Dong et al., 2019b] approach this via image warping and subsequent refinement of a previously generated image. While 2D warping improves the quality of synthesized videos, limitations of 2D methods are still present.

3D Reconstruction. These methods focus on recovering the actual 3D shape of a captured garment alone, the body shape of a subject wearing the garment, or both simultaneously. Methods utilizing controlled RGB [Jeong et al., 2015] and RGB-D images [Chen et al., 2015] have been presented, that select and refine 3D garment templates based on image observations. While [Jeong et al., 2015] utilizes a tailor’s dummy, [Chen et al., 2015] expects the subject to hold a certain pose. Other methods focus on recovering the shape and detailed garment wrinkles of clothing items in less controlled settings [Popa et al., 2009, Daněřek et al., 2017, Bednarik et al., 2018, Jin et al., 2018, Lahner et al., 2018]. While these methods can produce detailed geometry, none of them focuses on the appearance of the item. Another branch of research aims at reconstructing the whole human in 3D, including clothing. This can be achieved by optimization-based [Alldieck et al., 2018b,a, Weng et al., 2019] or learning-based methods [Alldieck et al., 2019a, Lazova et al., 2019] that utilize silhouettes or semantic segmentation of a short monocular video clip or recently even from single images [Natsume et al., 2019, Alldieck et al., 2019b, Saito et al., 2019, Gabeur et al., 2019] and point clouds [Chibane et al., 2020]. Other methods utilize KinectFusion-like approaches [Izadi et al., 2011, Newcombe et al., 2011] to scan people using RGB-D sensors [Shapiro et al., 2014, Li et al., 2013, Zeng et al., 2013, Cui et al., 2012]. Having a 3D reconstruction of body and clothing, it can be used to non-rigidly track the subject [Xu et al., 2018, Habermann et al., 2019]. All these methods fuse body and clothing in a single template. Virtual try-on applications, however, often require separate meshes [Hauswiesner et al., 2011]. Therefore, methods that reconstruct the naked body shape or both body shape and clothing have been developed. The naked body shape alone has been estimated using several RGB images [Bälan and Black, 2008] or more accurately using a sequence of clothed scans [Zhang et al., 2017]. Body shape and garments have been reconstructed simultaneously and fully automatically from a series of scans [Pons-Moll et al., 2017], RGB-D images [Tao et al., 2019], and recently even from a small set of RGB images

[Bhatnagar et al., 2019]. Yang et al. [Yang et al., 2018b] present garment and body-shape recovery from a single image but heavily rely on physical priors and human interaction. In order to enable dynamic virtual try-on and clothing re-targeting, joint models of clothing and the human body have been developed [Neophytou and Hilton, 2014, Yang et al., 2018a, Patel et al., 2020, Ma et al., 2020]. Again, all these works focus mainly or exclusively on the geometry of the garment, not on their appearance. Other works also learn to predict correspondences from depth maps to surfaces [Taylor et al., 2012, Pons-Moll et al., 2015, Wei et al., 2016] and from images to surfaces [Kulkarni et al., 2019, Güler et al., 2018, Grigorev et al., 2019], but they address different problems.

Automatic Texturing. Automatic texturing of 3D models from photos has been presented, but the shape has to be first aligned with the input image [Park et al., 2018, Wang et al., 2016]. This alignment is expensive and error-prone as silhouette and feature cues may be ambiguous. The most related work to Ch. 4 is [Chen et al., 2016], which maps texture from clothing items to the 3D anguelov2005scape [Anguelov et al., 2005] body model. Their focus is not photorealistic mapping, but rather to generate synthetic training data with texture variation to learn discriminative detectors. Their core texture mapping is based on 2D image warping. In contrast to all previous work, the method presented in Ch. 4 creates textured 3D garments fully automatically, without requiring prior alignment at test time, enabling virtual try-on and novel-view synthesis in real time.

2.2 BODY-MOUNTED SENSORS

IMU-Based 3D Human Pose Estimation. Although commercial solutions for IMU-based pose estimation have improved the stability of earlier solutions [Roetenberg et al., 2007], they still suffer from severe drift, especially in the global orientation and location of the body. Early work [Vlasic et al., 2007] developed a custom suit to capture 3D human pose during daily activities. One line of work has focused on reducing the amount of IMUs necessary to capture motion via space-time optimization [von Marccard et al., 2017] or with deep learning [Huang et al., 2018]. In order to reduce drift and improve accuracy, visual-inertial approaches combine IMUs with multiple external cameras [von Marccard et al., 2016, Pons-Moll et al., 2010, Trumble et al., 2017, Malleson et al., 2017], a depth camera [Helten et al., 2013, Zheng et al., 2018], or even a single hand-held RGB camera [von Marccard et al., 2018]—which enabled collecting the 3DPW [von Marccard et al., 2018] dataset with accurate 3D poses outdoors. However, these approaches all require an external camera, which limits the field of view that can be captured or requires someone to follow the person being tracked. In Ch. 5, this limitation is addressed by mounting an outward-facing camera on the head (approximating the person’s field of view) and using it to self-localize the wearer in a pre-scanned 3D scene.

Ego-Centric Capture and Prediction. In contrast to the outward-facing setup of Ch. 5, most ego-centric body-capture approaches mount the camera on the head looking

towards the body. While ego-centric capture has received considerable attention for activity recognition [Bhatnagar et al., 2017, Fathi et al., 2011, Ma et al., 2016, Cao et al., 2017, Yonemoto et al., 2015, Rogez et al., 2015], methods at most detect the upper body. For full-body capture, a pioneering method [Rhodin et al., 2016] relied on a helmet with sticks holding a camera away from the body. More recent methods [Xu et al., 2019, Tome et al., 2020] work reasonably well even when the camera is close to the head; however, their overall accuracy is still far from desired.

Another group of methods place the camera looking outwards (as in Ch. 5) and aim at estimating 3D pose from the ego-centric view alone, but 3D poses are inaccurate and have high uncertainty [Jiang and Grauman, 2017, Yuan and Kitani, 2018, 2019]. An alternative approach places many cameras on the body looking out and uses multi-camera structure from motion [Shiratori et al., 2011], but it can only recover slow motions.

Camera Localization. Most 6-DoF camera localization algorithms can be split into three groups. The first group is *structure-based* [Svärm et al., 2016, Toft et al., 2018, Sattler et al., 2016, Liu et al., 2017, Taira et al., 2018], which matches 2D keypoints on the query image with 3D scene keypoints to estimate the camera pose by minimizing reprojection error. While they provide precise localization in small scenes, they do not scale to large scenes as matching becomes ambiguous and computationally expensive.

The second group is *image-based*. The idea is to retrieve nearest neighbors in an image database using a global descriptor [Arandjelovic et al., 2016, Torii et al., 2018, Weyand et al., 2016]. These methods are more robust and scalable compared to structure-based methods, but less precise, and the quality depends on the size of the image database.

The third group comprises *hybrid approaches* [Sarlin et al., 2019, 2018] which combine the benefits of the previous two: first, a set of relevant database images is found using an image-based method, and then the precise camera pose is recovered using structure-based matching. Another set of methods directly regress the camera pose using a CNN [Sattler et al., 2019, Walch et al., 2017], but their accuracy is limited. Hybrid approaches have been shown to be both precise and scalable to large scenes, and the self-localization component of Ch. 5 builds upon them.

2.3 MOTION SYNTHESIS AND ANIMATION

Human Motion Prediction without the 3D Scene. Predicting the dynamics of human motion is a long-studied problem in computer vision and graphics. Classic works explored using Hidden Markov Chains [Brand and Hertzmann, 2000] and Gaussian Processes [Wang et al., 2007], as well as physics-based models [Liu et al., 2005] for predicting future motion. More recently, recurrent neural networks [Graves, 2013, Hochreiter and Schmidhuber, 1997] have been used for motion prediction [Fragkiadaki et al., 2015, Martinez et al., 2017, Alahi et al., 2016], also in combination with Graph Neural Networks [Kipf and Welling, 2017, Mao et al., 2019, Li et al., 2020b, Dang et al., 2021] and variational auto-encoders [Kingma and Welling, 2014] to add diversity

[Habibie et al., 2017, Zhang et al., 2021b, Yuan and Kitani, 2020]. An intrinsic problem of recurrent methods is that they drift over time [Aksan et al., 2020].

More recent approaches employ transformers to generate unconditional or text- and music-conditioned motion sequences [Aksan et al., 2020, Li et al., 2021, 2020a, Petrovich et al., 2021, 2022]. Motion inbetweening [Duan et al., 2021, Harvey et al., 2020, Oreshkin et al., 2022, Yuan et al., 2022, Aksan et al., 2019, Kaufmann et al., 2020] is another classic paradigm for motion synthesis where the task is to fill in frames between animator-provided keyframes. The method presented in Ch. 6 builds on recent progress in transformer architectures [Li et al., 2021] and classical ideas such as motion inbetweening, combined with the novel idea of a canonical coordinate frame and action keypoint representation in order to generate motion in real 3D scenes.

Character Control in Video Games. Motion matching [Reitsma and Pollard, 2007], its learned variant [Clavet, 2016, Holden et al., 2020], and motion graphs [Lee et al., 2002, Fang and Pollard, 2003, Kovar et al., 2002b, Safonova et al., 2004, Safonova and Hodgins, 2007] are classical methods often employed in the industry for generating kinematic motion sequences, controlled by environment and user-specified constraints. With a similar goal to Ch. 6, some works [Puig et al., 2018, Cao et al., 2020] use a combination of these approaches and inverse kinematics (IK) to generate human behaviors in synthetic scenes. However, these approaches require significant human effort to author realistic animations, and IK approaches easily produce non-realistic results.

Deep learning variants such as Holden *et al.* [Holden et al., 2017] introduce phase-conditioning in a RNN to model the periodic nature of walking motion. In several works by Starke *et al.* [Starke et al., 2019, 2021, 2020] the idea of local phases is extended to synthesize scene-aware motion, basketball motion, and martial arts motion. All these methods generate convincing motion but phases are non-intuitive for non-periodic motion and often require manual labelling.

Static Human Pose Conditioned on Scenes. The relationship between humans, scenes, and objects is another recurrent subject of study in computer vision and graphics. Classical works include methods for 3D object detection [Gupta and Davis, 2007, Gupta et al., 2011] and affordance prediction using human poses [Delaitre et al., 2012, Grabner et al., 2011, Fouhey et al., 2014]. Several recent works generate plausible static poses conditioned on a 3D scene [Li et al., 2019a, Zhang et al., 2021b, Wang et al., 2017, Zhang et al., 2020b, Hassan et al., 2021b, Zhao et al., 2022] using recently captured human interaction datasets [Hassan et al., 2019, Savva et al., 2016, Bhatnagar et al., 2022, Taheri et al., 2020, Cao et al., 2020]. Instead of static poses, Ch. 6 generates *motion* coherent with the scene, which is significantly harder.

Scene-Aware Motion Synthesis. Some works leverage reinforcement learning to synthesize navigation in *synthetic* 3D scenes [Ling et al., 2020, Zhang and Tang, 2022]. Other works focus on a single action, such as grabbing [Taheri et al., 2022, Wu et al., 2022], but do not demonstrate transitions to new motions. These methods are not demonstrated in real 3D scenes with multiple objects and clutter. Recent real interaction

datasets [Hassan et al., 2019, Savva et al., 2016, Bhatnagar et al., 2022, Taheri et al., 2020, Cao et al., 2020] have powered methods to synthesize 3D scene-aware motion [Wang et al., 2022, 2021a, Cao et al., 2020, Wang et al., 2021b]. These datasets are crucial to drive progress, but do not capture the richness and variety of real-world scenes. Hence, these methods are often demonstrated only on small scenes from PROX [Hassan et al., 2019] and Matterport [Chang et al., 2017].

Ch. 6 draws inspiration from Hassan *et al.* [Hassan et al., 2021a], who combine path planning with neural motion synthesis, and from Zhang *et al.* [Zhang et al., 2022], who synthesize contact-controlled human–chair interaction. These methods require the geometry of the isolated interacting object as input, which makes them hard to generalize to real 3D scenes. Unlike these methods, Ch. 6 demonstrates *long chained sequences of actions in generic real 3D scenes*, enabled by origin canonicalization and action keypoints.

Human Animation with Learned Controllers. Beyond classical motion matching [Reitsma and Pollard, 2007, Clavet, 2016, Holden et al., 2020] and motion graphs [Lee et al., 2002, Fang and Pollard, 2003, Kovar et al., 2002a, Safonova et al., 2004, Safonova and Hodgins, 2007], diffusion models [Tevet et al., 2023] have emerged as a powerful paradigm for motion synthesis, with extensions for physics-aware generation [Yuan et al., 2023], blended-positional encoding [Barquero et al., 2024], and fine-grained control [Karunratanakul et al., 2024, Pinyoanuntapong et al., 2025, Xie et al., 2024b]. Reinforcement learning [Zhang and Tang, 2022, Zhao et al., 2023] provides another widely used paradigm for scene-aware locomotion. Latent diffusion motion models [Zhao et al., 2025a] combine RL with latent-space optimization. However, all these methods operate on clean mesh-based scenes and do not address photorealistic rendering of human–scene interactions, which is addressed in Chs. 8 and 7 of this thesis.

2.4 HUMAN-SCENE AND HUMAN-OBJECT INTERACTION

Affordances and Interaction Datasets. The relationship between humans, scenes, and objects is a recurrent subject of study in computer vision and graphics. Classical works include methods for 2D pose and object detection [Desai and Ramanan, 2012, Yao and Fei-Fei, 2010, Iqbal et al., 2017, Pishchulin et al., 2013, Kjellström et al., 2011, Gupta and Davis, 2007], 3D object detection using human poses [Grabner et al., 2011, Gupta et al., 2011], learning to insert people in scenes [Fouhey et al., 2014, Wang et al., 2017, Li et al., 2019a, Zhang et al., 2020c], constraining pose estimation with scene geometry [Zanfir et al., 2018a, Hassan et al., 2019], and estimating forces [Li et al., 2019b] or predicting long-term motion [Cao et al., 2020] conditioned on the scene.

Large-scale interaction datasets have been crucial to driving progress. PROX [Hassan et al., 2019] captures humans moving in 12 indoor scenes with fitted SMPL-X bodies, using scene contact and penetration constraints during pose estimation. GRAB [Taheri et al., 2020] captures whole-body grasping of 51 everyday objects with articulated hand and face motion via SMPL-X. BEHAVE [Bhatnagar et al., 2022] provides multi-view RGBD frames with fitted body meshes, object meshes, and annotated contact maps for

20 objects. COUCH [Zhang et al., 2022] focuses on contact-controlled human–chair interaction with fine-grained hand contact annotations. PiGraphs [Savva et al., 2016] records interaction snapshots encoding physical contact and visual attention linkages between body parts and scene geometry. AMASS [Mahmood et al., 2019] unifies 15 optical marker-based motion capture datasets into a single SMPL representation, providing over 40 hours of motion data, while HumanML3D [Guo et al., 2022] pairs 14,616 motion clips with 44,970 natural language descriptions for text-conditioned motion generation. Further datasets [Hassan et al., 2021a, Jiang et al., 2024a, Cheng et al., 2023, Punnakal et al., 2021, Cao et al., 2020] extend coverage to additional scenes, actions, and modalities.

Several recent works generate plausible static poses conditioned on a 3D scene. PSI [Li et al., 2019a] fuses semantic knowledge from 2D poses with 3D geometric knowledge from voxel scene representations to predict where and how humans can be placed in indoor scenes. PLACE [Zhang et al., 2020b] models the proximity between body surface and scene geometry via a conditional VAE, generating SMPL-X bodies whose contact regions match the predicted proximity field. COINS [Zhao et al., 2022] synthesizes static human–scene interactions controlled by compositional semantic specifications (e.g., , “sit on the chair and hold the cup”) using a transformer that jointly encodes body surface points and object geometry. Further methods [Zhang et al., 2021b, Wang et al., 2017, Hassan et al., 2021b] explore similar directions. Most approaches predict only static poses in a single room and reason from a third-person perspective.

Joint Human–Object Reconstruction. Joint 3D reconstruction of human–object interactions from RGB observations has been addressed by several recent methods. PHOSA [Zhang et al., 2020a] recovers plausible 3D spatial arrangements of humans and objects from a single in-the-wild image without 3D supervision, using a scale loss, an occlusion-aware silhouette reprojection loss, and a spatial-layout interaction loss that provides “3D common sense” to resolve depth ambiguity. CHORE [Xie et al., 2022a] jointly reconstructs human and object geometry plus their contacts from a single image by representing both as implicit unsigned distance fields with correspondence and object-pose prediction. VisTracker [Xie et al., 2023] extends this to video by jointly tracking humans, objects, and contacts across frames, using a transformer that leverages object visibility and human motion from neighboring frames to handle heavy occlusion. Xie *et al.* [Xie et al., 2024a] further remove the need for object templates by training a hierarchical diffusion model on procedurally generated interaction data, reconstructing unseen object instances without known CAD models. These methods rely on mesh representations and thus cannot generate photorealistic images. In Ch. 7, we address this gap by jointly reconstructing humans and objects as separate controllable 3DGS models, enabling photorealistic rendering under occlusion.

Object-Conditioned and Scene-Aware Interaction Synthesis. Object-conditioned motion synthesis methods generate human motion conditioned on object geometry. Hassan *et al.* [Hassan et al., 2021a] combine path planning—computing collision-free navigation paths to target objects—with stochastic neural motion synthesis that models

different styles of performing actions such as sitting and lying down. Zhang *et al.* [Zhang *et al.*, 2022] synthesize contact-controlled human–chair interaction, where users specify hand contact positions on the object surface and the method predicts contact-aware control signals to guide the motion. CG-HOI [Diller and Dai, 2024] is the first method to generate dynamic 3D human–object interaction sequences from text descriptions and object geometry, jointly modeling human motion, object motion, and contact in a single diffusion process. Starke *et al.* [Starke *et al.*, 2019] use local phase-conditioned networks for scene-aware motion synthesis. Reinforcement learning approaches [Zhang and Tang, 2022, Zhao *et al.*, 2023] formulate scene-aware behavior as a Markov decision process with latent action spaces learned from motion capture data, enabling virtual humans to navigate 3D scenes and interact with objects. Latent diffusion motion models [Zhao *et al.*, 2025a] combine diffusion-based autoregressive motion generation with latent-space optimization and RL-based control for spatial constraints, achieving real-time generation. However, all these methods operate on mesh-based scenes and cannot produce photorealistic renderings. Chs. 7 and 8 of this thesis address human–object and human–scene interaction entirely within the 3DGS framework.

2.5 NEURAL RENDERING FOR HUMANS AND SCENES

Neural Radiance Fields and 3D Gaussian Splatting. NeRF [Mildenhall *et al.*, 2020] represents scenes as a continuous 5D function mapping spatial coordinates and viewing direction to volume density and view-dependent color, parameterized by an MLP and rendered via differentiable volume rendering along camera rays. This approach sparked wide interest in neural rendering [Xie *et al.*, 2022b], yet the per-scene optimization cost and slow ray-marching inference remained practical limitations. Follow-up works addressed this through multi-resolution hash encoding for near-instant training [Müller *et al.*, 2022], anti-aliased rendering for unbounded 360-degree scenes [Barron *et al.*, 2022], and their combination for both quality and speed [Barron *et al.*, 2023, Tancik *et al.*, 2023].

3D Gaussian Splatting (3DGS) [Kerbl *et al.*, 2023] takes a fundamentally different approach by representing scenes as collections of explicit anisotropic 3D Gaussian primitives rendered via differentiable splatting [Westover, 1992], extending earlier sphere-based rendering [Lassner and Zollhöfer, 2021] (see Ch. 3 for a detailed description of the 3DGS representation and rendering pipeline). This explicit representation enables real-time rendering at quality comparable to or exceeding the best NeRF methods, while also enabling significantly faster optimization. Originally designed for static scenes, 3DGS has been extended to dynamic settings by allowing Gaussians to move and rotate over time with local-rigidity constraints [Luiten *et al.*, 2024], modeling 4D scenes via neural voxel deformations [Wu *et al.*, 2024], and other temporal decompositions [Shaw *et al.*, 2024, Lee *et al.*, 2024, Li *et al.*, 2024b]. Further extensions address SLAM-based reconstruction [Keetha *et al.*, 2024], mesh extraction [Huang *et al.*, 2024a, Guédon and Lepetit, 2024], novel-view synthesis from sparse views via neural field regularization of Gaussian attributes [Mihajlovic *et al.*, 2024] and embodied

views [Wang et al., 2025f], and ray-traced rendering [Moenne-Loccoz et al., 2024, Wu et al., 2025b, Byrski et al., 2025, Bi et al., 2024]. Chs. 7–10 of this thesis build upon 3DGS as the core scene and human representation.

Human Reconstruction with Neural Rendering. Parametric mesh templates such as SMPL [Loper et al., 2015] and SMPL-X [Pavlakos et al., 2019] recover 3D human shape and pose from images [Bogo et al., 2016, Kanazawa et al., 2018], and follow-up works refine appearance through learned texture maps and deformation offsets [Alldieck et al., 2018b, 2019a, Mir et al., 2020, Shin et al., 2024], but mesh-based representations fundamentally lack the capacity for photorealistic rendering of hair, clothing folds, and fine-grained surface detail. Dense body representations such as DensePose [Güler et al., 2018, Neverova et al., 2018, 2020] provide pixel-level correspondences between image pixels and a UV parameterization of the body surface, enabling detailed human understanding but not 3D reconstruction on their own.

Implicit functions [Mescheder et al., 2019, Park et al., 2019] offer an alternative: PIFuHD [Saito et al., 2020] reconstructs detailed clothed humans from a single image via pixel-aligned implicit occupancy predictions at multiple resolutions, SNARF [Chen et al., 2021] enables animation of implicit shapes via differentiable forward skinning with iterative root-finding in canonical space, and further works recover articulated bodies with varying levels of detail [Alldieck et al., 2021, He et al., 2021, Huang et al., 2020, Deng et al., 2020]. While these methods capture fine geometric detail, they still lack photorealism and are often not reproducible to novel poses.

NeRF-based methods address photorealism by learning volumetric radiance fields of humans. HumanNeRF [Weng et al., 2022] learns a canonical T-pose representation from monocular video together with a motion field decomposed into skeletal and non-rigid components, enabling free-viewpoint rendering of any frame. Peng et al. [Peng et al., 2021] introduce neural blend weight fields that constrain observation-to-canonical warps via skeleton transformations, enabling animation with novel poses. Further works extend this to video-based avatar optimization [Guo et al., 2023], monocular [Jiang et al., 2022b] and multi-view [Habermann et al., 2023, Zhu et al., 2024a] settings, articulated representations [Li et al., 2022, Liu et al., 2021, Xu et al., 2021], and category-level generalization [Iqbal et al., 2023], but all inherit NeRF’s speed limitations.

Recent 3DGS-based approaches overcome this speed bottleneck by building controllable human avatars with real-time rendering. HUGS [Kocabas et al., 2024] disentangles a static scene from an animatable human from monocular video in 30 minutes using SMPL-initialized Gaussians, while 3DGS-Avatar [Qian et al., 2024] optimizes canonical-space Gaussians with pose-conditioned non-rigid deformations and as-isometric-as-possible regularization for novel-pose generalization. GauHuman [Hu et al., 2024b] achieves 1–2 minute training with KL-divergence-guided Gaussian splitting and merging. Other works address drivable face and body avatars via tetrahedral cage deformations [Zielonka et al., 2025], expressive whole-body avatars combining SMPL-X mesh topology with Gaussians [Moon et al., 2024], and further extensions spanning relightable [Li et al., 2024c], animatable [Li et al., 2024d, Pang et al., 2024, Lei et al., 2024], and generalizable [Zheng et al., 2024, Qiu et al., 2025c] avatars, as well as

face-specific models [Dhamo et al., 2024, Xu et al., 2024b] and robust fitting [Guo et al., 2025, Moreau et al., 2024, Abdal et al., 2024, Jiang et al., 2024b, Junkawitsch et al., 2025, Moreau et al., 2025]. Extensions that model humans with their environment [Xue et al., 2024, Zhan et al., 2025b] assume the scene is static or that objects are an extension of the human hand. This thesis addresses several gaps: human–object interaction in 3DGS scenes (Ch. 7), human animation in 3DGS scenes (Ch. 8), shadow casting for composited avatars (Ch. 9), and avatar reconstruction from occluded video (Ch. 10).

2.6 INVERSE RENDERING, SHADOWS, AND RELIGHTING

Inverse Rendering, Shadows, and Relighting in 3DGS. Several methods decompose 3DGS scenes into materials and lighting to enable relighting. GS-IR [Liang et al., 2024] recovers self-shadows via baked occlusion with depth-based normal regularization, while Gao *et al.* [Gao et al., 2024] augment each Gaussian with normal vectors and BRDF parameters and use point-based ray tracing with bounding volume hierarchies for precomputed visibility, enabling physically-based shadow effects. GaussianShader [Jiang et al., 2024c] introduces a simplified shading function with environment lighting maps for reflective surfaces, GS-IR [Gu et al., 2025] employs differentiable ray tracing for full inverse rendering, and further approaches use deferred shading [Chen et al., 2025a], learned shadow direction parameters [Du et al., 2025, Zheng et al., 2026], or screen-space ray tracing [Wu et al., 2025c]. ShadowGS [Luo et al., 2026] applies physics-based shadow modeling to satellite imagery. Existing shadow casting for Gaussians [Bolanos et al., 2024]—which casts secondary shadow rays using a Gaussian density proxy—does not handle shadows *received* in 3DGS scenes, and relighting methods [Chen et al., 2024a] do not address dynamic avatars. All these approaches are per-scene optimizations on static geometry that cannot handle dynamic avatars or composited objects. In Ch. 9, we address this gap by casting shadows from animated 3DGS avatars onto 3DGS scenes at composition time, without mesh reconstruction.

Human Relighting. Portrait relighting methods trained on light-stage data [Sun et al., 2019, Kanamori and Endo, 2018] capture subjects under diverse controlled directional illumination, enabling neural networks to relight novel portraits from a single image. TotalRelighting [Pandey et al., 2021] extends this to a full pipeline of foreground estimation, per-pixel relighting with explicit diffuse and specular decomposition, and background compositing. SwitchLight [Kim et al., 2024] introduces a physics-driven architecture based on the Cook-Torrance reflectance model with self-supervised pre-training to overcome scarce light-stage training data. More recently, geometry-aware approaches [Ji et al., 2022] and diffusion-based methods [He et al., 2024a, Ho et al., 2020, Rombach et al., 2022, Song et al., 2021b,a] have emerged; DiffRelight [He et al., 2024a] trains a subject-specific diffusion model on one-light-at-a-time captures to accurately reproduce eye reflections, subsurface scattering, and self-shadowing for dynamic facial performances. Inverse-rendering approaches recover relightable avatars by decomposing appearance into materials and illumination [Hasselgren et al., 2022,

Jin et al., 2023, Guo et al., 2019, Chen and Liu, 2022, Saito et al., 2024, Iwase et al., 2023, Chen et al., 2024b, Zhang et al., 2021a, Sarkar et al., 2023, Wang et al., 2025c], recently extended with Gaussian-based relighting that learns pose-dependent Gaussian maps with physically-based material decomposition [Li et al., 2024c,a, Zhan et al., 2025a, Zhao et al., 2025b, Fan et al., 2025] and subsurface scattering [Dihlmann et al., 2024]. These methods study relighting in isolation; Ch. 9 of this thesis focuses instead on shadow casting for avatars composited into 3DGS scenes with temporal consistency.

2.7 MONOCULAR HUMAN RECONSTRUCTION AND GENERATIVE PRIORS

Optimization and Feed-Forward Methods. Early implicit methods reconstruct clothed humans from single images: PIFu [Saito et al., 2019] introduces pixel-aligned implicit functions that align 2D image features with 3D occupancy and surface predictions, ECON [Xiu et al., 2023] combines implicit and explicit representations via front/back normal maps with body-conditioned completion, and SIFU [Zhang et al., 2024] uses a side-view decoupling transformer with SMPL-X normals for high-quality reconstruction. Further implicit approaches [Zheng et al., 2021, Huang et al., 2024b] improve reconstruction fidelity. Optimization-based methods reconstruct from monocular video: SelfRecon [Jiang et al., 2022a] combines implicit and explicit representations with a consistency loss between both for space-time coherent clothed geometry, and GauHuman [Hu et al., 2024c] learns articulated Gaussian avatars with 1–2 minute training via SMPL-based canonical encoding and KL-divergence-guided Gaussian control.

Feed-forward models predict animatable avatars in seconds without per-scene optimization. LHM [Qiu et al., 2025a] is a large-scale transformer that encodes both body positional features anchored on SMPL-X surface samples and image features via attention, with a head feature pyramid for multi-scale facial detail. IDOL [Zhuang et al., 2025] disentangles human pose, shape, clothing geometry, and texture into a structured Gaussian representation that is directly animatable and editable, producing avatars in under one second. Persona [Sim and Moon, 2025] generates pose-rich videos from the input image via diffusion, then optimizes a 3D avatar with balanced sampling and geometry-weighted optimization for realistic clothing deformations. Further feed-forward approaches [Qiu et al., 2025b, Hong et al., 2024] extend this paradigm.

Text-to-3D methods generate humans from language prompts. DreamFusion [Poole et al., 2023] introduces Score Distillation Sampling (SDS), using a pretrained text-to-image diffusion model as a loss function to optimize a NeRF from text. DreamHuman [Kolotouros et al., 2023] generates animatable 3D human avatars from text by combining text-to-image diffusion with a deformable, pose-conditioned NeRF constrained by an implicit statistical body model, producing avatars that can be reposed at test time. DreamAvatar [Cao et al., 2024] introduces a dual-observation-space design with joint optimization in canonical and posed spaces for SMPL-guided text-to-3D human generation. These text-to-3D methods cannot reconstruct from visual obser-

variations. All the methods above assume a fully visible subject as input, excluding the vast majority of real-world footage where people are routinely occluded. In Ch. 10, we address this gap by recovering animatable avatars from occluded in-the-wild YouTube videos.

Diffusion-Based Multi-View Synthesis for 3D Humans. Many recent methods leverage diffusion models to synthesize multi-view images as an intermediate step for 3D human reconstruction from a single image. HumanSplat [Pan et al., 2024] predicts 3DGS properties in a feed-forward manner by combining a multi-view diffusion model with a latent reconstruction transformer that incorporates human structure priors and hierarchical losses for fine detail in faces and hands. CharacterGen [Peng et al., 2024] first canonicalizes the input body pose to A-pose while generating consistent multi-view images, then reconstructs a 3D mesh via a transformer-based sparse-view model, enabling downstream rigging and animation. SyncHuman [Chen et al., 2025b] jointly fine-tunes a 2D multi-view generative model and a 3D native generative model with pixel-aligned synchronization attention to produce geometrically consistent shapes and images. PSHuman [Li et al., 2025] jointly models full-body shape and local facial characteristics via cross-scale multi-view diffusion conditioned on SMPL-X, preventing geometric distortions common in face regions. AniGS [Qiu et al., 2025c] adapts a video generation model to synthesize multi-view canonical-pose images and recasts reconstruction as a 4D task using 4D Gaussian Splatting to handle view inconsistencies. AdaHuman [Huang et al., 2025] jointly synthesizes consistent multi-view images and 3D Gaussian Splats at each diffusion step via a pose-conditioned 3D joint diffusion model, with a compositional refinement module for local body part details. Further methods address additional aspects such as magic-view synthesis [He et al., 2024b], multi-view generation with Gaussian aggregation [Lu et al., 2025], animated reconstruction [Buehler et al., 2026, Dong et al., 2025], and temporal consistency [Taubner et al., 2025]. Extensions to 4D and unconstrained settings include DressRecon [Tan et al., 2025], which disentangles body versus clothing deformations as separate motion model layers from monocular video of loose clothing, and Disco4D [Pang et al., 2025], which disentangles clothing Gaussians from the body model for compositional garment extraction and 4D animation. Further 4D methods [Shao et al., 2024, Yang et al., 2024, Lin et al., 2025] extend coverage to diverse motion and appearance. These methods produce compelling results but still assume fully visible subjects and cannot handle significant occlusion.

Video Diffusion Models. Denoising diffusion probabilistic models (DDPMs) [Ho et al., 2020] generate images by learning to reverse a forward Gaussian noise process via a parameterized Markov chain. Latent diffusion models [Rombach et al., 2022] dramatically reduce computational cost by operating in the latent space of a pretrained autoencoder, enabling flexible conditioning on text and other modalities via cross-attention layers, and forming the foundation of large-scale text-to-image synthesis. Further image generation architectures [Saharia et al., 2022, Peebles and Xie, 2023, Esser et al., 2024, Imagen Team Google, 2024] extend this paradigm. Video diffusion models extend image diffusion with temporal modeling: Stable Video Diffusion

(SVD) [Blattmann et al., 2023] adds temporal convolution and attention layers trained on a curated large video dataset, CogVideoX [Yang et al., 2025] uses a 3D VAE for joint spatial-temporal compression with an expert transformer for deep text-video fusion, and Sora [Brooks et al., 2024, OpenAI, 2025] operates on spacetime patches to generate high-fidelity video at variable durations and resolutions. Further video generation models [Wan Team, 2025, Kong et al., 2024, ByteDance Seed Team, 2025a, Wu et al., 2025a, ByteDance Seed Team, 2025b, 2026] continue scaling this approach.

A large body of work specializes video diffusion for human-centric synthesis. Pose-driven character animation methods such as Animate Anyone [Hu et al., 2024a, 2025]—which uses a ReferenceNet for appearance preservation and a pose guider for motion control—and MagicAnimate [Xu et al., 2024d]—which uses an appearance encoder with video fusion for smooth long-video generation—have achieved strong results on fashion and dance benchmarks. Champ [Zhu et al., 2024b] conditions on 3D parametric SMPL guidance (rendered depth, normals, and semantic maps) for improved shape alignment, and MimicMotion [Zhang et al., 2025] introduces confidence-aware pose guidance to adaptively weight joint influence. Further methods [Guo et al., 2024b, Wang et al., 2025d,e, Gan et al., 2025, Zhou et al., 2025, Luo et al., 2025, Men et al., 2025, Tu et al., 2025, Chang et al., 2024, Xu et al., 2024c] extend pose-driven animation. Portrait animation methods such as LivePortrait [Guo et al., 2024a]—which discovers compact implicit keypoints from large-scale training for efficient 12.8 ms per-frame inference—and VASA [Xu et al., 2024a]—which generates audio-driven talking faces with disentangled facial dynamics at up to 40 FPS—enable real-time face animation. Further portrait methods [Xie et al., 2024c, Wei et al., 2024, Xu et al., 2025, Ma et al., 2024] and video editing approaches [Jiang et al., 2025] build on this paradigm, extending earlier warping-based methods [Siarohin et al., 2019, Zhao and Zhang, 2022] that modeled motion via learned keypoints and local affine transformations.

LoRA-based personalization [Abdal et al., 2025b,a, Dravid et al., 2024] enables identity-specific finetuning of video diffusion models by injecting trainable low-rank matrices into transformer layers, reducing trainable parameters by orders of magnitude while producing compact adapter weights. Inversion techniques [Abdal et al., 2019, 2020, Mokady et al., 2023, Wang et al., 2025b, Rout et al., 2025] allow embedding real videos into the model’s latent space for faithful reconstruction and editing; RF-Inversion [Rout et al., 2025] uses dynamic optimal control derived via a linear quadratic regulator for efficient inversion and editing with rectified flow models. Video diffusion has also been used as supervision for 3DGS under sparse-view setups [Zhong et al., 2025, Nazarczuk et al., 2025]. In Ch. 10, we leverage video diffusion as a prior for 3DGS-based avatar reconstruction from occluded monocular video.

BACKGROUND

This chapter provides the technical foundations for the representations and algorithms used throughout this thesis. We first introduce Linear Blend Skinning (Sec. 3.1), a classical technique for skeleton-driven mesh deformation. We then describe SMPL (Sec. 3.2), a learned statistical body model built on LBS that parametrizes human shape and pose. We present 3D Gaussian Splatting (Sec. 3.3), an explicit neural rendering method that represents scenes as collections of anisotropic 3D Gaussian primitives. Finally, we compare meshes and 3DGS (Sec. 3.4), highlighting the differences in surface representation, rendering quality, reconstruction requirements, and animation that motivate the two-part structure of this thesis. SMPL serves as the human body representation in Chs. 4–10, while 3DGS is the core scene and human rendering representation in Chs. 7–10.

3.1 LINEAR BLEND SKINNING

Linear Blend Skinning (LBS) is the standard technique for deforming a 3D mesh according to an articulated skeleton. It is widely used in computer graphics, animation, and game engines due to its simplicity, efficiency, and compatibility with GPU rasterization pipelines. LBS serves as the deformation model in SMPL (Sec. 3.2) and in the 3DGS-based avatar methods presented in Chs. 7–10.

Skeleton and Kinematic Tree. An articulated body is represented by a skeleton of J joints organized in a kinematic tree rooted at the pelvis. Each joint j has a parent $\text{pa}(j)$ in the tree (except the root), a rest-pose location $\mathbf{j}_j \in \mathbb{R}^3$, and a local rotation $\mathbf{R}_j \in \text{SO}(3)$ specified by the pose parameters $\boldsymbol{\theta}$. The world-space transformation of joint j is computed by chaining local transformations from the root to j along the kinematic tree:

$$\mathbf{G}_j(\boldsymbol{\theta}, \mathbf{J}) = \prod_{k \in \text{path}(\text{root}, j)} \begin{pmatrix} \mathbf{R}_k(\boldsymbol{\theta}) & \mathbf{j}_k - \mathbf{j}_{\text{pa}(k)} \\ \mathbf{0}^\top & 1 \end{pmatrix}, \quad (3.1)$$

where $\mathbf{j}_k - \mathbf{j}_{\text{pa}(k)}$ is the bone vector from parent to child in the rest pose, and $\mathbf{R}_k(\boldsymbol{\theta})$ is the local rotation of joint k derived from the pose parameters (typically represented as axis-angle vectors). The product traverses joints along the path from root to j , accumulating translations and rotations to obtain the global rigid transformation of each joint.

Blend Weights. Each vertex i of the mesh is associated with a set of non-negative blend weights $\{w_{i,j}\}_{j=1}^J$ satisfying $\sum_{j=1}^J w_{i,j} = 1$. These weights encode the influence of each joint’s transformation on the vertex. In practice, the weights are sparse: each vertex is typically influenced by only a small number of nearby joints. The full set of blend weights for all N vertices is stored in the blend weight matrix $\mathbf{W} \in \mathbb{R}^{N \times J}$.

Skinning Equation. Given a rest-pose mesh with N vertex positions $\{\mathbf{v}_i\}_{i=1}^N$, the LBS function W computes the deformed position of each vertex as a weighted combination of joint transformations:

$$\mathbf{v}'_i = W(\mathbf{v}_i, \boldsymbol{\theta}, \mathbf{J}, \mathbf{W}) = \left(\sum_{j=1}^J w_{i,j} \mathbf{G}_j(\boldsymbol{\theta}, \mathbf{J}) \mathbf{G}_j(\mathbf{0}, \mathbf{J})^{-1} \right) \tilde{\mathbf{v}}_i, \quad (3.2)$$

where $\tilde{\mathbf{v}}_i = (\mathbf{v}_i^\top, 1)^\top$ denotes the homogeneous coordinate. The term $\mathbf{G}_j(\mathbf{0}, \mathbf{J})$ is the rest-pose world-space transform obtained by setting all local rotations to identity, and its inverse removes the rest-pose configuration. The product $\mathbf{G}_j(\boldsymbol{\theta}, \mathbf{J}) \mathbf{G}_j(\mathbf{0}, \mathbf{J})^{-1}$ thus applies only the relative deformation from rest pose to target pose. Intuitively, Eq. (3.2) transforms each vertex into the local coordinate frame of each influencing joint via the inverse rest transform, applies the joint’s target-pose rotation, transforms back to world space, and blends the results according to the weights.

Properties and Limitations. LBS is efficient, fully differentiable, and adds negligible computational overhead, making it the dominant skinning method in both real-time graphics and learning-based avatar methods. However, the linear interpolation of 4×4 transformation matrices can produce artifacts at joints with large rotations. The most prominent artifact is the “candy-wrapper” effect: when a joint such as the wrist rotates significantly around its longitudinal axis, the interpolated transformation causes the mesh to collapse along the rotation axis, producing unrealistic volume loss. Dual quaternion skinning addresses some of these artifacts by interpolating in the space of unit dual quaternions, which preserves rigidity. However, SMPL and most learning-based methods use standard LBS and instead compensate for its limitations through learned pose-dependent corrective blend shapes (Sec. 3.2).

LBS Beyond Meshes: Deforming 3D Gaussians. The LBS formulation above is defined for mesh vertices, but in this thesis we also apply LBS to deform 3D Gaussian primitives (introduced in Sec. 3.3). Unlike mesh vertices, which are fully characterized by a position $\mathbf{v}_i \in \mathbb{R}^3$, each Gaussian carries additional geometric attributes—most importantly a covariance matrix $\boldsymbol{\Sigma}_k \in \mathbb{R}^{3 \times 3}$ encoding its shape and orientation. Deforming a Gaussian therefore requires transforming not only its center $\boldsymbol{\mu}_k$ but also its covariance. The center is deformed exactly as a mesh vertex via Eq. (3.2), blending joint transformations with per-Gaussian skinning weights. The covariance, being a second-order quantity, must be rotated by the blended rotation matrix: $\boldsymbol{\Sigma}'_k = \bar{\mathbf{R}}_k \boldsymbol{\Sigma}_k \bar{\mathbf{R}}_k^\top$, where $\bar{\mathbf{R}}_k = \sum_j w_{k,j} \mathbf{R}_j^{\text{SMPL}}$ is the weighted sum of joint rotations (distinct from the Gaussian’s own rotation $\mathbf{R}_k$ in Eq. (3.9)). This first-order approximation ignores shear effects from blending non-commuting rotations, but is accurate in practice and mirrors

the behavior of standard vertex LBS.

A second distinction is the origin of skinning weights. For meshes, blend weights $\mathbf{W}$ are either hand-painted by artists or learned from registered scans (as in SMPL). For Gaussians, skinning weights can be inherited from the nearest SMPL surface vertex, or predicted by a neural network that learns them jointly with the Gaussian attributes. The latter approach allows the model to discover skinning assignments that better suit the Gaussian representation, particularly for loose clothing or hair that deviates from the body surface. The Gaussian-based chapters of this thesis (Chs. 7–10) present the specific LBS adaptations used in each method, including learned skinning weights, covariance deformation strategies, and the integration of LBS with pose-conditioned neural networks that predict Gaussian attributes.

3.2 SMPL: A SKINNED MULTI-PERSON LINEAR MODEL

SMPL [Loper et al., 2015] is a learned parametric model of the human body that maps shape and pose parameters to a triangular mesh with $N = 6,890$ vertices and 13,776 faces. Learned from thousands of registered 3D body scans, SMPL captures the statistical variation in human body shape and the pose-dependent deformations of soft tissue. It is compatible with standard graphics pipelines and serves as the human body representation throughout this thesis (Chs. 4–10). Its extended version, SMPL-X [Pavlakos et al., 2019], additionally models articulated hands (via the MANO hand model) and an expressive face (via the FLAME head model), increasing the total number of joints and shape parameters.

At a high level, SMPL is a differentiable function:

$$M(\boldsymbol{\beta}, \boldsymbol{\theta}; \Phi) : \mathbb{R}^{|\boldsymbol{\beta}|} \times \mathbb{R}^{|\boldsymbol{\theta}|} \rightarrow \mathbb{R}^{3 \times N}, \quad (3.3)$$

that maps shape parameters $\boldsymbol{\beta} \in \mathbb{R}^{|\boldsymbol{\beta}|}$ and pose parameters $\boldsymbol{\theta} \in \mathbb{R}^{|\boldsymbol{\theta}|}$ to vertex positions, where $\Phi = \{\mathbf{T}, \mathbf{W}, J, B_S, B_P\}$ are the learned model parameters.

Template Mesh and Joints. The model is defined on a mean template mesh $\mathbf{T} \in \mathbb{R}^{3 \times N}$ in a canonical rest pose (T-pose), with $J = 24$ joints organized in a kinematic tree (Sec. 3.1). Each vertex is associated with one of the body parts defined by the skeleton segmentation. Joint locations are not fixed but depend on the body shape: a learned joint regressor J maps from mesh vertices to joint positions:

$$\mathbf{J}(\boldsymbol{\beta}) = J(\mathbf{T} + B_S(\boldsymbol{\beta})), \quad (3.4)$$

where $J \in \mathbb{R}^{J \times N}$ is a sparse regression matrix learned from registered training meshes. This ensures that joint locations move consistently with changes in body proportions—for example, a taller body shape produces wider-spaced hip and shoulder joints.

Shape Blend Shapes. Body shape variation—tall versus short, thin versus heavy, broad versus narrow shoulders—is captured by a linear subspace learned via principal component analysis (PCA) over a large corpus of registered body scans. The shape

blend shapes add identity-dependent vertex offsets to the template:

$$B_S(\beta) = \sum_{n=1}^{|\beta|} \beta_n \mathbf{S}_n, \quad (3.5)$$

where $\{\mathbf{S}_n \in \mathbb{R}^{3N}\}$ are the principal components (orthogonal displacement directions over all vertices) and $\beta = (\beta_1, \dots, \beta_{|\beta|})$ are the corresponding coefficients. Typically, $|\beta| = 10$ components suffice to represent the dominant shape variations, though more components can be used for finer control.

Pose Blend Shapes. When the body articulates, soft tissue deforms in ways that LBS alone cannot capture—muscles bulge, skin stretches, and flesh compresses around joints. SMPL addresses this with pose-dependent corrective blend shapes that add vertex offsets as a function of the current pose:

$$B_P(\theta) = \sum_{n=1}^{9(J-1)} (r_n(\theta) - r_n(\theta^*)) \mathbf{P}_n, \quad (3.6)$$

where $\mathbf{r}(\theta) \in \mathbb{R}^{9(J-1)}$ vectorizes the upper 3×3 entries of the rotation matrices for all non-root joints (each axis-angle pose vector is converted to a rotation matrix and then vectorized), $\theta^* = \mathbf{0}$ is the rest pose, and $\{\mathbf{P}_n \in \mathbb{R}^{3N}\}$ are the learned pose blend shape directions. This formulation predicts vertex corrections based on how far each joint’s rotation deviates from rest, effectively compensating for LBS artifacts such as the candy-wrapper effect at elbows and knees, and modeling soft-tissue bulging at flexed muscles.

Full SMPL Pipeline. Combining the template, shape blend shapes, and pose blend shapes, the final posed mesh is obtained by first computing the rest-pose vertices with all corrections, then applying LBS (Eq. (3.2)):

$$M(\beta, \theta; \Phi) = W \left(\underbrace{\mathbf{T} + B_S(\beta) + B_P(\theta)}_{\text{rest-pose vertices } T(\beta, \theta)}, \mathbf{J}(\beta), \theta, \mathbf{W} \right). \quad (3.7)$$

The pipeline proceeds in four stages (Fig. 3.1): (a) the mean template $\mathbf{T}$ with its joint segmentation and blend weights $\mathbf{W}$; (b) shape blend shapes $B_S(\beta)$ sculpt body proportions and the joint regressor J repositions joints accordingly; (c) pose blend shapes $B_P(\theta)$ add pose-dependent soft-tissue corrections; (d) LBS W articulates the corrected mesh according to the skeleton pose.

Notation Conventions Across the Thesis. Throughout this thesis, different chapters use slightly different subsets of SMPL parameters depending on which quantities are explicitly manipulated. The canonical form defined above is $M(\beta, \theta)$, where the pose vector $\theta \in \mathbb{R}^{3J}$ encodes the axis-angle rotation of all $J = 24$ joints—including the root (pelvis) joint as its first three components—and shape β always precedes pose. The following equivalent forms appear in later chapters:

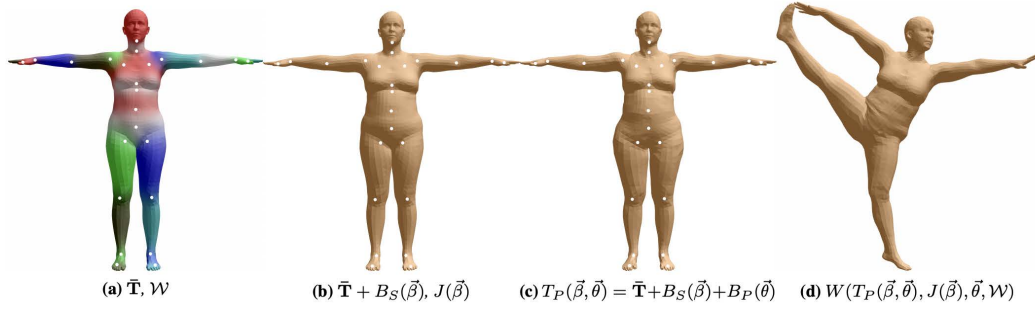


Figure 3.1: **The SMPL body model pipeline.** a) Mean template $\mathbf{T}$ with joint segmentation and blend weights $\mathbf{W}$. b) Shape blend shapes $B_S(\beta)$ sculpt identity-specific body proportions; joint regressor J adapts joint positions. c) Pose blend shapes $B_P(\theta)$ add pose-dependent vertex corrections. d) Linear blend skinning W articulates the corrected mesh. Image source: [Loper et al., 2015].

- **With explicit translation:** $M(\beta, \theta, \mathbf{t})$, where $\mathbf{t} \in \mathbb{R}^3$ denotes the root translation that positions the body in world coordinates (Chs. 5, 6). Translation is omitted from the canonical form above because the SMPL pipeline itself (Eq. (3.7)) produces vertices centered at the root joint; translation is applied as a post-processing step to place the body in the scene.
- **With explicit root orientation:** $\psi \in \mathbb{R}^3$ denotes the global body orientation—i.e., the axis-angle of the root joint, the first three components of the full pose vector. In Ch. 6, ψ is separated from θ in the function signature: $M(\beta, \psi, \theta, \mathbf{t})$, where θ then refers only to the remaining $J-1 = 23$ body joint rotations; this decomposition is useful for motion synthesis, where the walking direction and body articulation must be controlled independently. In Ch. 5, ψ denotes the root component of θ when referenced independently (e.g., for orientation alignment and smoothness), without changing the function signature.
- **With shape dropped:** When shape parameters remain static throughout a sequence, some chapters drop β for notational convenience, writing $M(\theta, \mathbf{t})$ or simply $M(\theta)$ (Chs. 5, 7).

In all cases, **shape β precedes pose θ** when both are present, and **root orientation ψ precedes body pose θ** when they are decomposed. These notational variants all refer to the same underlying SMPL model defined in Eq. (3.7).

3.3 3D GAUSSIAN SPLATTING

3D Gaussian Splatting (3DGS) [Kerbl et al., 2023] is a real-time neural rendering method that represents scenes as collections of explicit anisotropic 3D Gaussian primitives. Unlike Neural Radiance Fields (NeRF) [Mildenhall et al., 2020], which encode scenes as implicit volumetric functions that require costly per-ray sampling through an MLP, 3DGS uses an explicit point-based representation that enables both fast optimization

from multi-view images and real-time rendering via differentiable rasterization. 3DGS serves as the core scene and human representation in Chs. 7–10 of this thesis.

Gaussian Representation. A 3DGS scene consists of N anisotropic 3D Gaussian primitives. Each Gaussian k is defined by five groups of learnable parameters: a mean (center position) $\boldsymbol{\mu}_k \in \mathbb{R}^3$, a covariance matrix $\boldsymbol{\Sigma}_k \in \mathbb{R}^{3 \times 3}$ encoding the shape and orientation of the Gaussian ellipsoid, an opacity $\alpha_k \in [0, 1]$, and view-dependent color represented by spherical harmonic (SH) coefficients $\mathbf{c}_k$. The spatial influence of Gaussian k at a point $\mathbf{x} \in \mathbb{R}^3$ is given by:

$$\mathcal{G}_k(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right). \quad (3.8)$$

The full scene representation is the set $\mathcal{G} = \{(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \alpha_k, \mathbf{c}_k)\}_{k=1}^N$.

Covariance Parameterization. To ensure that $\boldsymbol{\Sigma}_k$ remains positive semi-definite during gradient-based optimization, it is decomposed into a rotation matrix $\mathbf{R}_k \in \text{SO}(3)$ (stored as a unit quaternion $\mathbf{q}_k \in \mathbb{R}^4$) and a per-axis scaling vector $\mathbf{s}_k \in \mathbb{R}_{>0}^3$:

$$\boldsymbol{\Sigma}_k = \mathbf{R}_k \text{diag}(\mathbf{s}_k \odot \mathbf{s}_k) \mathbf{R}_k^\top, \quad (3.9)$$

where $\odot$ denotes the element-wise (Hadamard) product. This factorization allows unconstrained optimization of the quaternion and log-scale parameters while guaranteeing a valid covariance matrix. The three scale values control the extent of the Gaussian along each principal axis, and the quaternion controls the orientation of these axes in world space.

Splatting: Projection to 2D. Rendering a 3DGS scene proceeds by projecting each 3D Gaussian onto the image plane. A 3D Gaussian with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$ in world space is projected to a 2D Gaussian with mean $\boldsymbol{\mu}'_k \in \mathbb{R}^2$ and 2×2 covariance $\boldsymbol{\Sigma}'_k$ on the image plane. Following the EWA (Elliptical Weighted Average) splatting formulation [Westover, 1992] (Fig. 3.2), the projection uses a local affine approximation of the perspective transformation:

$$\boldsymbol{\Sigma}'_k = \mathbf{J}_k \mathbf{V} \boldsymbol{\Sigma}_k \mathbf{V}^\top \mathbf{J}_k^\top, \quad (3.10)$$

where $\mathbf{V} \in \mathbb{R}^{3 \times 3}$ is the rotation component of the world-to-camera viewing transformation and $\mathbf{J}_k \in \mathbb{R}^{2 \times 3}$ is the Jacobian of the perspective projection evaluated at the camera-space position of $\boldsymbol{\mu}_k$. The 2D mean $\boldsymbol{\mu}'_k$ is obtained by standard perspective projection of $\boldsymbol{\mu}_k$.

Alpha Compositing. Given a pixel location $\mathbf{p}$, the rendered color is computed by blending the contributions of all overlapping Gaussians, sorted front-to-back by depth:

$$C(\mathbf{p}) = \sum_{k=1}^N \mathbf{c}_k \alpha'_k \prod_{j=1}^{k-1} (1 - \alpha'_j), \quad (3.11)$$

where the effective per-pixel opacity α'_k combines the learned per-Gaussian opacity α_k with the 2D Gaussian evaluation at the pixel:

$$\alpha'_k(\mathbf{p}) = \alpha_k \exp\left(-\frac{1}{2}(\mathbf{p} - \boldsymbol{\mu}'_k)^\top \boldsymbol{\Sigma}'_k{}^{-1}(\mathbf{p} - \boldsymbol{\mu}'_k)\right). \quad (3.12)$$

The product $T_k = \prod_{j=1}^{k-1} (1 - \alpha'_j)$ is the transmittance, representing the fraction of light that has not been absorbed by Gaussians closer to the camera. This rendering equation is analogous to classical volume rendering used in NeRF, but with the key advantage that the explicit Gaussian primitives avoid the need for per-ray point sampling and MLP evaluation.

Optimization. All Gaussian parameters—positions $\boldsymbol{\mu}_k$, scales $\mathbf{s}_k$, rotations $\mathbf{q}_k$, opacities α_k , and SH coefficients $\mathbf{c}_k$ —are jointly optimized from a set of posed calibrated images by minimizing a photometric reconstruction loss combining $\mathcal{L}_1$ and D-SSIM:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}}, \quad (3.13)$$

with $\lambda = 0.2$. Gaussians are typically initialized from a sparse point cloud obtained via Structure-from-Motion (SfM). The entire rendering pipeline (Eqs. (3.10)–(3.12)) is differentiable, enabling gradient-based optimization of all parameters via backpropagation through the rasterizer.

Adaptive Density Control. During optimization, an adaptive density control mechanism periodically adjusts the number and configuration of Gaussians to improve reconstruction quality. The mechanism operates based on the magnitude of view-space positional gradients accumulated over a window of training iterations:

- **Splitting:** Large Gaussians covering regions with high positional gradients are split into two smaller Gaussians, redistributing the volume to better capture fine detail.
- **Cloning:** Small Gaussians in under-reconstructed regions (also indicated by high gradients) are duplicated and displaced along the gradient direction to increase coverage.
- **Pruning:** Gaussians whose opacity α_k falls below a threshold ϵ are removed, preventing the accumulation of near-transparent primitives that waste computation without contributing to the rendered image.

This process progressively refines the representation, concentrating Gaussians where geometric and appearance detail is needed while removing redundant primitives.

Real-Time Rendering. 3DGS achieves real-time rendering performance through a custom tile-based rasterization pipeline implemented in CUDA. The image is divided into 16×16 -pixel tiles, and each Gaussian is assigned to the tiles it overlaps based on its projected 2D extent. Within each tile, overlapping Gaussians are sorted by depth and alpha-composited in a single forward pass on the GPU. This tile-based

parallelization avoids the sequential per-ray marching required by volumetric methods such as NeRF, enabling rendering at over 30 FPS at 1080p resolution while achieving visual quality comparable to or exceeding the best implicit neural rendering methods. Training a 3DGS scene from multi-view images typically requires minutes rather than hours, a significant practical advantage over NeRF-based alternatives.

Historical Context. Triangle rasterization has been the dominant rendering paradigm in computer graphics for decades, driven by dedicated GPU hardware that processes millions of triangles per second through a fixed vertex-to-fragment pipeline. Triangles offer well-defined connectivity, normals, and texture coordinates, making them the natural primitive for real-time rendering engines and the foundation of the entire graphics hardware ecosystem.

Splatting emerged as an alternative for rendering point-sampled data—such as laser scans or volumetric datasets—where explicit triangle connectivity is unavailable. Westover [1992] introduced volume splatting, where each voxel is projected onto the screen as a Gaussian footprint and accumulated via compositing. Zwicker et al. [2001] extended this to surface rendering, proposing surface splatting with elliptical Gaussian kernels that adapt to the local surface orientation (Fig. 3.2). In a follow-up work, Zwicker et al. [2002] formulated EWA (Elliptical Weighted Average) splatting, which combines a reconstruction kernel with a low-pass filter to produce high-quality, alias-free renderings from point clouds—the same mathematical framework that underlies the projection in 3DGS (Eq. (3.10)). Despite producing high-quality renderings, splatting remained a niche technique: without dedicated hardware support and without the explicit connectivity that triangles provide, it could not compete with triangle rasterization for general-purpose real-time rendering.

In parallel, a separate line of research pursued differentiable 3D reconstruction from images. The key idea is to make the rendering process differentiable so that a 3D representation can be optimized by comparing rendered images against observed photographs. Neural Radiance Fields (NeRF) [Mildenhall et al., 2020] demonstrated that an MLP mapping 3D coordinates and viewing directions to color and density, combined with differentiable volume rendering, could synthesize photorealistic novel views from a set of posed images. NeRF achieved unprecedented visual quality and sparked an explosion of follow-up work, including extensions to unbounded scenes [Barron et al., 2022] and dramatically faster training via hash-based feature grids [Müller et al., 2022]. However, NeRF and its variants share a fundamental computational bottleneck: rendering requires dense per-ray sampling through the volumetric field, with each sample requiring an MLP forward pass. This makes both training and inference orders of magnitude slower than rasterization-based approaches, limiting practical deployment in real-time applications.

The convergence of these two threads—differentiable rendering from images and efficient point-based splatting—led to the development of 3D Gaussian Splatting. On the splatting side, the turning point came when the rendering pipeline was made differentiable, enabling point-based representations to be learned from data. Yifan et al. [2019] proposed Differentiable Surface Splatting (DSS), making the entire splatting pipeline differentiable and enabling inverse rendering—optimizing point positions,

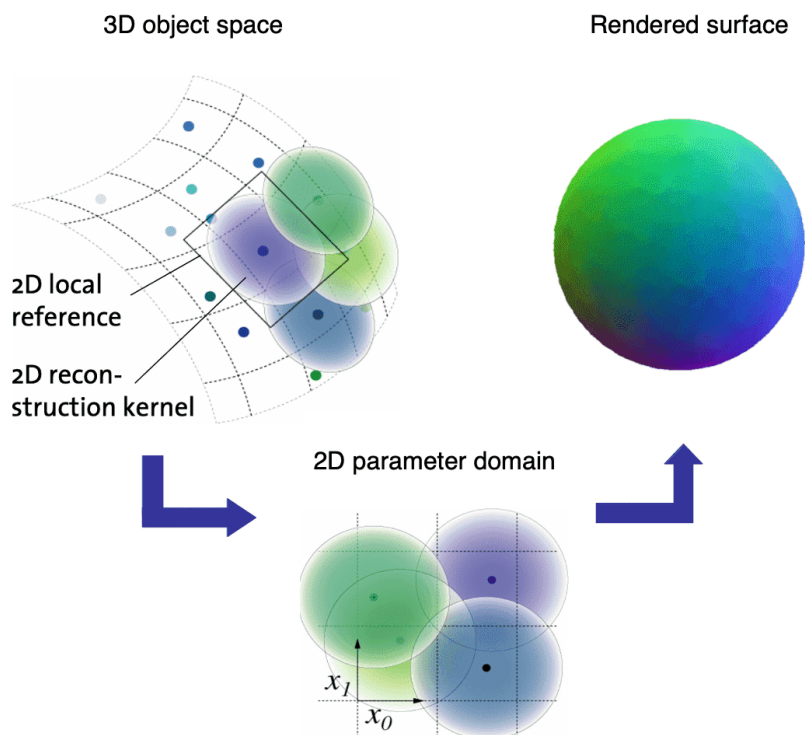


Figure 3.2: **Surface splatting with Gaussian reconstruction kernels.** Each surface point defines a 2D local reference frame on the object surface (top left). A Gaussian reconstruction kernel is centered at each point and blended in a 2D parameter domain (bottom), producing a smooth rendered surface (top right). This process—projecting anisotropic Gaussian primitives from 3D object space to screen space and blending them via alpha compositing—is the core rendering mechanism of 3D Gaussian Splatting. Image source: [Zwicker et al. \[2001\]](#).

normals, and colors from image observations via backpropagation. DSS demonstrated that splatting-based representations could be learned end-to-end from images. [Lassner and Zollhöfer \[2021\]](#) proposed Pulsar, an efficient differentiable renderer for sphere-based point primitives that achieved real-time rendering of millions of spheres on the GPU, demonstrating that differentiable point-based rendering could scale.

3D Gaussian Splatting [[Kerbl et al., 2023](#)] built on these foundations by combining anisotropic Gaussian primitives with adaptive density control and, crucially, a custom tile-based CUDA rasterization kernel that sorts and alpha-composites Gaussians in parallel across screen-space tiles. This fast differentiable CUDA kernel was the key engineering contribution that made 3DGS practical: it enabled both real-time rendering and efficient gradient computation for millions of Gaussians, achieving photorealistic novel-view synthesis at interactive rates from posed images—a capability that neither the classical splatting methods nor the earlier differentiable renderers achieved at comparable quality and speed.

3.4 MESHES VS. 3D GAUSSIAN SPLATTING

This thesis uses two fundamentally different 3D representations—triangle meshes and 3D Gaussian Splatting—each with distinct strengths that make them suitable for different aspects of the human–scene interaction pipeline. This section highlights the key differences between the two representations and how they are leveraged across the thesis.

Surface Representation. Triangle meshes represent surfaces explicitly as a set of vertices connected by edges and faces. This yields well-defined surface normals, watertight boundaries, and signed distance fields—geometric quantities that enable physics simulation, collision detection, and path planning. 3DGS, by contrast, represents scenes as unstructured collections of volumetric primitives with no explicit surface. Each Gaussian is a soft, semi-transparent ellipsoid; surfaces emerge only implicitly from the accumulation of many overlapping primitives. This lack of explicit geometry is a significant challenge for downstream tasks such as navigation and contact modeling, which we address in Chs. 8 and 9.

Rendering Quality. Meshes approximate smooth surfaces with flat triangles and rely on complex material models, texture maps, and global illumination algorithms to achieve photorealism—a process that is computationally expensive and requires significant artistic effort. 3DGS captures view-dependent appearance, fine geometric detail, and complex material properties naturally through its learnable per-Gaussian color and opacity parameters, producing photorealistic renderings at real-time rates without explicit material decomposition. This quality gap is a central motivation for Part II of this thesis.

Reconstruction. Mesh-based human reconstruction typically requires dense multi-view capture systems with calibrated cameras, or depth sensors, to obtain sufficient geometric constraints for surface fitting. Template-based methods such as SMPL registration further require careful initialization and regularization. 3DGS reconstruction, by contrast, requires only posed images—which can be as few as a sparse set of monocular video frames—without the need for depth sensors or controlled capture. This drastically lower input requirement is what enables the reconstruction of 3D human assets from in-the-wild YouTube video in Ch. 10, where body and camera parameters are estimated directly from the video, a setting where mesh-based methods cannot operate.

Animation and Deformation. Both representations can be animated using LBS (Sec. 3.1). For meshes, LBS transforms vertex positions, and the face connectivity automatically follows. For 3DGS, each Gaussian’s center is transformed like a mesh vertex, but the covariance matrix must additionally be rotated to preserve the Gaussian’s shape and orientation (Sec. 3.1). In both cases, per-primitive skinning weights determine joint influences. A key difference is that mesh blend weights are defined on a fixed topology and can be hand-crafted, while Gaussian skinning weights are

typically learned during reconstruction.

Usage Across the Thesis. Part **I** (Chs. 4–6) relies on meshes and point clouds, leveraging their explicit geometry for texture transfer via UV maps, sensor-based pose registration via scan alignment, and motion synthesis via walkable surfaces and signed distance fields. Part **II** (Chs. 7–10) adopts 3DGS as the sole 3D representation, demonstrating that all geometric structures needed for animation—navigable surfaces, contact constraints, shadow casting—can be extracted or approximated directly from raw Gaussian fields, while achieving photorealistic rendering quality unattainable with meshes.

Part I

Humans in Scenes with Classical Representations

The preceding chapters introduced the central question of this thesis—how to represent 3D humans and their environments to enable realistic interaction—and surveyed the landscape of related work and technical foundations. In this first part, we address this question using *classical representations*: meshes and point clouds that provide explicit geometry for physics, collision detection, and path planning.

We begin with human appearance, a prerequisite for any downstream application. Ch. 4 presents Pix2Surf, a method for transferring realistic clothing textures from 2D images to 3D garments worn atop SMPL. We then turn to capturing human behavior in 3D environments: Ch. 5 presents HPS, the first method to recover the full 3D pose of a human registered with large 3D scenes using only body-mounted sensors. Finally, Ch. 6 presents CoHMo, a method to synthesize long-horizon human motion across diverse 3D scenes by composing short-term motions in a canonical coordinate frame. Together, these three chapters address the foundational problems of appearance, capture, and motion synthesis using classical geometry. The second part of this thesis then asks whether the same problems can be solved—with superior visual fidelity—using neural representations alone.

PIX2SURF: LEARNING TO TRANSFER TEXTURE FROM CLOTHING IMAGE TO 3D HUMANS



Figure 4.1: **Real-time texture transfer to 3D garments.** Our model Pix2Surf digitally maps the texture of online retail store clothing images to the 3D surface of virtual garment items, enabling 3D virtual try-on in real time.

As the first contribution of Part I, this chapter addresses the appearance of 3D humans using classical mesh representations. We present a simple yet effective method to automatically transfer textures of clothing images (front and back) to 3D garments worn on top of SMPL [Loper et al., 2015], in real time. We first automatically compute training pairs of images with aligned 3D garments using a custom non-rigid 3D to 2D registration method, which is accurate but slow. Using these pairs, we learn a mapping from pixels to the 3D garment surface. Our idea is to learn dense correspondences from garment image silhouettes to a 2D-UV map of a 3D garment surface using shape information alone, completely ignoring texture, which allows us to generalize to the wide range of web images. Several experiments demonstrate that our model is more accurate than widely used baselines such as thin-plate-spline warping and image-to-image translation networks while being orders of magnitude faster. Our model opens the door for applications such as virtual try-on, and allows for generation of 3D humans with varied textures which is necessary for learning. Video results are available at <https://virtualhumans.mpi-inf.mpg.de/pix2surf/>.

4.1 INTRODUCTION

Our goal is to learn a model capable of transferring texture from two photographs (front and back) of a garment to the 3D geometry of a garment template, automatically and in real time. Such a model can be extremely useful for photorealistic rendering of humans, varied generation of synthetic data for learning, virtual try-on, art and design. Despite the many applications, automatic transfer of clothing texture to 3D garments has received very little attention. The vast majority of recent methods work in image space directly. Most works focus on either image based person re-posing [Zhao et al., 2018, Ma et al., 2017, 2018, Pumarola et al., 2018, Siarohin et al., 2018, Dong et al., 2018, Balakrishnan et al., 2018, Song et al., 2019], or virtual try-on [Raj et al., 2018, Zanfir et al., 2018b, Han et al., 2018, Wang et al., 2018, Yu et al., 2019, Dong et al., 2019a, Zheng et al., 2019]. Re-posing methods learn to synthesize image pixels to produce novel poses of a person, whereas virtual try-on methods learn to morph an image of a clothing item to drape it on a target person. The advantage of these methods is that they can be trained on large-scale datasets. The critical disadvantage is that they operate in pixel-space instead of 3D, they cannot synthesize difficult poses, and struggle to produce temporally consistent results. Another line of work extracts texture by fitting 3D human models (SMPL [Loper et al., 2015]) to images [Bhatnagar et al., 2019, Alldieck et al., 2018b, 2019a], but texture quality quickly deteriorates for complex poses. Other works map texture to 3D meshes [Varol et al., 2017, Gabeur et al., 2019, Lazova et al., 2019, Pons-Moll et al., 2017] by 3D scanning people, but the number and variety of clothing textures is limited because 3D scanning is time consuming.

To break the lack of 3D data barrier, our idea is to learn a dense mapping from images of clothing items, which are ubiquitous on the internet, to the 3D surface of a parametric garment template directly [Bhatnagar et al., 2019]. However, accurate texture transfer is far from trivial: web images vary in texture, garment size and style, pose, and background. Nonetheless, in comparison to clothing worn by humans, web-store images have less variation, which we exploit for learning our model. Instead of manually annotating garment landmarks [Ziwei et al., 2016, Güler et al., 2018], our idea is to collect training pairs by non-rigidly aligning a parametric 3D garment template to images. We leverage the parameterized garment models of MGN [Bhatnagar et al., 2019], non-rigidly fit their surface to image silhouettes. While our alignment usually produces good results, it is slow (5 – 15 minutes per image), and fails in 5% of the cases. Consequently, using only the successful fits, we learn a direct mapping from image pixels which runs in milliseconds, and is more robust than the optimization based approach. Our key hypothesis is that the *mapping is determined by the image silhouette shape alone*, and not by appearance. Hence, we train a CNN to predict correspondences from a UV map of the garment surface to pixel locations, given the silhouette shape alone as input. Since the model learns invariances to shape and pose as opposed to appearance, it generalizes to a wide variety of garment images of varied textures. We refer to the correspondence predictor as Pix2Surf. Pix2Surf allows us to digitally map the texture of a clothing image to the 3D surface of a garment in real time, see Fig. 4.1. Once the texture is mapped to 3D, we generalize to novel views, shapes and pose by applying 3D transformations to the 3D geometry [Bhatnagar et al.,



Figure 4.2: **Automatic texture generation for 3D garments.** Given online retail store images, our method produces textures for pre-defined garment templates that can be used to virtually dress SMPL [Loper et al., 2015] in 3D.

2019] as illustrated in Fig. 4.2. Pix2Surf enables, for the first time, virtual try-on from images in 3D directly and in real time, enabling applications such as VR/AR, gaming, and 3D content production.

To stimulate further research in this direction, we will make Pix2Surf publicly available for research purposes. This will allow researchers and users to add texture to the SMPL+Garments model [Bhatnagar et al., 2019], generate synthetic humans, and visualize garment images in 3D directly.

4.2 METHOD

Our key idea is to learn a *mapping* from images to the UV map of the garment, *without using texture information, but silhouette shape alone*. We first explain the parametric 3D garment models (Sec. 4.2.1) we use to regularize an automatic mesh to image silhouette fitting procedure (Sec. 4.2.2). Since fitting is expensive and error-prone, we learn an efficient neural mapping (Pix2Surf), which transfers the image texture onto the mesh in real time (Sec. 4.2.3).

4.2.1 Preliminary: Parametric Garment Model

We leverage publicly available 3D garments templates [Bhatnagar et al., 2019] parameterized as displacements from the SMPL body model [Loper et al., 2015]. For every garment category (T-shirt, short pants, long pants), we define a garment template $T^G \in \mathbb{R}^{m \times 3}$ as a sub-mesh of the SMPL body template $T \in \mathbb{R}^{n \times 3}$. An indicator matrix $\mathbf{I} \in \mathbb{Z}^{m \times n}$ evaluates to $\mathbf{I}_{i,j} = 1$ if a garment vertex $i \in \{1 \dots m\}$ is associated with a

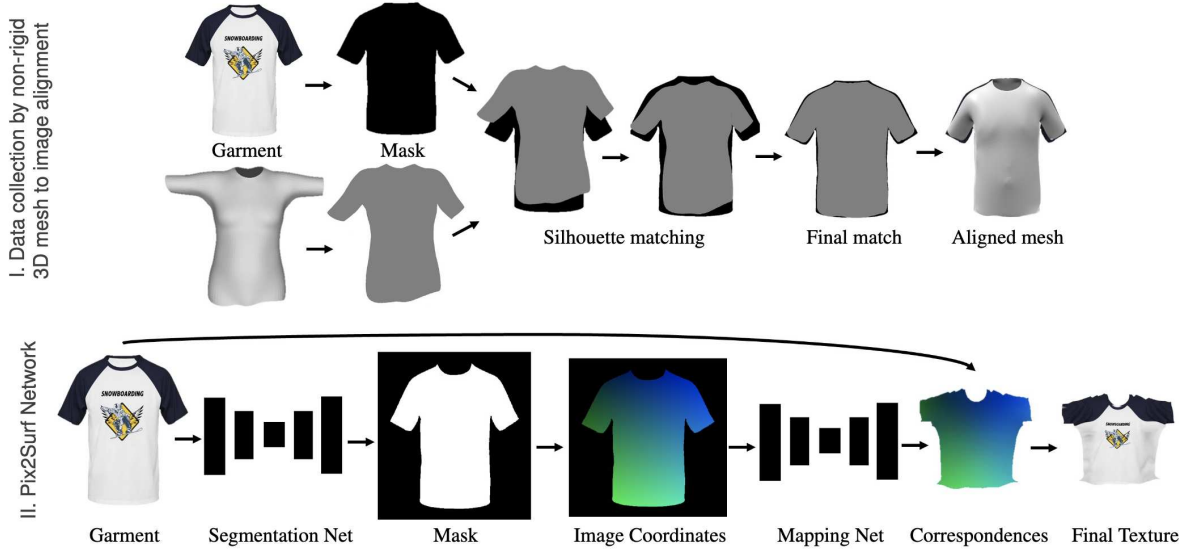


Figure 4.3: **Overview of method.** We build a large paired dataset of 3D garments and online retail store images by non-rigidly aligning 3D garment templates with automatically computed image segmentations (I). This allows us to train our model Pix2Surf (II) to predict correspondences from the UV map of the garment surfaces to image pixel locations.

body shape vertex $j \in \{1 \dots n\}$. This correspondence allows representing garments as displacements $\mathbf{D} \in \mathbb{R}^{m \times 3}$ from the unposed SMPL body. Given shape β and pose θ , we can articulate a garment using SMPL:

$$T^G(\beta, \theta, \mathbf{D}) = \mathbf{I} T(\beta, \theta) + \mathbf{D} \quad (4.1)$$

$$G(\beta, \theta, \mathbf{D}) = W(T^G(\beta, \theta, \mathbf{D}), J(\beta), \theta, \mathbf{W}), \quad (4.2)$$

with joints $J(\beta)$ and linear blend-skinning $W(\cdot)$ with weights $\mathbf{W}$. Since in this work we keep $\mathbf{D}$ fixed, we denote the garment model as $G(\beta, \theta)$. After the texture has been transferred to the surface, the geometry can still be changed with $\mathbf{D}$.

4.2.2 Non-Rigid Garment Fitting to Retail Images

To find a correspondence map between retail images in the web and 3D garments, we could non-rigidly deform its 3D surface to fit the image foreground. This is, however, not robust enough as different retailers photograph garments in different poses, backgrounds, and clothing itself varies in shape. Hence, we first automatically segment images, and then leverage the parametric garment model $G(\beta, \theta)$ defined in Sec. 4.2.1 to regularize the fitting process.

4.2.2.1 Automatic Segmentation

We use an automated version of GrabCut [Rother et al., 2004]. Since garments are typically photographed over simple backgrounds, we obtain an approximate foreground mask using thresholding. We then run a closing operation to fill the holes on this mask, and erode it to obtain a prior for "absolute foreground". The difference region between the mask and its eroded version is marked as "probable foreground". Analogously, we obtain "absolute background", and "probable background" using dilation. Using these prior maps to initialize GrabCut, we obtain accurate silhouettes without manual annotation.

4.2.2.2 Garment Fitting

We fit the garment surface to silhouettes in two stages. In the first stage, we minimize the following objective

$$E_1(\mathbf{f}, \mathbf{t}) = w_s E_s + w_\beta E_\beta + w_\theta E_\theta, \quad (4.3)$$

w.r.t. garment pose, shape and camera translation $\mathbf{t} \in \mathbb{R}^3$. The objective in Eq. (4.3) consists of a silhouette E_s , a shape regularization term E_β and a pose prior term E_θ , which we explain in the following. The different terms are balanced using weights w_* .

Silhouette term: It is defined as:

$$E_s(\beta, \theta, \mathbf{t}) = \Phi(w_i \Psi(I_r(G(\beta, \theta), \mathbf{t})) + w_o \hat{\Psi}(1 - I_r(G(\beta, \theta), \mathbf{t}))). \quad (4.4)$$

Here, Ψ and $\hat{\Psi}$ are the distance transform, and the inverse distance transform, respectively, of the silhouette image, Φ is a Gaussian pyramid function, and $I_r(G(\beta, \theta), \mathbf{t})$ is the binary garment silhouette image obtained with a differentiable renderer. The weights w_i and w_o balance the inner and outer distance transform contributions, respectively. Consequently, the objective in Eq. (4.4) maximizes overlap between the garment image and the rendered mesh, and penalizes model leakage into the background.

Shape regularization: In order to regularize the fitting process in Eq. (4.3), we use a Mahalanobis prior

$$E_\beta(\beta) = \beta^\top \Sigma_\beta^{-1} \beta \quad (4.5)$$

on the shape parameters, where Σ_β^{-1} is the diagonal covariance matrix from the SMPL dataset.

Pose prior: The term penalizes deviations of the pose from an A-pose θ_A

$$E_\theta(\theta) = \|\theta - \theta_A\|^2 \quad (4.6)$$

To minimize Eq. (4.3), we initialize the pose θ with an A-pose, as this approximates the pose of most garment images on the web. Additionally, we use scheduling: for shirts we first optimize shape and translation holding pose fixed and optimize all variables jointly afterwards. For pants and shorts the scheduling order is reversed. Stage 1 provides us with a coarse match to the silhouette of the garment, but the final

mesh is restricted by the parametric model $G(\beta, \theta, \mathbf{t})$. To perfectly match silhouette boundaries, we non-rigidly deform free form vertices $\mathbf{G} \in \mathbb{R}^{m \times 3}$ of a mesh initialized with the optimized garment result of the first stage $G(\beta, \theta)$. Specifically, we optimize a more sophisticated version of Eq. (4.3):

$$E_2(\mathbf{G}, \theta, \beta) = w'_s E'_s + w'_c E'_c + w'_l E'_l + w'_e E'_e + w'_b E'_b. \quad (4.7)$$

E'_s is the same as in Eq. (4.4), but now we optimize the free form vertices $\mathbf{G}$ instead of the latent parameters of the model $G(\beta, \theta)$. and E'_c , E'_l , E'_e and E'_b are coupling, Laplacian, edge constraint, and boundary smoothing terms, which we explain next.

Coupling term: It penalizes deviations of the free form vertices $\mathbf{G}$ from the parametric garment model $G(\beta, \theta)$:

$$E_c(\mathbf{G}, \beta, \theta) = \|\mathbf{G} - G(\beta, \theta)\|^2 \quad (4.8)$$

Edge Constraint Term: The belt or waistline in shorts and pants retail images forms an almost perfect *horizontal line*. We exploit this by matching the top ring (waistline) of pants and shorts 3D meshes to this horizontal line in image space. Let $\mathbf{G}_i \in \mathcal{R}$ denote the set of top ring vertices, $\pi(\mathbf{G}_i)_y$ denote the y coordinate of vertex $\mathbf{G}_i$ after projection $\pi(\cdot)$ onto the image, and let y_{max} denote the y coordinate of the horizontal line in the image. We incorporate the following penalty:

$$E_e(\mathbf{G}) = \sum_{\mathbf{G}_i \in \mathcal{R}} \|\pi(\mathbf{G}_i)_y - y_{max}\|^2 \quad (4.9)$$

Laplacian Term: In order to enforce garment smoothness and minimize distortion, we include a Laplacian term [Sorkine et al., 2004]. Given a mesh with adjacency matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$, the graph Laplacian $\mathbf{L} \in \mathbb{R}^{m \times m}$ is obtained as

$$\mathbf{L} = \mathbf{I}_m - \mathbf{K}^{-1} \mathbf{A}, \quad (4.10)$$

where $\mathbf{I}_m$ is the $m \times m$ identity matrix and $\mathbf{K}$ is a diagonal matrix such that $\mathbf{K}_{ii}$ stores the number of neighbors of vertex i . We minimize the mesh Laplacian:

$$E_l(\mathbf{G}) = \|\mathbf{L}\mathbf{G}\|_F^2 \quad (4.11)$$

Boundary Smoothing Term: To ensure that the boundaries remain smooth, we penalize high second order derivatives along the boundary rings, similar to [Pons-Moll et al., 2017].

The output of the fitting are 3D garment vertices $\mathbf{G}$, which together with their faces $\mathbf{F}$ define a deformed mesh $\mathcal{G} = \{\mathbf{G}, \mathbf{F}\}$ accurately aligned with image silhouette.

4.2.3 Learning Automatic Texture Transfer

The fitting procedure of Sec. 4.2.2 produces accurate garment-to-image correspondences, but it is slow—requiring several minutes per image—and occasionally fails when the silhouette is ambiguous or the garment deviates significantly from the parametric model. These limitations make it impractical as a real-time texture transfer

pipeline and prevent scaling to the large variety of garment images found on retail websites.

To overcome these limitations, we run the fitting method on a large collection of internet images, manually remove the unsuccessful fits, and use the resulting correspondences as training data for an efficient neural model, referred to as Pix2Surf. Pix2Surf learns to directly transfer texture from images to the 3D model surface based on the silhouette shape alone, bypassing the expensive optimization at test time. Once trained, the model runs in real time, enabling interactive applications such as virtual try-on. Next, we explain the key components of Pix2Surf, namely its input and output representation, and the losses used during training.

4.2.3.1 Pix2Surf: Input and Output Representation

The curated fits of Sec. 4.2.2 provide dense correspondences from image pixels (i, j) to the 3D garment surface $\mathcal{G} \subset \mathbb{R}^3$. Learning a mapping to the surface $\mathcal{G}$ embedded in $\mathbb{R}^3$ is hard, and does not allow leveraging fully 2D convolutional neural networks. Hence, we compute a 2D UV-map parameterization (Sec. 4.3) of the garment surface,

$$u : \mathcal{B} \subset \mathbb{R}^2 \mapsto \mathcal{G} \subset \mathbb{R}^3, \quad (4.12)$$

where $u(\cdot)$ maps 2D points (k, l) from the UV space to the surface $\mathcal{G}$ embedded in $\mathbb{R}^3$. In this way, all our targets live in a UV-space. Specifically, from the fits we generate RGB texture maps $\mathbf{Y} \in \mathbb{R}^{K \times L \times 3}$ using projective texturing, where K and L are the height and width of the UV map, and *UV correspondence maps*

$$\mathbf{C} \in \mathcal{C} \subset \mathbb{R}^{K \times L \times 2}, \quad (4.13)$$

which store, at every pixel (k, l) of the (front/back) UV map, the (i, j) coordinates of the corresponding image pixel:

$$\mathbf{C}_{k,l} = (i, j). \quad (4.14)$$

The input representation for the garment images is a *coordinate mask*

$$\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^{M \times N \times 2}, \quad (4.15)$$

where M and N are the image height and width, storing at every image pixel location its own coordinates if the pixel belongs to the foreground $\mathcal{F}$, and zero otherwise:

$$\mathbf{X}_{ij} = \begin{cases} (i, j) & \forall (i, j) \in \mathcal{F} \\ (0, 0) & \forall (i, j) \notin \mathcal{F}. \end{cases} \quad (4.16)$$

The foreground mask $\mathcal{F}$ is predicted at test time using a *segmentation network* trained using a standard cross-entropy loss—we compute segmentation labels for training using the automatic, but slow, GrabCut-based method of Sec. 4.2.2.1. For the front view of T-shirts, we additionally ask annotators to segment out the back portion of the shirt, which is visible in the front view image. The segmentation network learns to remove this portion, ensuring that the back portion is not mapped to the UV map. With this, we collect a dataset $\mathcal{D}$ of N training samples, consisting of inputs $\mathbf{X}$ and targets $\{\mathbf{Y}, \mathbf{C}\}$:

$$\mathcal{D} = \{\mathbf{X}^i, \{\mathbf{Y}^i, \mathbf{C}^i\}\}_{i=1}^N. \quad (4.17)$$

4.2.3.2 Pix2Surf: Learning

We could naively attempt to predict the texture maps $\mathbf{Y}$ directly from images $\mathbf{I}$ using image-to-image translation, but this is prone to overfit to image textures as we demonstrate in our experiments (Sec. 4.5.2). Instead, we follow a more geometric approach and learn a mapping

$$f(\mathbf{X}; \mathbf{w}) : \mathcal{X} \mapsto \mathcal{C} \quad (4.18)$$

from coordinate masks $\mathbf{X}$ to UV correspondence maps $\mathbf{C}$, forcing the network to reason about the input shape. This effectively learns to predict, for every UV map location (k, l) , the corresponding pixel coordinates (i, j) in the image:

$$\mathbf{C}_{k,l} = f_{k,l}(\mathbf{X}; \mathbf{w}) = (i, j). \quad (4.19)$$

Our key insight and assumption is that this smooth mapping depends only on silhouette shape $\mathbf{X}$, and not on the image texture $\mathbf{I}$. Because the coordinate mask encodes only the spatial layout of the foreground region, the network must learn geometric reasoning about how image pixels map onto the garment surface, rather than relying on appearance cues. This makes the learned mapping robust to the wide variety of colors, patterns, and textures encountered in retail clothing images.

During training, we minimize the following loss

$$\mathcal{L}_{\text{total}} = \lambda_{\text{reg}} \mathcal{L}_{\text{reg}} + \lambda_{\text{perc}} \mathcal{L}_{\text{perc}} + \lambda_{\text{recon}} \mathcal{L}_{\text{recon}} \quad (4.20)$$

over the training set $\mathcal{D}$, where each term in the loss is explained next.

Coordinate Regression Loss: $\mathcal{L}_{\text{reg}}$ evaluates an $L_2(\cdot)$ norm of the difference between the network output $f(\mathbf{X}^i; \mathbf{w})$ and the pseudo ground truths $\mathbf{C}$ obtained using the silhouette fitting algorithm of Sec. 4.2.2:

$$\mathcal{L}_{\text{reg}} = \sum_{i=1}^N \|f(\mathbf{X}^i; \mathbf{w}) - \mathbf{C}^i\|_2^2 \quad (4.21)$$

Reconstruction Loss: To provide more supervision to the network we use a differentiable sampling kernel [Jaderberg et al., 2015] to infer a texture map directly from the correspondence map. We minimize a dense photometric loss between predicted texture maps and target texture maps $\mathbf{Y}$ obtained with projective texturing (Sec. 4.2.3.1):

$$\mathcal{L}_{\text{recon}} = \sum_{i=1}^N \sum_{k=1}^K \sum_{l=1}^L \|\mathbf{I} [f_{k,l}^1(\mathbf{X}^i; \mathbf{w}), f_{k,l}^2(\mathbf{X}^i; \mathbf{w})] - \mathbf{Y}_{k,l}^i\|_1 \quad (4.22)$$

where $\mathbf{I}$ denotes the input RGB image and $\mathbf{I}[\cdot, \cdot]$ denotes differentiable bilinear sampling at continuous pixel coordinates. The image is sampled at locations $(i, j) = (f_{k,l}^1(\mathbf{X}^i; \mathbf{w}), f_{k,l}^2(\mathbf{X}^i; \mathbf{w}))$ provided by the predicted correspondence map.

Perceptual Loss: $\mathcal{L}_{\text{perc}}$ is the perceptual loss as defined in [Zhang et al., 2018b] between $\mathbf{I}[f(\mathbf{X}; \mathbf{w})] \in \mathbb{R}^{K \times L \times 3}$ (the sampled texture map) and $\mathbf{Y}$.

Once the network predicts a correspondence map for an input image, we use it and the parallelizable kernel to generate the final image by sampling points from the input image.

4.3 IMPLEMENTATION DETAILS

In this section, we present the implementation details of our method.

Neural Network Implementation: We use Adam optimizer for training both networks with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We use a constant learning rate of 0.0001. All mapping networks are trained for 200 epochs and all segmentation networks are trained for 20 epochs. During the training of the segmentation networks we use heavy color jittering to prevent the network from overfitting to the textures in the training data. The weights are set as $\lambda_{reg} = \lambda_{recon} = \lambda_{perc} = 1$. For the segmentation network we use a UNet with instance normalization and use color jittering in the input data to improve performance. For Pix2Surf we use a six block ResNet. The choice of normalization and activation functions is the same as [Isola et al., 2017].

Non-Rigid Garment Fitting: In all our experiments, we use the neutral gendered SMPL model. For each garment class we only use a subset of angles in the input axis-angle representation of pose as our optimization variables. For shirts, we use only the pose parameters corresponding to the shoulder joints - numbered 13, 14, 16, 17 in the canonical ordering. For shorts, joint corresponding to the hip joint - joints 1 and 2 are used. Joints 1, 2, 3, 4 which correspond to the hip and knee joints are used while registering pants.

The weights used in the optimization based registration are given below. For registration the front of T-shirts, we set $w_\beta = 3.0, w_s = 100, w_\theta = 1.0, w'_s = 2.0, w'_c = 0.3, w'_b = 2.0, w'_l = 30.0, w'_e = 0$. The same weights when registering the back are: $w_\beta = 5.0, w_s = 100, w_\theta = 0, w'_s = 2.0, w'_c = 0.3, w'_b = 2.0, w'_l = 30.0, w'_e = 0$.

For the front view of shorts, we set $w_\beta = 4.0, w_s = 100, w_\theta = 30.0$ in the first stage when shape is not being optimized and $w_\theta = 20$ when it is. $w'_s = 100, w'_c = 4.0, w'_b = 2.0, w'_l = 45.0, w'_e = 0.001$. For the back $w_\beta = 3.0, w_s = 160.0, w_\theta = 40.0$ when shape is not an optimization variable and $w_\theta = 30$ when it is. $w'_s = 100, w'_c = 3.8, w'_b = 6.0, w'_l = 75.0, w'_e = 0.01$.

For both front and back of pants, we set $w_\beta = 4.0, w_s = 100, w_\theta = 40.0$ in the stage when just the pose is optimized $w_\theta = 5$ when both shape and pose are optimization variables. $w'_s = 100, w'_c = 0.5, w'_b = 2.0, w'_l = 45.0, w'_e = 0.1$.

Baseline Implementation: For the SC-matching TPS based baseline, all parameters are set to their default values. The pix2pix baseline is implemented with the weight $\lambda_{gan} = 1$. We observed that increasing the weight of the adversarial loss worsens performance.

Custom UV Maps: Since we use cutouts of SMPL to define 3D garments, there are no available UV maps for use. To address this problem, we create custom UV maps for all 3D garments. All the UV maps used are created in blender. We obtained best performance when the shape of the UV maps is similar to the shapes of the input garments. The exact UV maps used can be seen in Fig. 4.4

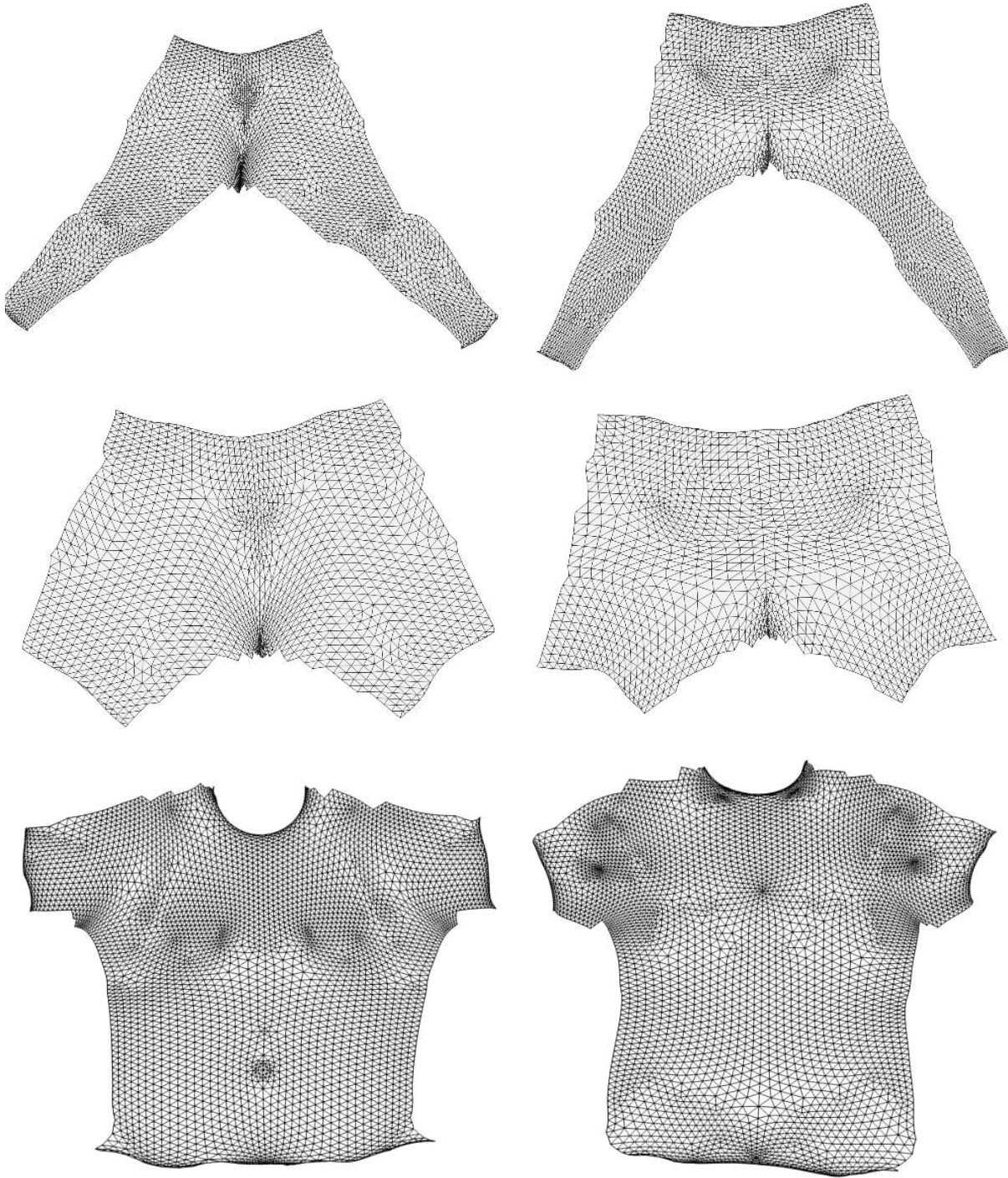


Figure 4.4: **Custom UV maps.** From top to bottom: pants, shorts, and shirt UV maps created in Blender. The UV map shapes are designed to resemble the corresponding garment silhouettes.

4.4 RESULTS

In this section, we present qualitative results of our texture transfer method. Figs. 4.5, 4.7, and 4.6 show textured garments obtained from online retail store images, rendered on SMPL bodies in various poses and shapes.



Figure 4.5: **Textured garments obtained using our method.** From the online retail store images (left), we create textures for three different garment categories (T-shirt, pants, shorts). We use the textured garments to virtually dress SMPL (right).



Figure 4.6: Textures mapped by Pix2Surf rendered atop SMPL.



Figure 4.7: Textures mapped by Pix2Surf rendered atop SMPL.



Figure 4.8: **Rendering textured garments on varied SMPL bodies.** Once textures are mapped to their corresponding texture maps, they can be rendered atop SMPL in different shapes and poses.

4.5 EXPERIMENTS

Since we are solving a problem for which there is no ground-truth data, we evaluate our proposed method qualitatively and with a user study. Once textures are mapped, they can be rendered on SMPL bodies in different shapes and poses (Fig. 4.8). We show results for three different garment types (T-shirts, shorts and long-pants), see Fig. 4.5. Notice that the texture is nicely mapped without transferring background. Notice also how the 3D textures are complete without *holes and seams*. We compare our method against the popular Shape-context with Thin Plate Spline (TPS) matching baseline, an image-to-image translation which operates on pixels directly, and image based virtual try on methods.

4.5.1 Dataset

To train our models, we create datasets of garment images by scraping the websites of clothing stores – specifically the websites of Zalando [Zalando], Tom Tailor [Tom-Tailor] and Jack and Jones [Jack-Jones]. The back view of clothing images is scarcely available on the internet, so we were unable to create a dataset of back view images large enough for training mapping networks. We leverage the fact that the distribution of garment silhouettes is similar for both front and back views, and either augment the dataset used for training the mapping from the back view image to the back UV map by combining front view images with back view images or using *only* front view images. We create a dataset consisting of 2267 front images of T-shirts. The networks pertaining to the back are trained using a dataset of 2267 images of which 964 are back view images and the rest are front images. The front shorts dataset has 2277 items. We use

the same dataset to train the networks pertaining to the back views. For pants we collect a dataset of 3410 front views. We create a dataset for back views by horizontally flipping the front view images and their corresponding silhouettes. The back dataset has 3211 items. The failure of the optimization based registration method explains the discrepancy between the number of items in the front and back views. Exploiting these front-back symmetries turns out to work well in practice, and allows to compensate for the unavailability of back view images.

4.5.2 Shape Context Matching and Image-to-Image Translation Baselines

We compare our method with shape context (SC) matching plus TPS and image-to-image translation baselines. We implement the shape context matching baseline as follows. We first render the garment mesh to obtain a silhouette. To make the baseline as strong as possible, we first pose the garment in the same pose as the image. We then warp the garment image to match the rendered garment. We then lift the texture from the warped image onto the mesh using projective texturing. Our experiments demonstrate that SC matching and TPS [Belongie et al., 2002] is not powerful enough to precisely match the contours of the two shapes so the final texture always has artifacts along the sides. See Fig. 4.9.

We also compare Pix2Surf to a Pix2Pix [Isola et al., 2017] baseline, which learns to produce a texture map from an image. To train Pix2Pix, we use the same texture maps we use to train Pix2Surf. Unlike Pix2Surf, Pix2Pix produces blurry results and does not preserve details of the input texture. See Fig. 4.9.

For further evaluation, we have conducted a *user study* with 30 participants. We created 20 videos containing images of reference clothing textures and two rotating textured 3D avatars “wearing” the reference clothing. One is textured using a baseline method and the other using Pix2Surf. We ask participants to choose the better looking avatar. In 100% of all comparisons, the avatar textured with Pix2Surf was preferred over the one textured using the baselines.

4.5.3 Towards Photorealistic Rendering

Provided with the texture map of a 3D avatar without clothes, Pix2Surf allows to map the texture atop the avatar from potentially an infinite number of clothing items scraped from online websites, *automatically* and in *real time*. See Fig. 4.10

Using these photorealistic renderings, we are able to compare our method with Image based virtual try-on method such as VITON [Han et al., 2018] and CP-VTON [Wang et al., 2018]. Since our method requires a 3D avatar as input, to be able to compare, we first render a photorealistic image using our method, and use VITON to change the upper garment of the rendering. Fig. 4.11 shows that VITON and CP-VTON work reasonably well for some poses, but fail badly for extreme poses while our method does not. We note that these methods are related but conceptually different compared to our method, so a direct fair comparison is not possible. These



Figure 4.9: **Comparison with SC matching and image-to-image translation baselines.** SC matching produces textures with artefacts and Pix2Pix produces blurry results. Left: Pix2Pix, Middle: SC matching, Right: Pix2Surf.



Figure 4.10: **Photorealistic virtual try-on.** Given the garment images and a 3D avatar with texture, we show our automatically textured garments on top.

methods are also not explicitly trained to handle such poses, but the figure illustrates a general limitation of image based virtual try-on approaches to generalize to novel view-points and poses. In stark contrast, we only need to generalize to silhouette shape variation, and once the texture is mapped to the 3D avatar, novel poses and viewpoints are trivially generated by applying transformations in 3D.

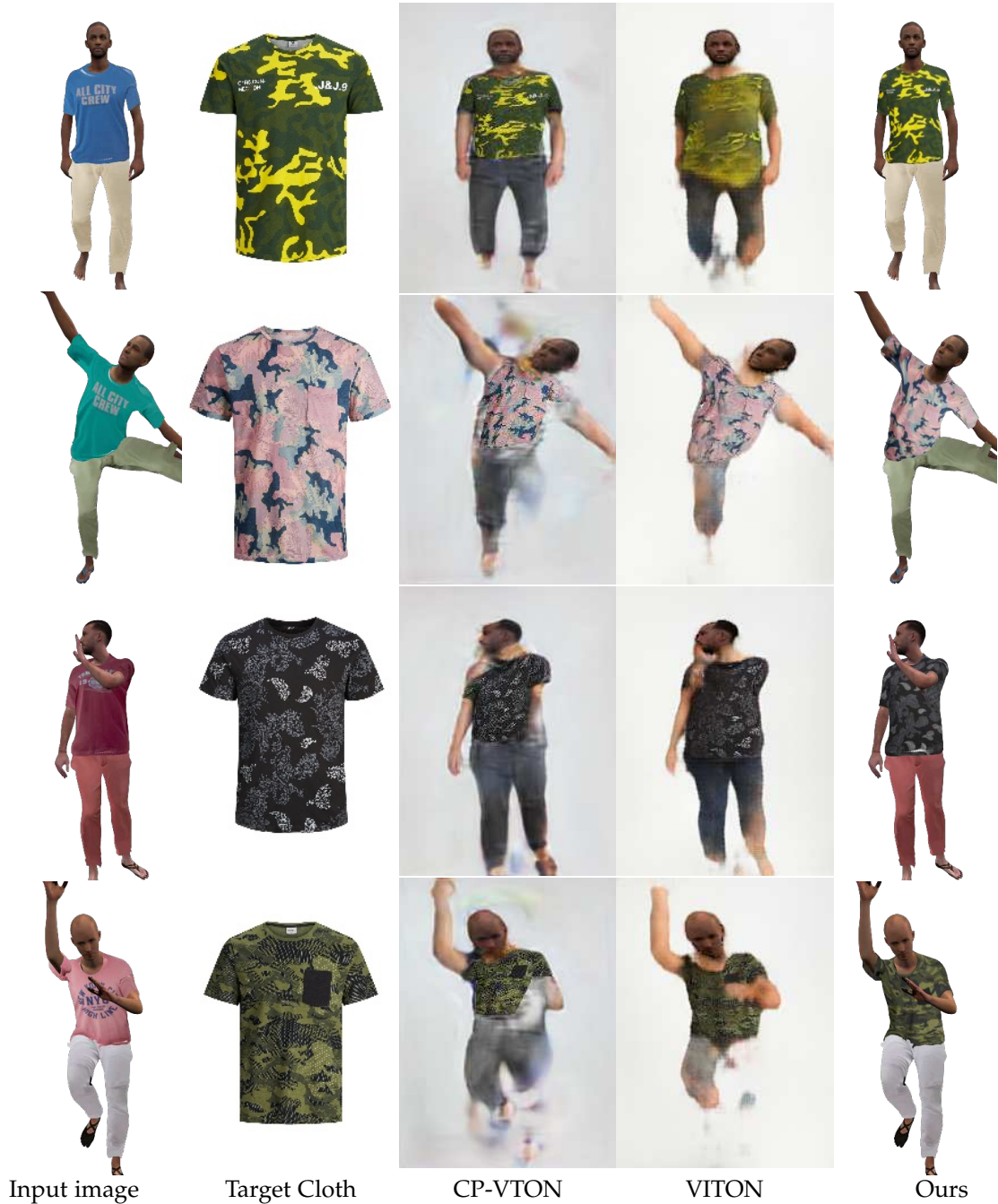


Figure 4.11: **Comparison with VITON and CP-VTON.** Image-based methods fail for extreme poses while our 3D approach handles them correctly.

4.6 CONCLUSION

We have presented a simple yet effective model that learns to transfer textures from web images of clothing to 3D garments worn by virtual humans. Our experiments show that our non-linear optimization method is accurate enough to compute a training set of clothing images aligned with 3D mesh projections, from which we learn a direct mapping with a neural model (Pix2Surf). While the optimization method takes up to ten minutes to converge, Pix2Surf runs in real time, which is crucial for many applications such as virtual try-on. Our key idea is to learn the correspondence map from image pixels to a 2D UV parameterization of the surface, *based on silhouette shape alone instead of texture*, which makes the model invariant to the highly varying textures of clothing and consequently generalize better. We show Pix2Surf performs significantly better than classical approaches such as 2D TPS warping (while being orders of magnitude faster), and direct image-to-image translation approaches.

We believe our model represents an important step towards learning a generative model of textures directly in 3D. We plan to address this in the future since it is lacking in current models like SMPL [Loper et al., 2015]. We focused on texture, and assume garment geometry is given, but we think it should be possible to infer geometry from images. Since clothing is laid out on a flat surface on web photographs, geometry inference will require modeling how 3D garments deform when they are flattened out on a surface.

HUMAN POSEITIONING SYSTEM (HPS): 3D HUMAN POSE ESTIMATION AND SELF-LOCALIZATION IN LARGE SCENES FROM BODY-MOUNTED SENSORS

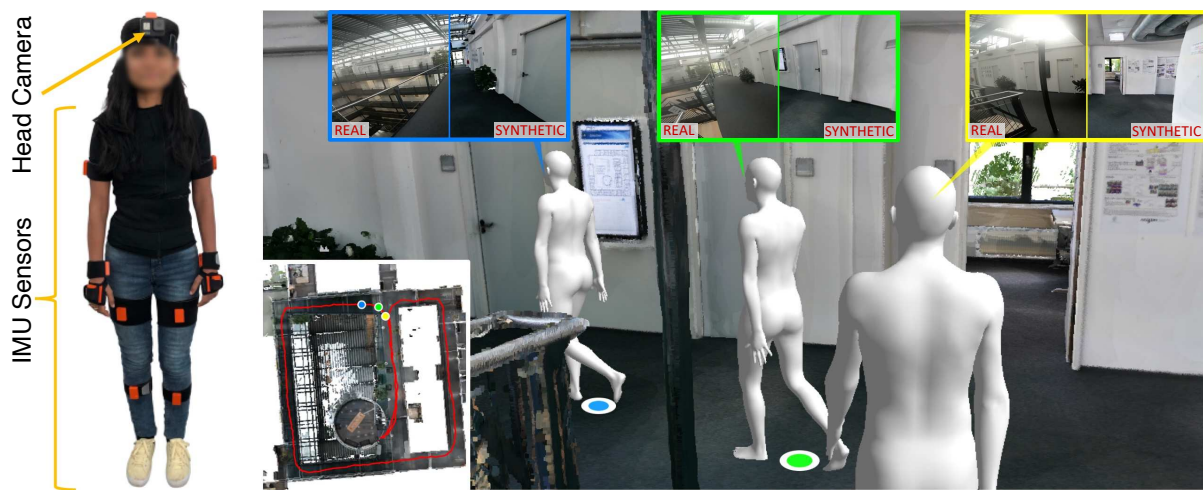


Figure 5.1: **HPS system overview.** HPS jointly estimates the full 3D pose and location of a subject within large 3D scenes, using only body-mounted sensors. Left: subject wearing IMUs and a head-mounted camera. Right: using the camera, HPS localizes the human in a pre-built map of the scene.

Having addressed human appearance in Ch. 4, we now turn to the problem of capturing human behavior in 3D scenes. In this chapter, we introduce Human POSEitioning System (HPS), a method to recover the full 3D pose of a human registered with a 3D scan of the surrounding environment using body-mounted sensors. Using IMUs attached at the body limbs and a head mounted camera looking outwards, HPS fuses camera based self-localization with IMU-based human body tracking. The former provides drift-free but noisy position and orientation estimates while the latter is accurate in the short-term but subject to drift over longer periods of time. We show that our framework exploits the benefits of the two, resulting in accuracy free of drift. Furthermore, we integrate 3D scene constraints into our optimization resulting in physically plausible motion. HPS complements more common third-person based 3D pose estimation methods. It allows capturing larger recording volumes and longer periods of motion, and could be used for VR/AR applications where humans naturally

interact with the scene without requiring direct line of sight with an external camera, or to train agents that navigate and interact with the environment based on first-person visual input, like real humans.

With HPS, we recorded a dataset of humans interacting with large 3D scenes (300–2500 m^2) consisting of 7 subjects and more than 3 hours of motion. The dataset and code have been made publicly available at <https://virtualhumans.mpi-inf.mpg.de/hps/>

5.1 INTRODUCTION

Capturing the full 3D pose of a human, while localizing and registering it with a 3D reconstruction of the environment, using *only body-mounted sensors*, would open the door to many applications and new research directions. For example, it would allow AR/VR users to move freely and interact with virtual objects in the scene, without the need for external cameras. From the captured data, we could train digital humans that plan and move like real humans. Moreover, by relying only on ego-centric data, we could capture a wider variety of human motion, outside of a restricted recording volume imposed by external cameras.

However, the dominant approach in vision has been to analyze humans from a *external third-person camera*, often without considering scene context [Kanazawa et al., 2018, Omran et al., 2018, Alldieck et al., 2019b]. A few recent methods capture 3D scenes and humans [Hassan et al., 2019], but again using a third-person camera. Capturing with external cameras is undoubtedly a central problem in vision, but it has its limitations – occlusions are a problem, and interactions across multiple rooms or beyond the camera area of sight cannot be captured; consequently recordings are typically short. There exist ego-centric capture approaches [Rhodin et al., 2016, Tome et al., 2020], but cameras are mounted on the body looking *towards* the person, and do not consider the environment.

We propose *Human POSEitioning System* (HPS), the first method to recover the full 3D pose of a human registered with a large 3D scan of the surrounding environment relying *only on body-mounted sensors* – body-mounted IMUs and a head mounted camera, approximating the visual field of view of the human. Inspired by visual-inertial odometry and localization [Lynen et al., 2015], as well as IMU based human pose estimation [Pons-Moll et al., 2010, von Marcard et al., 2018, von Marcard et al., 2017], HPS fuses information coming from body-mounted IMUs and from a head-mounted camera [Sattler et al., 2016, Sarlin et al., 2019, Taira et al., 2018] (see Fig. 5.1). Instead of placing the camera towards the body [Rhodin et al., 2016, Tome et al., 2020], we place it towards the scene, which allows us to capture what the human observes together with their 3D pose. In comparison to third-person pose methods, the body is not seen by the camera, which poses new challenges.

Pure IMU based tracking is known to drift over time and camera localization produces many outliers. By jointly integrating IMU tracking with camera self-localization, we are able to remove drift, and recover the human trajectory when self-localization fails. Furthermore, since we can approximately locate the person in the 3D scene, we further incorporate scene constraints when foot contact is detected. Overall, with HPS

we recover natural human motions, registered with the 3D scene and free of drift, during *long periods* of time, and over *large areas*.

To demonstrate the capabilities of HPS, we capture a dataset of real people moving in large scenes. Our HPS-dataset consists of 8 types of environments spanning 300–2500 m^2 , and 7 subjects performing a variety of activities such as walking, exercising, reading, eating, or simply working in the office. The dataset can be used as a testbed for ego-centric tracking with scene constraints, learn how humans interact and move within large scenes over long periods of time, or learn how humans react and process visual input arriving at their eyes.

Overall, we make the following contributions:

1. To the best of our knowledge, HPS is the first approach to estimate the full 3D pose of a human while localizing the person within a pre-scanned large 3D scene using body-mounted sensors.
2. We introduce a joint optimization which integrates camera localization, IMU-tracking and scene constraints, resulting in smooth and accurate human motion estimates.
3. We provide the *HPS dataset*, a new dataset consisting of 3D reconstructions of large scenes (spanning 300–2500 m^2), ego-centric video, IMU data, and our 3D reconstructed humans moving and interacting with the environment.

In contrast to existing 3D pose datasets, which are captured from a third-person view, ours is captured from an egocentric view. We believe both HPS and the HPS-dataset will provide a significant step towards developing future algorithms to understand and model 3D human motion and behavior within the 3D environment from an egocentric perspective.

5.2 METHOD

Our goal is to recover the 3D body pose and location of a subject in a known scene from egocentric measurements. To this end, our method requires as input: 1) a head-mounted camera, 2) body-mounted IMUs, and 3) a pre-built 3D scan of a scene, along with a database of RGB scene images with known camera parameters. Using camera data, our method localizes the person within a pre-scanned 3D scene (Sec. 5.2.2), estimates their 3D pose using IMUs (Sec. 5.2.3) and in a joint optimization step, (Sec. 5.2.5) integrates camera localization, IMU pose estimates and scene constraints, resulting in smooth and accurate human motion estimates. For an overview of our method, see Fig. 5.2.

We use the Skinned Multi-Person Linear (SMPL) body model [Loper et al., 2015] to represent the human subject. SMPL is a differentiable function

$$M(\beta, \theta, \mathbf{t}) : \mathbb{R}^{10} \times \mathbb{R}^{72} \times \mathbb{R}^3 \rightarrow \mathbb{R}^{6890 \times 3}$$

that maps shape β , pose θ , and translation $\mathbf{t}$ parameters to the vertices of a watertight human mesh. The underlying skeleton of SMPL has 24 joints. The pose parameters

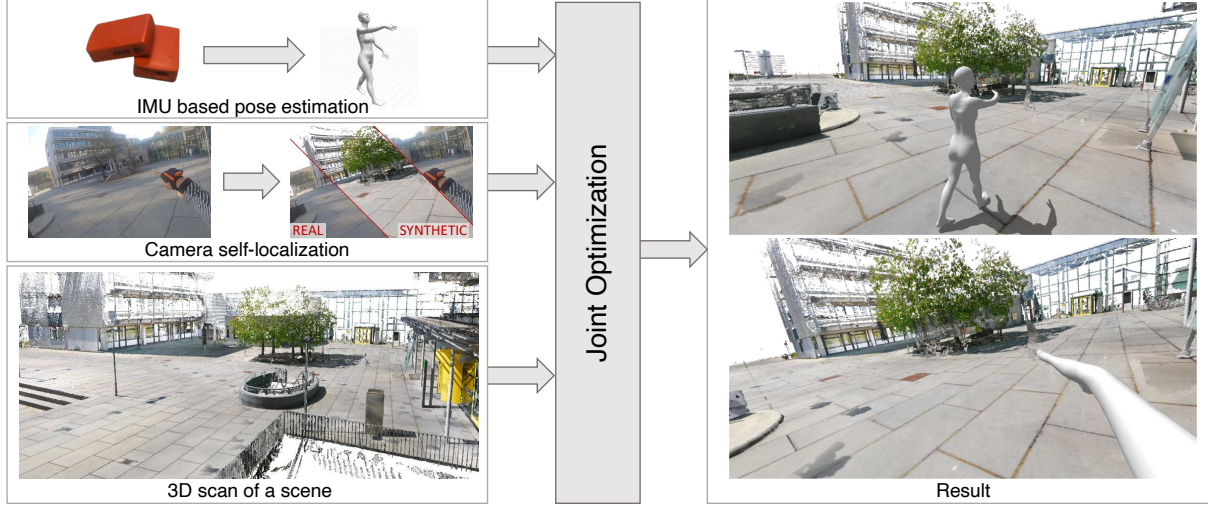


Figure 5.2: **Overview.** Our method uses IMU data, RGB video from a head mounted camera, and a pre-scanned scene as input. We obtain approximate 3D body pose using IMU data, and head camera self-localization to localize the subject within the 3D scene. We then integrate the approximate pose, the camera position and orientation, along with the scene landscape information in a joint optimization framework to obtain the final location and pose estimates.

$\theta \in \mathbb{R}^{72}$ correspond to the relative orientation of each joint in the SMPL skeleton, expressed in axis-angles. The shape parameters $\beta \in \mathbb{R}^{10}$ are the PCA coefficients of a shape space learnt from a corpus of registered scans. We use the notation $M_n(\beta, \theta, \mathbf{t}) \in \mathbb{R}^3$ to indicate the n^{th} vertex of SMPL. We obtain approximate shape parameters β of a person from body measurements. We assume that β remains static for a sequence and aim to recover θ and $\mathbf{t}$ of the subject registered with the 3D environment. Henceforth we drop β for notational convenience. We denote the global body orientation—the axis-angle of the root joint, i.e., the first three components of θ —by $\psi \in \mathbb{R}^3$.

5.2.1 3D Scene Reconstruction and RGB Image Database

To obtain a representation of the 3D scene, we use a standard commercial solution to obtain a very dense scene point cloud: NavVis M6 [NavVis] mobile capture system. It makes use of 4 LiDAR sensors and 6 RGB cameras. The scene is reconstructed from the LiDAR data and RGB images using a proprietary algorithm. The algorithm, in the process, also provides the extrinsic and intrinsic parameters for all the RGB images.

5.2.2 Camera Self-localization

The camera self-localization stage aims to estimate the position and orientation of the human head from a head-mounted camera. For the head-mounted camera, we use a GoPro camera. We ensure that it remains tightly attached at the forehead throughout

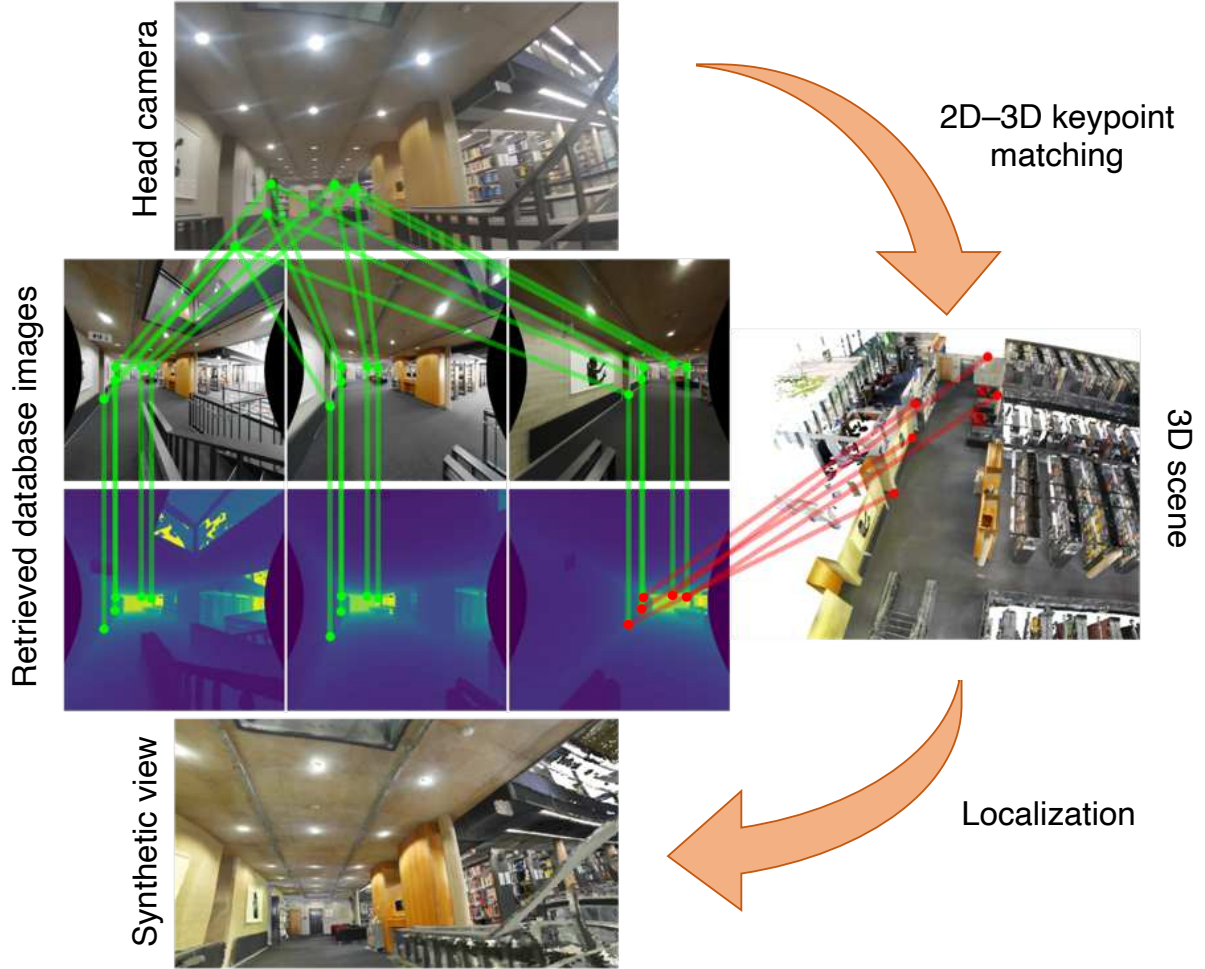


Figure 5.3: **Camera self-localization.** We match the head camera image keypoints with the keypoints from the prefiltered database with known 2D-3D scene correspondences. We then localize the camera on the scene by minimizing a reprojection error of the keypoints. *From top to bottom:* head camera image (query), top-3 matches from a dataset, depthmaps rendered from the same position to map database keypoints to 3D, synthetic view of the scene from the inferred camera position.

recordings. To scale to large scenes, we use a hierarchical structure-based localization algorithm [Sarlin et al., 2019, 2020] (Fig. 5.3). It first identifies a set of potentially relevant database images, *i.e.*, images used to build the 3D scene map, through image retrieval via NetVLAD [Arandjelovic et al., 2016] descriptors. 2D-3D matches are established between local SuperPoint [DeTone et al., 2018] features extracted in the query image and 3D points visible in the top-40 retrieved images. These matches are then used to estimate the camera pose by applying a P3P solver [Kneip et al., 2011, Haralick et al., 1994, Kukulova et al., 2010] inside a RANSAC loop [Fischler and Bolles, 1981] with local optimization [Lebeda et al., 2012]. Rather than building a separate sparse Structure-from-Motion point cloud for localization, as originally used in [Sarlin et al., 2019], we obtain 3D point positions from our dense scene 3D model [Taira et al., 2018]. For each pixel in a database image, we obtain the corresponding 3D



Figure 5.4: **Real versus rendered images.** Comparison of a real image from a database and a result of our rendering.

point by rendering the 3D model from the known pose of the image. 2D-2D matches between the query and the top-40 retrieved database images thus yield the required 2D-3D matches. From the camera self-localization step, we obtain estimates for camera orientation $\mathbf{R}^C$ and position $\mathbf{t}^C$.

To obtain 2D-3D correspondences, for each image in the dataset we produced a rendering (Fig. 5.4) of the scene pointcloud using known extrinsic and intrinsic camera parameters (Fig. 5.4). For rendering, we use the surface splatting technique [Zwicker et al., 2001] with fixed splat size. Together with the color rendering we produce a point mapping of the image. The resolution of the map is the same as the resolution of the color image. Each pixel of this map contains the index of the point in the pointcloud which is visible in that pixel. As NavVis scanners use fisheye cameras, photos obtained from them are heavily distorted. This affects keypoint detection performance. To alleviate this, we undistort both camera images and point mappings by generating a mapping between our fisheye camera image plane and a fixed perspective camera plane. Undistorted images are run through the SuperPoint [DeTone et al., 2018] keypoint detector. In the end, each keypoint is mapped to the position of the 3D point on the scan according to the generated point

For feature extraction and matching, we use SuperPoint [DeTone et al., 2018] and SuperGlue [Sarlin et al., 2020] PyTorch implementation pretrained on MegaDepth [Li and Snavely, 2018] and ScanNet [Dai et al., 2017] datasets. We detect 4096 keypoints at max for each image and perform 40 sinkhorn algorithm iterations at matching stage.

For database prefiltering we use NetVLAD PyTorch implementation pretrained on Pittsburgh 250k dataset [Torii et al., 2018]. We select 40 most similar database images based on the cosine distance between query and database NetVLAD descriptor. We use COLMAP [Schönberger and Frahm, 2016, Schönberger et al., 2016] software to minimize the reprojection error of the matched keypoints. Overall, the pipeline takes around 3s to localize a frame at 1920×1080 resolution using Nvidia Q8000 GPU.

To retrieve intrinsic parameters of the head-mounted camera, we take several photos of a checkerboard pattern and use OpenCV [Bradski, 2000] camera calibration tools to get the parameters. In our camera self-localization pipeline, we use an OpenCV camera model with 2 radial and 2 tangential distortion coefficients.

5.2.3 IMU-Based Pose Estimation

We use a commercial inertial mo-cap system provided by XSens [Paulich et al., 2018], which uses 17 IMUs attached to the body with velcro-straps or a suit. IMUs provide the orientation and acceleration in the local coordinate frame of the sensor which is then used to estimate the pose of the person.

XSens IMUs provide 3D pose estimates, denoted as θ^I and location estimates relative to the starting position of recording - denoted as $\mathbf{t}^I$, using a proprietary algorithm based on a Kalman filter and a kinematic model of the human body to reduce drift. While it provides accurate articulation, our experiments show that the global orientation and position drift significantly over time. (Fig. 5.5, 5.10). Using acceleration information, IMUs also detect feet contacts with the ground, which we integrate in our joint optimization algorithm. It uses a Kalman filter developed specifically for human motion and a kinematic model of the human body to correct for local orientation drift. This provides very accurate articulations but the global orientation and position remain inaccurate. It also provides very accurate foot contacts by integrating IMU acceleration with the kinematic model.

5.2.4 IMU-Camera synchronization

To synchronize IMU and camera data, we ask each subject to clap at the beginning of recordings. We detect these claps in both modalities to obtain synchronized starting frames. To check for time drift (Fig. 5.6) between the two data streams, we perform the following experiment: we made a special 1 hour long recording with a subject clapping every 5 minutes. We synchronize our system with the first clap and measure the difference in time we get for each consecutive clap after that.

5.2.5 Joint Optimization

Our joint optimization algorithm finds the pose parameters of the SMPL body model in order to satisfy scene and smoothness constraints.

Mathematically, fusion entails minimizing the following objective over a batch of T

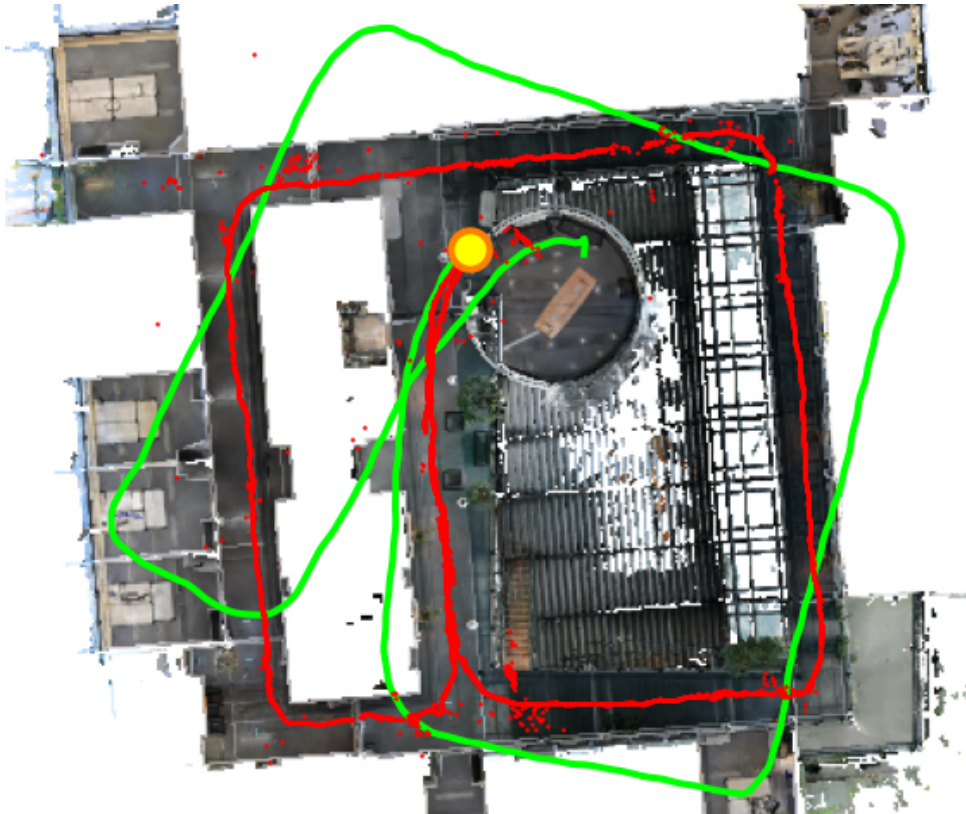


Figure 5.5: **Comparison of IMU and camera trajectories.** IMU trajectory (green) and camera self-localization (red). The yellow dot marks the start. Notice the red trajectory is free of drift but has outliers.

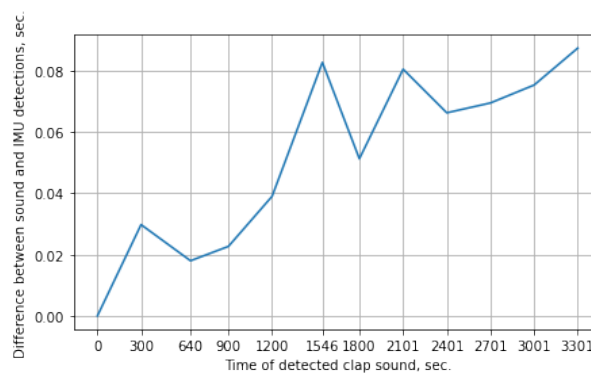


Figure 5.6: **Time synchronization between camera and IMU.** Measured time drift between camera and IMU clap detections after synchronization with the first clap.

frames

$$\mathcal{L}(\boldsymbol{\theta}_{1:T}, \mathbf{t}_{1:T}) = w_i \mathcal{L}_{\text{int}} + w_s \mathcal{L}_{\text{scene}} + w_{\text{sm}} \mathcal{L}_{\text{sm}} \quad (5.1)$$

with respect to pose $\boldsymbol{\theta}_{1:T}$ and translation $\mathbf{t}_{1:T}$ parameters. $\boldsymbol{\theta}_{1:T} \in \mathbb{R}^{72T}$ and $\mathbf{t}_{1:T} \in \mathbb{R}^{3T}$ are stacked model poses and translations for each time step $j = 1 \dots T$. In practice we fix the pose parameters corresponding to body joints and only optimize for the global body orientation, as we find that IMUs recover the articulation of the body very well. In the following, we explain the terms in more detail.

Scene Contact Term $\mathcal{L}_{\text{scene}}$: IMUs detect foot contacts directly. For sitting, we develop a pose-based heuristic that compares the root-to-heel distance with the standing reference and checks for low velocity; the heuristic details, thresholds, and manually defined contact vertices are described in Sec. 5.3.

When a foot or hip contact is detected, we force that body part to be in contact with the ground by using an energy term consisting of two subterms $\mathcal{L}_{\text{scene}} = w_c \mathcal{L}_{\text{contact}} + w_v \mathcal{L}_{\text{slide}}$. Let $\mathcal{B}_k$ with $k \in [1, 2, 3, 4, 5]$ denote 5 sets of manually defined vertex indices in the SMPL corresponding to the toe and heel regions for the left and right foot and the hip, and let $c_j^k \in [0, 1]$ be a binary variable indicating if part k is in contact with the ground at frame j . We define the following contact term, which snaps the foot or the hip vertices to the closest scene vertices.

$$\mathcal{L}_{\text{contact}} = \frac{1}{5T} \sum_{j=1}^T \sum_{k=1}^5 \sum_{n \in \mathcal{B}_k} \frac{1}{|\mathcal{B}_k|} c_j^k \|M_n(\boldsymbol{\theta}_j, \mathbf{t}_j) - v(n)\|_2, \quad (5.2)$$

where $M_n(\boldsymbol{\theta}_j, \mathbf{t}_j)$ is the n^{th} vertex of the SMPL mesh at frame j , and

$$v(n) = \underset{\mathbf{v}_s \in \mathcal{V}_s}{\operatorname{argmin}} \|M_n(\boldsymbol{\theta}_j, \mathbf{t}_j) - \mathbf{v}_s\|_2$$

returns the closest point $\mathbf{v}_s$ in the scene point cloud $\mathcal{V}_s$ to $M_n(\boldsymbol{\theta}_j, \mathbf{t}_j)$. To prevent the feet and hips from sliding when in contact with the scene, we constrain the distance between foot or hip parts in contact with the scene in two successive frames to be zero.

$$\mathcal{L}_{\text{slide}} = \frac{1}{5(T-1)} \sum_{j=1}^{T-1} \sum_{k=1}^5 \sum_{n \in \mathcal{B}_k} \frac{1}{|\mathcal{B}_k|} c_j^k c_{j+1}^k \|M_n(\boldsymbol{\theta}_j, \mathbf{t}_j) - M_n(\boldsymbol{\theta}_{j+1}, \mathbf{t}_{j+1})\|_2. \quad (5.3)$$

The contact vertex groups $\mathcal{B}_k$ are manually defined in Blender (see Sec. 5.3 and Fig. 5.7).

Smoothness Term $\mathcal{L}_{\text{sm}}$: We impose smoothness constraints on global translation, and global orientation

$$\mathcal{L}_{\text{sm}} = w_T \mathcal{L}_T + w_G \mathcal{L}_G, \quad (5.4)$$

where the translation term equals:

$$\mathcal{L}_T = \frac{1}{T-1} \sum_{j=1}^{T-1} \max(\|(\mathbf{t}_j - \mathbf{t}_{j+1})\|_2 - \|(\mathbf{t}_j^I - \mathbf{t}_{j+1}^I)\|_2, 0). \quad (5.5)$$

We define $\mathbf{R}^G(\boldsymbol{\psi}) = \exp(\hat{\boldsymbol{\psi}}) \in \text{SO}(3)$, where the $\hat{\cdot}$ operator maps an axis-angle to its corresponding skew-symmetric matrix and $\exp(\cdot)$ yields the rotation matrix via the

Rodrigues formula. The global orientation smoothness term is defined as

$$\mathcal{L}_G = \frac{1}{T-1} \sum_{j=1}^{T-1} \|(\log((\mathbf{R}^G(\psi_j))^\top \mathbf{R}^G(\psi_{j+1})))^\vee\|_2 \quad (5.6)$$

where the log operation recovers the skew-symmetric matrix from the relative rotation matrix, and the $^\vee$ operator converts it to its axis-angle representation.

Interpenetration Term $\mathcal{L}_{\text{int}}$: To penalize interpenetration between SMPL and the scene mesh, we force points inside SMPL to be in the space unoccupied by the scene. We first define an occupancy field of the scene by voxelizing the scene pointcloud. By trilinearly sampling the occupancy field, we define a function $S : \mathbb{R}^3 \mapsto [0, 1]$ that maps every point in space to an occupancy value. To obtain points inside the mesh, we project face centroids inside the mesh using the face normals. The face normal is

$$\mathbf{n}(f, \boldsymbol{\theta}, \mathbf{t}) = \frac{(M_{f_1} - M_{f_2}) \times (M_{f_3} - M_{f_2})}{\|(M_{f_1} - M_{f_2}) \times (M_{f_3} - M_{f_2})\|_2}, \quad (5.7)$$

where we write M_{f_k} as shorthand for $M(\boldsymbol{\theta}, \mathbf{t})_{f_k}$. The projected centroid is

$$P(f, \boldsymbol{\theta}, \mathbf{t}) = \frac{\sum_{k=1}^3 M(\boldsymbol{\theta}, \mathbf{t})_{f_k}}{3} - \pi \mathbf{n}(f, \boldsymbol{\theta}, \mathbf{t}), \quad (5.8)$$

where f is a face in SMPL defined by three vertex indices f_1, f_2, f_3 and π is a hyperparameter that controls the depth of projection. The interpenetration loss is defined as

$$\mathcal{L}_{\text{int}} = \frac{1}{T|\mathcal{F}|} \sum_{j=1}^T \sum_{f \in \mathcal{F}} S(P(f, \boldsymbol{\theta}_j, \mathbf{t}_j)) \quad (5.9)$$

where $\mathcal{F}$ is the set of SMPL faces.

5.2.6 Initialization

Since the objective function in Eq. (5.1) is highly non-convex, convergence to a good minimum hinges on good initialization. We initialize translation parameters $\mathbf{t}_j$ using filtered camera localization estimates $\mathbf{t}_j^F$. Camera localization often fails and results are typically noisy. IMU location estimates, though accurate in the short term, suffer from drift. We use IMU location estimates to identify erroneous camera localizations. Using correct camera localizations, we orient IMU trajectories to match the person's overall trajectory and interpolate between the filtered frames using these corrected IMU trajectories. See Alg. 1

For poses $\boldsymbol{\theta}$, the simplest choice is to initialize with IMU poses $\boldsymbol{\theta}_j = \boldsymbol{\theta}_j^I$. However, the root orientation ψ^I of this estimate is typically perpendicular to its trajectory, which deviates from the more accurate self-localization trajectory (see Fig. 5.5, 5.10).

Hence, our idea is to rotate the IMU root orientation to align it to the filtered self-localization trajectory. To this end, we first estimate the tangent direction of the

Algorithm 1: Filtering algorithm.

Input: η, ϵ , IMU positions: $\mathbf{t}_{1:N}^I$, Camera localizations: $\mathbf{t}_{1:N}^C$, a function contiguous that returns an array of values - (start,end) of contiguous blocks in an array of indices

Output: Filtered positions: $\mathbf{t}_{1:N}^F$

Init: $\text{vel_max} = \infty, \mathbf{t}_{1:N}^F = \mathbf{t}_{1:N}^C, \text{invalids} = []$

while $\text{vel_max} > \text{ffl}$ **do**

for j **in** $1 \dots N$ **do**

$v_j^F = \|\mathbf{t}_{j+1}^F - \mathbf{t}_j^F\|_2, v_j^I = \|\mathbf{t}_{j+1}^I - \mathbf{t}_j^I\|_2$

if $v_j^F > \eta v_j^I$ **then**

$\text{invalids} = \text{invalids} + [j]$

for (start,end) **in** contiguous(invalids) **do**

$\mathbf{v}_{ta} = \frac{\mathbf{t}_{end+1}^F - \mathbf{t}_{start-1}^F}{\|\mathbf{t}_{end+1}^F - \mathbf{t}_{start-1}^F\|_2}, \mathbf{v}_{in} = \frac{\mathbf{t}_{end+1}^I - \mathbf{t}_{start-1}^I}{\|\mathbf{t}_{end+1}^I - \mathbf{t}_{start-1}^I\|_2}$

for k **in** start $\dots$ end **do**

$\mathbf{t}_k^F = \mathbf{t}_{k-1}^F + \exp(\widehat{\mathbf{v}_{in} \times \mathbf{v}_{ta}})(\mathbf{t}_k^I - \mathbf{t}_{k-1}^I)$

$v_{1:N-1}^F = \|\mathbf{t}_{2:N}^F - \mathbf{t}_{1:N-1}^F\|_2$

$\text{vel_max} = \max(v_{1:N-1}^F)$

self-localization and IMU trajectories

$$\mathbf{v}_j^F = \frac{\mathbf{t}_{j+\gamma}^F - \mathbf{t}_j^F}{\|\mathbf{t}_{j+\gamma}^F - \mathbf{t}_j^F\|_2}, \quad \mathbf{v}_j^I = \frac{\mathbf{t}_{j+\gamma}^I - \mathbf{t}_j^I}{\|\mathbf{t}_{j+\gamma}^I - \mathbf{t}_j^I\|_2},$$

($\gamma = 3$ in our case) and correct the IMU root orientation $\exp(\widehat{\psi_j^I})$ with the following formula

$$\psi_j^{I,*} = (\log(\exp(\widehat{\mathbf{v}_j^I \times \mathbf{v}_j^F}) \exp(\widehat{\psi_j^I})))^\vee, \quad (5.10)$$

where $\exp(\widehat{\mathbf{v}_j^I \times \mathbf{v}_j^F})$ is the rotation that aligns $\mathbf{v}_j^I$ with $\mathbf{v}_j^F$.

We initialize translation parameters $\mathbf{t}_j$ using filtered camera localization estimates $\mathbf{t}_j^F$. The camera is located on the head of the person, but $\mathbf{t}_j$ corresponds to root joint of SMPL - which is approximately located at the center of the human stomach. To account for this difference in height, we only use the x,y coordinates of camera self-localization. In all frames, we replace the z value of $\mathbf{t}_j^F$ by the z value of $\mathbf{t}_j^I$. We observe that there is not a huge amount of accumulation in the z dimension of the positions recovered from IMUs.

5.2.7 Coordinate Frame Alignment

While the camera estimates $\mathbf{R}^C$ and $\mathbf{t}^C$ are in the 3D scene coordinates, IMU estimates θ^I and $\mathbf{t}^I$ are not. Before the Initialization step (Sec. 5.2.6) of our joint optimization

algorithm, (Sec. 5.2.5) we align the IMU coordinate frame with the 3D scene frame by finding a planar rotation that orients the SMPL head at frame zero to match the camera orientation at the same frame. Mathematically, this entails minimizing the following objective

$$\mathbf{R}_A^* = \operatorname{argmin}_{\mathbf{R}_A \in \mathcal{R}} \|(\log((\mathbf{R}_A \mathbf{R}^H(\boldsymbol{\theta}_0^I))^{\top} \mathbf{R}_0^C))^{\vee}\|_2. \quad (5.11)$$

To obtain head bone orientation, we traverse the kinematic chain of SMPL

$$\mathbf{R}^H(\boldsymbol{\theta}) = \prod_{i \in \mathcal{P}_{\text{Head}}} \exp(\widehat{\boldsymbol{\theta}^i}), \quad (5.12)$$

where $\mathcal{P}_{\text{Head}}$ is an ordered list of all the parents to the head joint and $\boldsymbol{\theta}^i \in \mathbb{R}^3$ denotes the axis-angle subvector of $\boldsymbol{\theta}$ corresponding to joint i . We use the axis-angle parameterization to define the set of rotation matrices

$$\mathcal{R} = \{\exp(\widehat{x\mathbf{e}_z}) : x \in \mathbb{R}\},$$

where $\mathbf{e}_z = [0, 0, 1]^{\top}$ is the z-axis unit vector. The IMU root orientation $\boldsymbol{\psi}_j^I$ and position $\mathbf{t}_j^I$ of each subsequent frame are aligned to the 3D scene reference frame by

$$\boldsymbol{\psi}_j^I = (\log(\mathbf{R}_A^* \exp(\widehat{\boldsymbol{\psi}_j^I})))^{\vee}, \quad \mathbf{t}_j^I = \mathbf{R}_A^* \mathbf{t}_j^I. \quad (5.13)$$

5.3 IMPLEMENTATION DETAILS

We describe the hardware setup, camera localization implementation, contact detection details, optimization hyperparameters, and the batched optimization procedure.

5.3.1 Hardware Setup

Our system uses three types of sensors.

Head-mounted camera. We use a GoPro camera tightly attached at the forehead, recording at 1920×1080 resolution. We ensure that it remains tightly fixed throughout recordings to minimize relative motion between the camera and the head. Intrinsic camera parameters are obtained by photographing a checkerboard pattern and running OpenCV [Bradski, 2000] camera calibration. We use an OpenCV camera model with 2 radial and 2 tangential distortion coefficients.

Body-mounted IMUs. We use the XSens [Paulich et al., 2018] commercial inertial motion capture system, which uses 17 IMUs attached to the body with velcro straps. XSens provides 3D pose and translation estimates using a proprietary Kalman filter and a kinematic model of the human body. It also detects foot contacts with the ground by integrating IMU acceleration with the kinematic model.

3D scene scanner. Scenes are captured with the NavVis M6 [NavVis] mobile capture system, which uses 4 LiDAR sensors and 6 RGB cameras. The scene point cloud is reconstructed from LiDAR and RGB data using a proprietary algorithm, which also provides extrinsic and intrinsic parameters for all database RGB images.

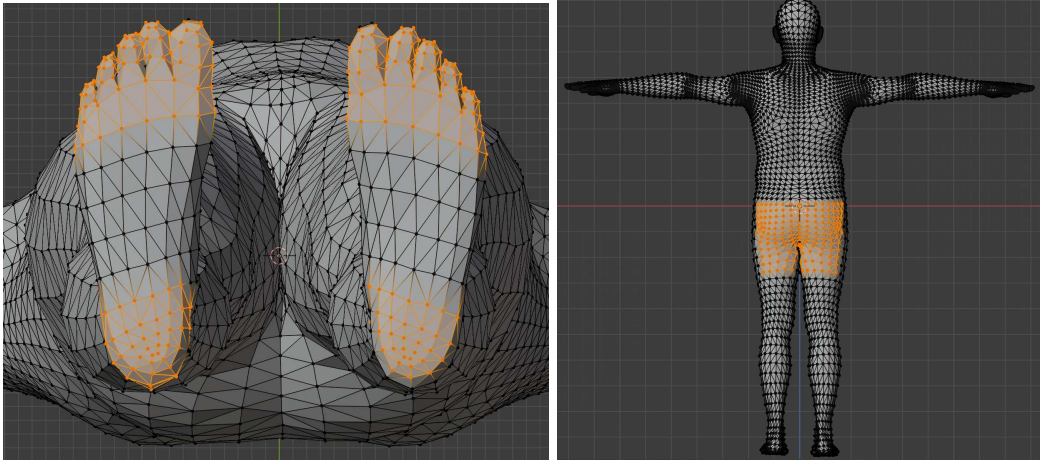


Figure 5.7: Manually defined back and feet vertices.

5.3.2 Camera Localization Implementation

Feature extraction and matching. We use the PyTorch implementations of SuperPoint [DeTone et al., 2018] and SuperGlue [Sarlin et al., 2020], pretrained on MegaDepth [Li and Snavely, 2018] and ScanNet [Dai et al., 2017]. We detect up to 4096 keypoints per image and perform 40 Sinkhorn algorithm iterations at the matching stage.

Database prefiltering. We use the NetVLAD [Arandjelovic et al., 2016] PyTorch implementation pretrained on Pittsburgh 250k [Torii et al., 2018]. For each query frame, we select the 40 most similar database images based on cosine distance between NetVLAD descriptors.

Pose estimation. We use COLMAP [Schönberger and Frahm, 2016, Schönberger et al., 2016] to minimize the reprojection error of matched keypoints. Overall, the pipeline takes approximately 3 seconds to localize a single frame at 1920×1080 resolution on an Nvidia Quadro RTX 8000 GPU.

Rendering for 2D-3D correspondences. To obtain 2D-3D correspondences from database images, we render the scene point cloud using known camera parameters with the surface splatting technique [Zwicker et al., 2001] and a fixed splat size (Fig. 5.4). Alongside the color rendering, we produce a point mapping at the same resolution: each pixel stores the index of the visible point cloud element. Since NavVis scanners use fisheye cameras, the resulting images are heavily distorted. We undistort both camera images and point mappings by computing a mapping from the fisheye image plane to a fixed perspective camera plane, and then run SuperPoint keypoint detection on the undistorted images. Each detected keypoint is mapped to a 3D point on the scene scan via the point mapping.

5.3.3 Contact Detection

Contact vertex groups. We manually define five sets of SMPL vertex indices $\mathcal{B}_k$ ($k \in [1, 2, 3, 4, 5]$) corresponding to the toe and heel regions of the left and right foot and the hip region. These vertices are selected in Blender and are shown in Fig. 5.7. The four foot groups are used with foot contact signals from the IMU system; the hip group is used with our sitting detection heuristic.

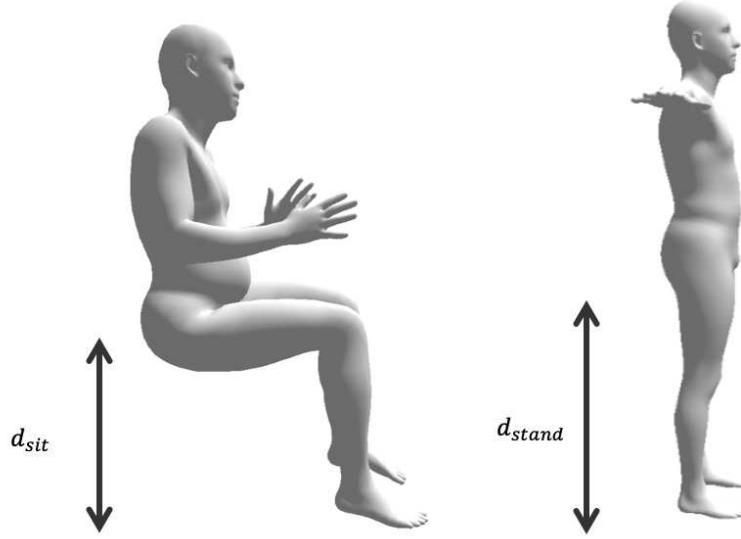


Figure 5.8: **Sitting detection.** We detect sitting by measuring the distance between a person’s root joint and a predefined vertex on the heel. If $d_{\text{sit}} < 0.6 d_{\text{stand}}$ and the frame velocity is below a threshold, we force the hips to be in contact with the scene.

Sitting detection heuristic. While foot contacts are detected directly by the IMUs, hip contacts (sitting) require a separate heuristic (Fig. 5.8). We define a vertex α on the heel of the left foot and compute

$$d^S(\theta, \mathbf{t}) = \|M(\theta, \mathbf{t})_\alpha - \mathbf{t}\|_2 ,$$

which measures the distance from the root joint to the heel. If

$$d^S(\theta_j^I, \mathbf{t}_j^I) < \delta_1 d^S(\mathbf{0}, \mathbf{t}_j^I) \quad \text{and} \quad \|\mathbf{t}_j^I - \mathbf{t}_{j-1}^I\| < \delta_2, \quad (5.14)$$

we assume that the hips are in contact with the scene at frame j . The first condition checks whether the root-to-heel distance has decreased significantly compared to the standing reference (indicating a bent posture), and the second condition checks for low velocity (ruling out crouching during locomotion). We set $\delta_1 = 0.66$ and $\delta_2 = 0.001$.

5.3.4 Scene Voxelization

The interpenetration term $\mathcal{L}_{\text{int}}$ (Sec. 5.2) requires a volumetric occupancy field of the scene. We construct this by voxelizing the scene point cloud into a regular 3D grid with

a voxel size of 5 cm. Each voxel is marked as occupied if it contains at least one scene point. An occupancy function $S : \mathbb{R}^3 \mapsto [0, 1]$ is then defined via trilinear interpolation of the voxel grid, yielding smooth occupancy values for arbitrary query points. For the centroid projection depth π in Eq. (5.8), we use $\pi = 0.01$ m, which projects face centroids slightly inside the SMPL mesh to detect intersections with the scene.

5.3.5 Optimization Hyperparameters

Loss weights. The weights in the objective (Eq. (5.1)) are: $w_i = 0.001$ (interpenetration), $w_s = 0.1$ (scene contact), $w_{sm} = 1.0$ (smoothness). Within the scene contact term, we use $w_c = 1.0$ (contact) and $w_v = 1.0$ (no-slide). Within the smoothness term, we use $w_T = 1.0$ (translation smoothness) and $w_G = 10.0$ (global orientation smoothness).

Filtering algorithm. The filtering algorithm (Alg. 1) uses a velocity ratio threshold $\eta = 3.0$ to identify camera localization outliers and a maximum velocity threshold $\epsilon = 0.5$ m/s as the stopping criterion. For the tangent direction estimation in the initialization (Sec. 5.2.6), we use $\gamma = 3$ frames.

5.3.6 Batched Optimization

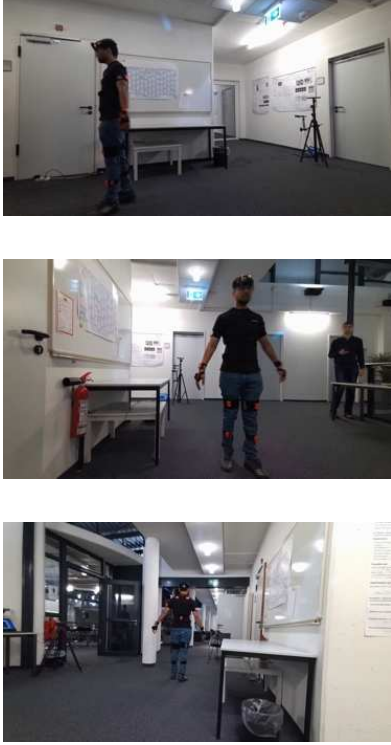
In each iteration of our optimization algorithm, we optimize for 100 frames in a single batch. To ensure a smooth transition between results from adjacent batches, we force the translation and global orientation of the first frame in a batch to be the same as the corresponding parameters of the last frame in the previous batch. We implement our method in PyTorch [Paszke et al., 2019]. We use the Adam optimizer [Kingma and Ba, 2015] with a learning rate of 0.03, and optimize each batch of frames for 200 to 2000 gradient steps. Using an Nvidia Volta V100 GPU, one gradient step takes about a quarter of a second, resulting in approximately 1–8 minutes per batch depending on the number of iterations.

5.4 DATASET

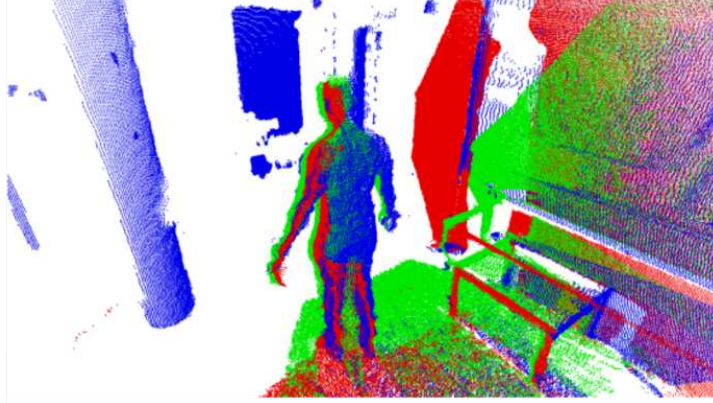
HPS allows us to collect the *HPS dataset* - a dataset of 3D humans interacting with large 3D scenes (300–2500 m^2). Our dataset contains images captured from a head-mounted camera coupled with the reference 3D pose and location of the person in a pre-scanned 3D scene. We capture 7 people in 8 large scenes performing activities such as exercising, reading, eating, lecturing, using a computer, making coffee, dancing. *All subjects have agreed to release their data for research purposes.* In total, the dataset provides more than 300K synchronized RGB images coupled with the reference 3D pose and location. This is a continuous dataset and we plan to improve it by adding more long-term motion recordings with a variety of scene interactions. Fig. 5.12 shows qualitative results from our dataset.

5.5 EXPERIMENTS

A) Input from Kinects



B) Raw point-cloud



C) Filtered point-cloud

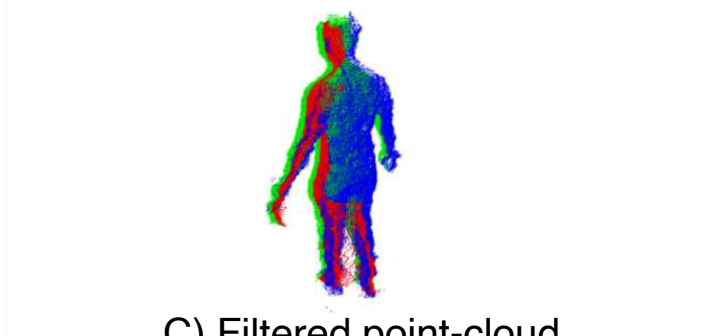


Figure 5.9: **Setup of our ground truth recording system.** A) Sample of images captured by Kinects. B) Raw point-cloud obtained by unprojecting the depth (RGB colors mark points from 1st, 2nd and 3rd Kinects respectively). C) Point-cloud after applying background removal procedure.

In this section, we demonstrate that HPS does not drift with time and distance traveled, is robust to non-persistent camera localization outliers, and satisfies scene constraints (feet stay on the ground during contact, and do not slide).

Since this is the first method to track humans in large scenes, there exist no published baselines to compare to, and ground truth 3D human pose and localization can not be obtained for un-bounded areas like ours. Hence, we use depth cameras to obtain ground truth dynamic point-clouds of the human in a small sub-area of the scene. Subjects are then asked to move freely in the large scene, and return to the sub-area, where we can evaluate accuracy and drift.

We evaluate the accuracy of our method by comparing our output SMPL mesh (including translation) with a dynamic *ground-truth point-cloud* of the person obtained from synchronized and calibrated 3 external depth cameras (Azure Kinect RGB-D sensors[[Azure-Kinect](#)]). In our experiments, we report the bidirectional Chamfer distance between the SMPL model (result) and ground truth point-cloud from depth sensors *without* Procrustes alignment. Crucially, the absence of Procrustes alignment

means that the reported error reflects both pose accuracy and global localization accuracy in the scene coordinate frame. This is a significantly harder evaluation setting than the one used in standard pose estimation benchmarks, which typically align predictions to ground truth before computing joint errors, thereby discarding any global positioning information.

5.5.1 Obtaining Ground Truth Point Clouds

To obtain the ground truth data for our comparisons we use a system of 3 Azure Kinect RGB-D sensors[Azure-Kinect] in a time-synchronized mode. The system is set up in a way that it covers all regions of the body and provides a 360° view of the subject every $1/30$ of a second (Fig. 5.9).

To obtain the RGB-D videos from Kinects we use a custom recorder written in C++ running on a separate computer for each Kinect. To ensure time synchronization, Kinects are connected sequentially through the special time synchronization input and recorders are controlled over the wireless network. The video is recorded at 30 FPS with a color resolution of 1920×1080 pixels and a depth resolution of 640×576 pixels.

To register the ground truth point-cloud to the static 3D scanned scene, we use the following 3-stage method:

- Visual localization: We obtain an approximate camera position for each depth-camera using the same visual localization method described in the paper.
- Depthmap-to-scene ICP: We align the partial point-cloud of each depth camera with the scene using ICP [Besl and McKay, 1992]. ICP is initialized using the camera parameters from the previous stage.
- Manual correction: Some results can still be erroneous due to depthmap artifacts or due to incorrect ICP initialization. These results are corrected manually. As a final step, we run ICP again.

Video recording preparation. Prior to the actual ground-truth video recording, we record a short video of the scene from all kinects with no one present in the field of view. Frames are averaged and average depth is used in 2 ways: in depthmap-to-scene ICP alignment and for the background subtraction.

Background subtraction. To compute our metrics using obtained point-clouds we need to make sure that we get only points corresponding to a subject. To achieve this, we first subtract the background from depth videos by ignoring all pixels with depth more or equal to the prerecorded empty scene depthmap. After that, we unproject the points from all 3 Kinects to the 3D space and run DBSCAN [Ester et al., 1996] clustering with the maximum distance between clusters of 0.12 meters. Finally, we remove all points that are not in the biggest cluster. An example of the result is shown in Fig. 5.9,C.

We validate the accuracy of the ground truth registration by measuring the residual alignment error between the Kinect depth maps and the NavVis scene scan after ICP convergence. The mean point-to-plane distance is below 1 cm across all three Kinects,

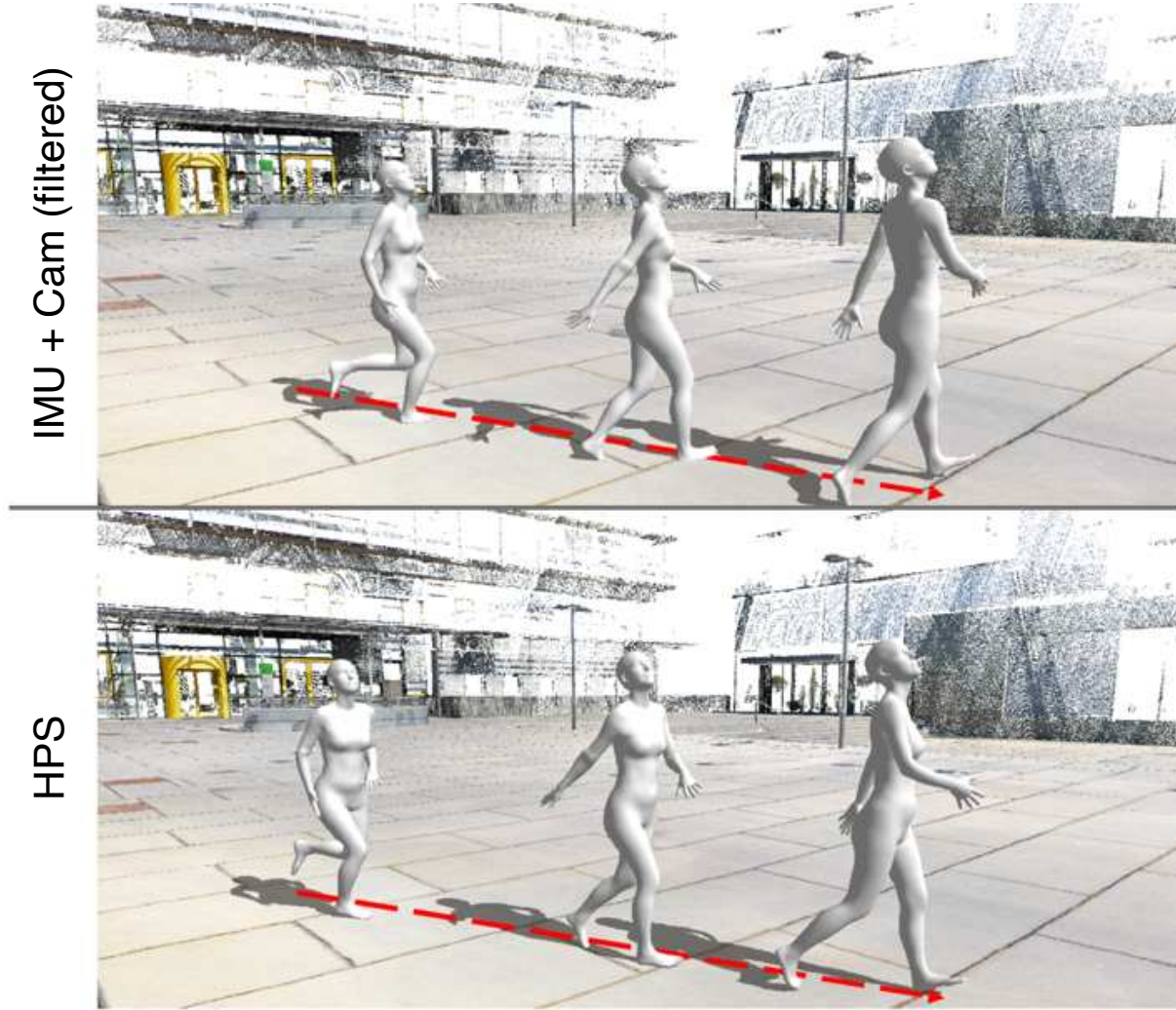


Figure 5.10: **Global body orientation improvement.** Combining the IMU pose with position from camera localization (IMU+Cam (filtered)) results in unnatural motion—the global body orientation does not face the direction of movement. By contrast, HPS correctly estimates the global orientation.

which is well within the depth sensor noise of the Azure Kinect (approximately 1–2 cm at the operating range of 1–3 m). This confirms that the ground truth point clouds are registered with the scene coordinate frame with sufficient accuracy for evaluating our method, whose localization errors are typically an order of magnitude larger.

5.5.2 Quantitative Evaluation:

Movements: For quantitative evaluation, we record using the following protocol: a subject starts within the recording volume of the 3 RGB-D sensors and performs different actions including standing in A-pose, leaning on a table and walking. The subject then leaves the RGB-D recording volume and moves within the scene, returns back and repeats the same actions inside the recording areas of the RGB-D sensors.

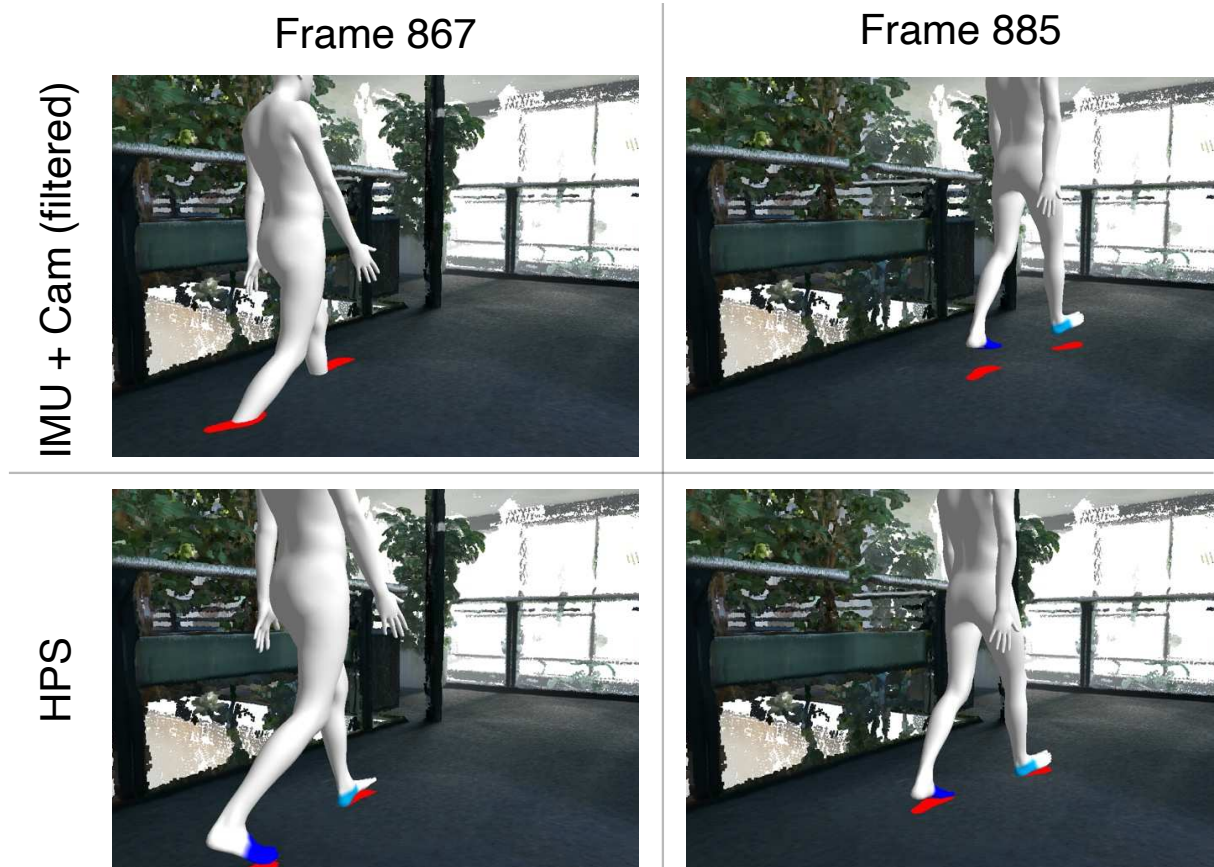


Figure 5.11: **Effect of integrating predicted 3D scene contacts.** As a baseline we used camera localization results for localizing SMPL model. Red regions mark closest surface to feet, heels and toes are colored with light blue and blue when IMUs detect ground contact.

The subject repeats the motion several times, each time choosing a different path. **Baselines:** As mentioned previously, there are no established baselines to compare to, as no other method tackles the same problem. Hence, to understand the influence of each component of HPS, we implement and compare to the following baselines: 1) **IMU:** pure IMU tracker, 2) **IMU+Cam:** pose from IMU, and translation from camera self-localization, 3) **IMU+Cam (filtered):** Like IMU+Cam but with filtered camera outliers (same as in Sec. 5.2.6), 4) **HPS w/o scene:** Optimization without 3D scene contact constraints. **Drift and Outliers:** In Tabs. 5.1 and 5.2, we compare HPS to the baselines. We observe that the IMU-only method drifts over time, particularly the global body translation and orientation. IMU+Cam corrects drift with camera localization, but produces translation noise and severe jitter. IMU+Cam (filtered) mitigates this, but lacks precision and suffers from global orientation errors (Fig. 5.10). HPS w/o scene further improves results, but without knowledge about foot-scene contacts, it is easily misled by incorrect camera localization, and the subject penetrates or flies over the ground. HPS results satisfy these scene constraints, and consistently achieves the best accuracy. HPS is inaccurate when filtered camera localization fails for a long period (see 2nd and 4th rows of Tab. 5.2), but it can recover once the camera

Table 5.1: **Drift and camera outliers.** 3D error (in cm) for the subject standing in A-pose after moving freely around the scene.

Distance traveled	IMU	IMU+Cam	IMU+Cam (filt.)	HPS w/o scene	HPS
At start	6.85	9.24	10.48	7.21	5.20
70 m	54.49	742.32	6.93	6.48	4.60
200 m	69.02	136.81	5.93	5.80	4.26
380 m	108.44	32.17	6.15	5.69	4.53

Table 5.2: **Drift and camera outliers (dynamic).** 3D error (in cm) for the subject walking, standing and leaning on the table, after moving around the scene. Error is measured from the dynamic ground truth point-cloud to the result (3D mesh in motion). Rows indicate distance traveled before evaluation.

Distance traveled	IMU	IMU+Cam	IMU+Cam (filt.)	HPS w/o scene	HPS
At start	6.77	2189.75	10.05	9.19	6.44
70 m	51.57	569.71	21.75	20.68	15.96
200 m	61.11	719.44	7.34	6.67	4.76
380 m	100.44	261.72	12.59	11.96	10.07

Table 5.3: **Foot contact evaluation.** For frames when foot contact is detected, we report (in cm) **Distance to surface**: average distance between foot vertices and the scene. **Foot Sliding**: average distance on the surface plane between foot vertices in two successive frames. Numbers are computed for a 3 minute long walking sequence.

Metric	IMU	IMU+Cam	IMU+Cam (filt.)	HPS w/o scene	HPS
Distance to surface	188.38	39.8	0.95	0.32	0.056
Foot Sliding	0.92	52.09	1.75	2.00	0.90

can be well localized in nearby frames (see 3rd row of Tab. 5.2). Overall, the analysis reveals that HPS does not drift (error does not increase with distance traveled or time), and is robust to non-persistent camera localization outliers.

Foot Contacts: We also report in Tab. 5.3 the average foot to scene distance and foot sliding along the surface distance during contacts detected with the IMUs. HPS better preserves foot contact with the surface than the baselines, and has slightly lower foot-sliding compared to the raw IMU tracker, which also integrates constraints with a *virtual* imaginary ground.

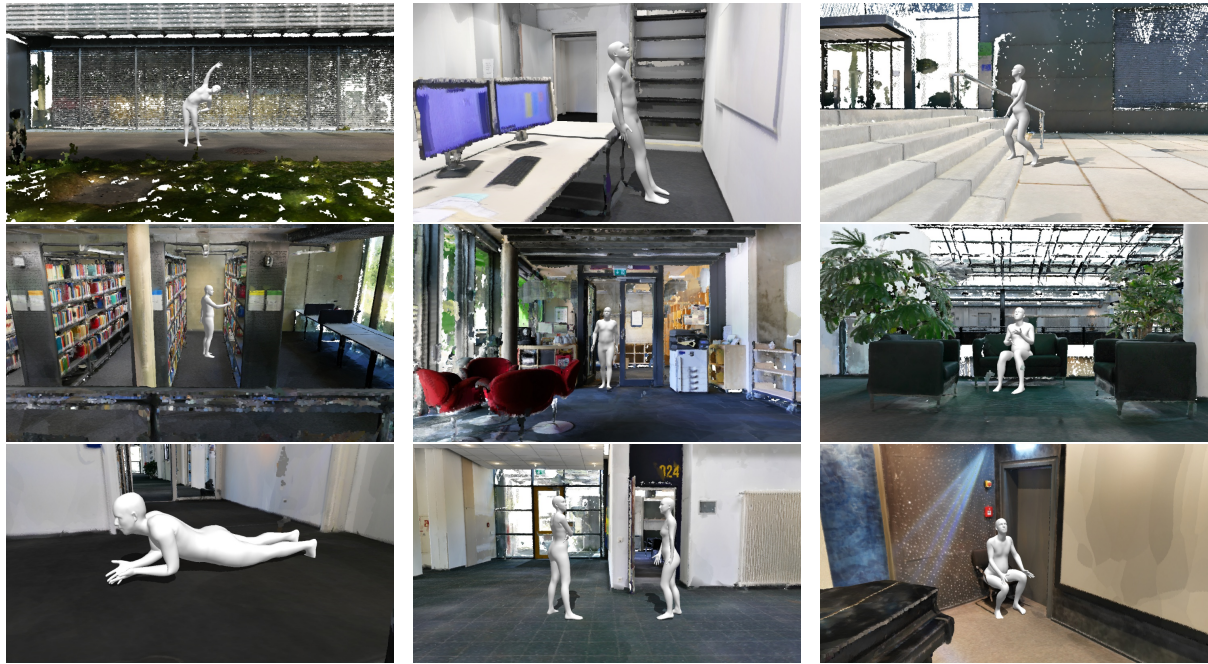


Figure 5.12: **We show qualitative results of our method.** Our method can localize and estimate the 3D pose of people performing activities as diverse as exercising, dancing, reading, sitting, eating, talking in a range of indoor and outdoor scenes, all *without* external cameras.

5.5.3 Qualitative Evaluation

In Fig. 5.11 we show qualitatively the effect of foot contact constraints. As we encourage contact with the scene surface each time a contact is detected, the resulting human mesh does not fly in the air or penetrate the ground like the baseline. The motion is more stable and physically correct. In Fig. 5.12 we show several examples of humans performing different actions including sitting, leaning on a table, dancing or performing push-ups.

These results demonstrate that HPS generalizes across a wide range of indoor and outdoor environments, including offices, libraries, staircases, and open corridors. The method handles diverse activities—from stationary poses such as reading and sitting, to dynamic motions such as exercising and playing piano—without requiring environment-specific tuning. Notably, the scene contact constraints produce plausible foot placement on varied surface types including flat floors, stairs, and elevated platforms, confirming that the voxelized scene representation is sufficiently flexible to accommodate complex scene geometry.

5.6 CONCLUSION

We introduced HPS, to the best of our knowledge, the first method to recover the full 3D pose of a human registered with a pre-scanned 3D environment from *only*

body-mounted sensors. We demonstrate that HPS produces natural human motion, removes the typical drift of pure IMU based systems, and is robust to non-persistent camera localization outliers. HPS is able to continuously track humans in large scenes ($300\text{--}2500\text{ m}^2$) including multiple rooms, and outdoors.

The error of HPS does not accumulate with time or distance traveled. However, if camera localization is inaccurate for long periods of time, HPS performance deteriorates. This can be seen in the errors, which range from 4cm to 15cm . Two factors influence localization accuracy: 1) Lack of features, 2) Scene changes between the static 3D scan and the real images, captured from the head camera.

While HPS achieves a remarkable accuracy and stability, many applications will require errors in localization and pose of less than 1cm . We envision many exciting research directions to improve HPS. First, a local map could be built on the fly to update the large static scene with objects that move, and to add new objects. This would improve localization and allow interaction with dynamic objects. In fact, it is not inconceivable that, in the future, a dynamic 3D reconstruction of the world will be stored on the cloud, and will be continuously updated from cameras worn by people [Aria]. Second, camera localization could incorporate semantics [Bloesch et al., 2018, Zhi et al., 2019], for example detecting which objects are static and reliable. Third, while HPS integrates foot and hip contacts, scene constraints with other body parts should further improve results. More powerful would be to incorporate a learned model to *anticipate human intent* to improve tracking. For example, we could detect when the person is about to sit on a chair, or about to grab an object. Conversely, HPS can be used to build human models of environment interaction and navigation from long recordings, consisting of several hours of motion.

HPS is the first step in a new exciting research direction. We have released the HPS dataset and code for research use, and hope it will foster new methods to perceive and model scenes and humans from an ego-centric perspective.

CoHMo: GENERATING CONTINUAL HUMAN MOTION IN DIVERSE 3D SCENES

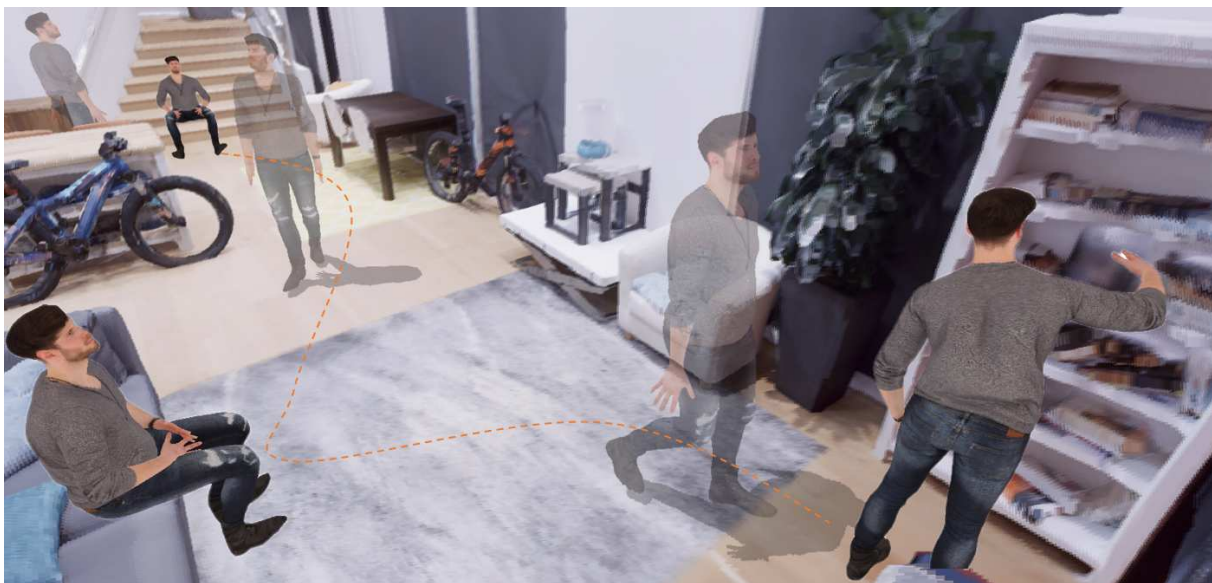


Figure 6.1: **Animator-guided human motion synthesis in diverse 3D scenes.** Our method synthesizes diverse actions such as sitting and grabbing in diverse 3D scenes.

While Ch. 5 addresses the capture of human behavior, this chapter tackles the complementary problem of synthesizing it. We introduce CoHMo (**C**ontinual **H**uman **M**otion), a method to synthesize animator-guided human motion across 3D scenes. Given a set of sparse (3 or 4) joint locations and a seed motion sequence in a 3D scene, our method generates a plausible motion sequence starting from the seed motion while satisfying the constraints imposed by the provided keypoints. We decompose the continual motion synthesis problem into walking along paths and transitioning in and out of the actions specified by the keypoints, which enables generation of long motions that satisfy scene constraints without explicitly incorporating scene information. Our method is trained only using scene agnostic mocap data. Our approach is deployable across 3D scenes with various geometries. For achieving plausible continual motion synthesis without drift, our key contribution is to iteratively generate motion in a goal-centric canonical coordinate frame where the next immediate target is situated at the origin. Our model can generate long sequences of diverse actions such as grabbing, sitting and leaning chained together in arbitrary order, demonstrated on scenes of varying

geometry: HPS, Replica, Matterport, ScanNet scenes. Several experiments demonstrate that our method outperforms existing methods that navigate paths in 3D scenes. Video results are available at https://virtualhumans.mpi-inf.mpg.de/origin_2/.

6.1 INTRODUCTION

Our goal is to generate animator guided rich long-term human behavior in arbitrary 3D scenes, including a variety of actions and transitions between them. Such a system should allow for goal-directed generation of humans moving about from one place to another, for example, walking towards the couch to sit on it, and then standing up and approaching the shelf to grab something from it, as illustrated in Fig. 6.1. It should allow users to specify with minimal interaction what kind of actions to perform, while keeping the realism and expressivity required for applications such as synthetic data generation, robotics, VR/AR, gaming, etc.

While the community has seen promising progress in animator guided motion synthesis in 3D scenes, most works are restricted to a single action and do not handle transitions [Zhang et al., 2022, Wu et al., 2022, Taheri et al., 2022], preventing them from producing long range diverse motion. They are also not deployable in a wide variety of real scenes [Starke et al., 2019, Wang et al., 2021a, 2022, Hassan et al., 2021a]. The reason for this is that they synthesize motion by conditioning on scene geometry and require training on a dataset featuring 3D humans interacting in 3D scenes and objects [Hassan et al., 2019, 2021a, Zhang et al., 2022]. Generalizing these methods to arbitrary 3D scenes would require collecting motion data registered to a myriad of possible 3D scenes and objects, which is not scalable.

In contrast, humans can navigate cluttered scenes, pick objects from a shelf they have never seen before, and sit on novel furniture and surfaces. Most of the clutter in the scene is often ignored, and what matters most are not the exact details of the object / scene geometry but whether they afford each action. Our hypothesis is that motion, to a large extent, is driven to avoid obstacles and focused on reaching the next immediate goal or target contacts in the environments. Thus, it should be possible to generate human motion without accounting for all the details in the 3D scene.

Based on this insight, we propose a novel framework for animator-guided motion synthesis in 3D scenes without relying on scene-registered motion data. As such, our method can be trained on regular mocap data, which is relatively easily captured and abundantly available [Mahmood et al., 2019]. Since our method does not explicitly condition on the geometry of the scene, it can be deployed across 3D scenes with varied geometry.

Our method relies on two key observations: first, we can represent actions in a 3D scene as a set of sparse desired target contacts (we use 3 or 4 contacts such as the location of the two feet and a hand or the location of two feet and the root) to be reached, which we refer to as *action keypoints*. These keypoints can be provided by an animator using an interface or generated by automated heuristics, allowing animators to trade off the speed and control over the generation motion. An interesting finding in this chapter is that *action keypoints* are a powerful abstraction of several

actions in 3D scenes, and can be used to execute instructions such as “sit there” or “grab at this height”. Second, avoiding obstacles in 3D scenes can be achieved by path following. The challenge is to follow arbitrarily long paths, smoothly making the human transition into and out of the action, and then walk towards the next target. For this, we break down motion into three pieces: walking, transition into and out of an action. For path following and transitions, we introduce the idea of training a motion synthesis model entirely with *scene-agnostic motion data* to reach the origin of a *canonical coordinate frame*. For navigating paths, this model is sampled iteratively to converge at the origin of the *canonical coordinate frame* defined using waypoints and tangents on the path. For transitions in and out of actions, motion is synthesized by placing target poses at the origin of the canonical coordinate frame. By iteratively synthesizing motion in the *canonical coordinate frame*, our method allows for long range motion synthesis that transitions between walks and various actions in a 3D scene.

Unlike existing methods for motion synthesis [Hassan et al., 2021a, Starke et al., 2019], our method allows for synthesizing motion without requiring any manual phase or action annotation.

For the first time, we demonstrate long-range human motion synthesis on a wide range of scene datasets: Replica [Straub et al., 2019], Matterport [Chang et al., 2017], HPS [Guzov* et al., 2021], Scannet [Dai et al., 2017] and a NeRF scene. Furthermore, we show that our model can perform actions at different places, such as grabbing from any shelf, table or cabinet at any height or sitting on any surface that affords sitting. We will make our code and models publicly available which can be used by animators to synthesize goal directed human motion across 3D scenes.

To summarize, our contributions are as follows:

1. We present a method that departs from existing methods for motion synthesis in 3D scenes by only using regular motion capture data and that is deployable across varied 3D scenes.
2. We introduce a novel idea of iteratively converging motion at the origin of a canonical coordinate frame, which allows us to synthesize long-range motion in 3D scenes.

6.2 METHOD

Our method takes as input a seed motion sequence and a list of N_a action keypoints $\{\mathbf{a}_1, \dots, \mathbf{a}_{N_a}\}$ specifying interactions at different locations in the scene. These keypoints can be specified by users or generated using language commands and scene segmentations (Sec. 6.3.1). Our goal is to synthesize motion that starts at the seed motion and transitions in and out each of the action keypoints in the input list.

The first step is to optimize for a pose that fits the action keypoints at target locations using Inverse Kinematics and a pose-prior (Sec. 6.2.1). These poses along with the starting seed motion act as anchors to guide the motion synthesis process.

Using scene-agnostic motion capture data placed in a goal-centric canonical coordinate frame (Sec. 6.2.2), we train *Walknet* (Sec. 6.2.4) to synthesize walking motion that

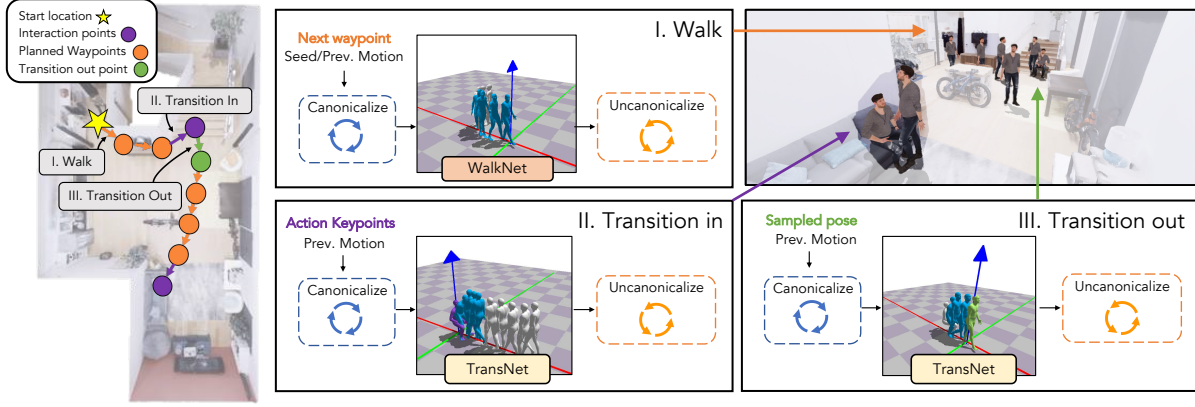


Figure 6.2: **Overview of our method.** We generate human motion satisfying action keypoint constraints by dividing it into 3 stages: a *Walk Motion*, which animates the human as it walks between actions, a *Transition-In*, which blends the walking motion with the target pose computed from keypoints and a *Transition-Out*, which animates the human back to the walking pose. We use an autoregressive transformer, *WalkNet*, to synthesize the walking motion, and a masked-autoencoder transformer to generate the blending motion *TransNet*.

converges at the origin of a canonical coordinate frame, and *TransNet* (Sec. 6.2.5) that synthesizes motion in between a seed motion sequence and a target pose also at the origin. At test time (see Fig. 6.2), *WalkNet* is used to reach canonicalized intermediate goals along a path computed with a path planning algorithm, thus creating long motion by successively reaching the origin. Once the walking motion reaches the vicinity of an anchor pose, *TransNet* synthesizes transition from walking motion to the anchor pose and vice versa. This allows to synthesize motion in 3D scenes without the need for motion data coupled with 3D scenes. Our framework is general and highly modular, which allows it to be updated with novel methods for motion synthesis.

We use the SMPL body model [Loper et al., 2015] to represent the human subject. SMPL is a differentiable function $M(\beta, \psi, \theta, \mathbf{t})$ that maps shape β , global body orientation ψ , body pose θ , and translation $\mathbf{t}$ parameters to the vertices of a human mesh along with the 3D joint locations of the SMPL skeleton. We assume that β remains static throughout our method. We denote motion sequences as an ordered list of SMPL parameter tuples. For example $\mathcal{C} = [(\mathbf{r}, \psi, \theta)]_{j=1:D}$ denotes a motion sequence of D frames, where $\mathbf{r}$ denotes the root position, corresponding to the SMPL translation parameter $\mathbf{t}$.

6.2.1 From Action Keypoints to an Anchor Pose

The first step is to infer a pose from the action keypoints in a target location $\mathbf{a} = \{\mathbf{k}_i\}_{i=1}^P$, where $\mathbf{k}_i \in \mathbb{R}^3$ indicates the desired locations for corresponding SMPL joints denoted as $m_i(\cdot)$. We find as few as three to four joints ($P = 3, 4$) are usually sufficient. Since the problem is heavily under-constrained we optimize the latent space of VPOSER [Pavlakos et al., 2019] denoted as $\mathbf{z}$. Denoting $f(\mathbf{z}) \mapsto (\psi, \theta)$ as

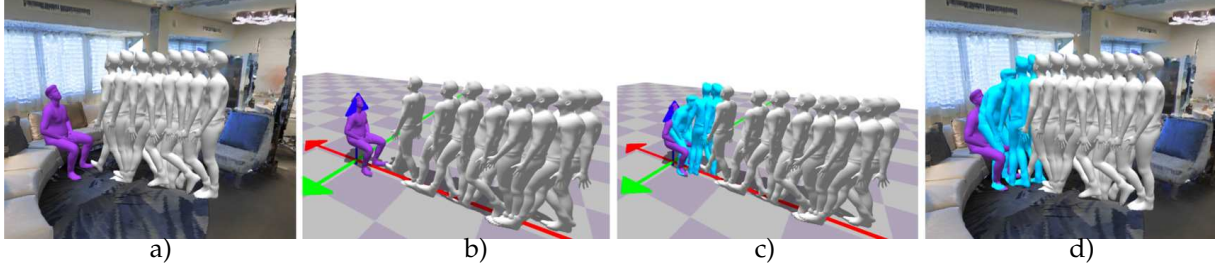


Figure 6.3: **Transition synthesis in the canonical coordinate frame.** Using a) the motion-anchor pose in the 3D scene (purple), b) we move the motion sequence into the canonical coordinate frame. c) There *TransNet* synthesizes transitions (blue) between the input motion and the pose placed at the origin (purple). d) Once the motion is synthesized, we move it back to the scene coordinate frame.

the mapping from the latent space $\mathbf{z}$ to the SMPL root orientation and body pose parameters, we minimize the following objective

$$\mathbf{z}, \mathbf{t} = \arg \min_{\mathbf{z}, \mathbf{t}} \sum_{i=1}^P \|m_i(f(\mathbf{z}), \mathbf{t}) - \mathbf{k}_i\|_2 \quad (6.1)$$

Please see the supplementary material for further details to make the optimization well behaved. We repeat this step for each target action $\mathbf{a}_1 \dots \mathbf{a}_{N_a}$, obtaining N_a pose-anchors $\mathcal{A} = \{\mathbf{t}_i^A, \psi_i^A, \theta_i^A\}_{i=1:N_a}$.

6.2.2 Canonical Coordinate Frame

One of our key ideas to synthesize motion in 3D scenes is to make transformers synthesize motion that always converge at the origin of a canonical coordinate frame. This way at test time long motion is composed by consecutively going to the next goal placed at the origin. Thus, we canonicalize the training sequence clips by using the planar translation $\mathbf{t}_C$, and rotation $\mathbf{R}_C$ of the last frame in a sequence clip as follows

$$\psi_j^C = \mathbf{R}_C^{-1} \psi_j, \mathbf{r}_j^C = \mathbf{R}_C^{-1}(\mathbf{r}_j - \mathbf{t}_C) . \quad (6.2)$$

By construction, this transformation outputs a new set of L frames $[(\mathbf{r}^C, \psi^C, \theta)]_{j=1:L}$, where the last pose is at the origin and oriented towards a canonical axis of orientation γ (arbitrary fixed axis). Let $\mathbf{X}$ denote a matrix whose columns are vectorized motion parameters (pose and translation combined). We will use the following notation to denote the canonicalization in Eq. (6.2) for a full sequence as

$$\mathbf{X}^C = C(\mathbf{X}; \mathbf{R}_C, \mathbf{t}_C) \quad (6.3)$$

Synthesizing motion in the goal-centric canonical coordinate frame, allows us to synthesize walking motion along paths in a 3D scene (Sec. 6.2.4) and transitions in and out of actions (Sec. 6.2.5) without the need for scene registered data.



Figure 6.4: **Scene map and path planning.** Right: scene point cloud projected down to get a map of the scene. Left: A* trajectory with keypoints in black and tangents in red.

6.2.3 Path Planning

To obtain a path from the starting position to the end, we employ the A* algorithm. To do so, we first obtain a map representation of the 3D scene. We use a horizontal plane to cut the pointcloud at some fixed height above the floor and another horizontal plane to cut the pointcloud some fixed height below the ceiling. Then we project all the remaining pointcloud vertices between the two horizontal planes onto the ground. The locations in the ground plane (2D map) where pointcloud vertices project to are marked as obstacles. We use a dilation operation to expand the obstacle regions so that A* finds paths well away from the walls. We show a sample scene map and a sample trajectory in Fig. 6.4.

6.2.4 WalkNet

Training. Using walking sequence clips of variable length L canonicalized (last pose at origin), we train *WalkNet*. *WalkNet* takes K motion frames as input $\mathcal{W}_{inp}$ and predicts the next K frames in the sequence $\mathcal{W}_{out}$:

$$\mathcal{W}_{inp} = [(\mathbf{r}^W, \psi^W, \theta^W)_j]_{j=1:K}, \quad \mathcal{W}_{out} = [(\mathbf{r}^W, \psi^W, \theta^W)_j]_{j=K:2K}.$$

The training sub-clips of size $2K < L$ are randomly sampled from the training walking sequences.

Expressing sequences as matrices (columns are translations and poses) as explained in the previous section, the transformer takes as input a matrix $\mathbf{X}_{in} \in \mathbb{R}^{K \times 219}$ and outputs a matrix $\mathbf{X}_{out} \in \mathbb{R}^{K \times 219}$, where $219 = 3 + 24 \times 9$ is the dimensionality of the vectorized root translation and joint rotation matrices. We denote the learned mapping as $T : \mathbf{X}_{in} \mapsto \mathbf{X}_{out}$. Note that we input the pose as vectorized joint rotation matrices, which make learning more stable compared to using joint angles.

Test time. We use the *WalkNet* to follow long paths, by breaking the path into intermediate goals that are canonicalized to the origin (Fig. 6.5). To traverse scenes avoiding ob-

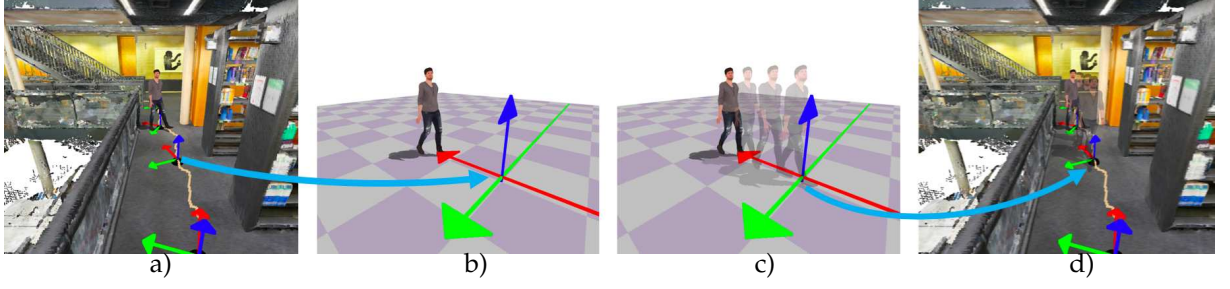


Figure 6.5: **Walking motion synthesis via goal-centric canonicalization.** a) Using keypoints and tangents along a path, we move motion from the scene coordinate frame into b) the goal-centric canonical coordinate frame, where c) *WalkNet* synthesizes motion that converges at the origin of the coordinate frame. d) Once the synthesized motion reaches the origin, we move it back to the scene coordinate frame, and iterate the process to reach the next waypoint.

stacles, we compute the path between the seed motion $\mathcal{I}$ and the first anchor pose $\mathcal{A}_1$ using A-star[Hart et al., 1968]. Along the path, we sample G intermediate goals and compute tangents to the path: $\{\mathbf{q}_g, \mathbf{l}_g \in \mathbb{R}^3\}_{g=1\dots G}$, where $\mathbf{q}_g$ is the goal position and $\mathbf{l}_g$ is the tangent direction. Then we recursively canonicalize such that tangents $\mathbf{l}_g$ align with the canonical axis γ . Hence, canonical translation and rotation are computed as follows

$$\mathbf{t}_C = \mathbf{q}_g, \quad \mathbf{R}_C = \exp(\widehat{\mathbf{l}_g \times \gamma}) \quad (6.4)$$

where $\exp(\cdot)$ is the exponential map recovering the rotation from the skew-symmetric matrix $\widehat{\mathbf{l}_g \times \gamma}$. With this, the motion sequence from goal $g - 1$ to goal g is obtained by canonicalizing, predicting future motion with the learned mapping T and uncanonicalizing

$$\mathbf{X}_{in} \xrightarrow{C(\cdot, \mathbf{R}_C, \mathbf{t}_C)} \mathbf{X}_{in}^C \xrightarrow{T} \mathbf{X}_{out}^C \xrightarrow{C(\cdot, \mathbf{R}_C^\top, -\mathbf{t}_C)} \mathbf{X}_{out}. \quad (6.5)$$

Although the transformer outputs K future frames, at test time, we use it recursively with a stride of 1 for better performance. That means we effectively predict one pose at a time, and we discard the $K + 1 : 2K$ frames. In this manner, the motion always goes to the origin, we never have to explicitly send the goal coordinates as input to the network, and we do not drift. When we are sufficiently close to an anchor pose, we predict the transition with TransNet.

6.2.5 TransNet

We synthesize transitions between walks and actions again in a canonicalized frame. To do so, we train *TransNet* - a transformer based motion inbetweenner - using AMASS sequences placed in the canonical coordinate frame. The task of *TransNet* is to fill in the motion from a seed sequence $\mathbf{X}_{in}$ to a target *anchor pose*.

Training. We train *TransNet* by asking it to recover training clips from masked out ones. We observe that directly asking to infill many frames does not work reliably.

Inspired by training of language models [Devlin et al., 2019], we progressively grow the mask during training until the desired length. Formally, let $\mathbf{X}$ be a training clip of length M , let $\mathbf{V} \in [0, 1]^{M \times 219}$ be a matrix mask with zero-column vectors for the frames that need to be infilled. The network is tasked to recover $\mathbf{X}$ from the masked out matrix $\mathbf{X} \odot \mathbf{V}$. The mask $\mathbf{V}$ is progressively grown to mask all the motion frames between $\frac{M}{2}$ to $M - 1$ frames – everything except the seed motion and the last anchor pose. For more details, please see supp. mat.

Test time We use *TransNet* to synthesize transitions in 3D scenes by moving $\frac{M}{2}$ frames of a motion sequence into the canonical coordinate frame by using the orientation and position of the motion-anchor pose - the motion-anchor pose is thus placed at the origin of the canonical coordinate frame. *TransNet* is then tasked to infill the missing frames. (Fig. 6.3)

6.2.6 Chained actions

With our models and representations we can chain actions trivially. At run time, we have to satisfy an arbitrary number of action keypoints $\{\mathbf{a}_1, \dots, \mathbf{a}_{N_a}\}$ at different locations. First we compute anchor poses as explained in Sec. 6.2.1. Obstacle free paths connecting the locations of actions are computed with A^* . We rely on *WalkNet* to follow paths until we are sufficiently close to the first anchor pose. Feeding *TransNet* with the last $M/2$ predicted frames of *WalkNet* and the anchor pose, we predict the transition into the first anchor pose. To transition out we also use *TransNet* with no modification. We sample a location along the path from $\mathbf{a}_1$ to $\mathbf{a}_2$ at a fixed distance δ and place a walking pose from our database. *TransNet* then can transition into this walking pose (Fig. 6.3). Then we activate *WalkNet* and the process is repeated until all actions are executed. In addition, we can repeatedly use *TransNet* to execute several actions at the same location, like grabbing at different heights.

6.3 IMPLEMENTATION DETAILS

We provide details on keypoint generation in scenes, architectural details of the transformer networks, the inverse kinematics heuristics used to stabilize motion anchors, and the progressive mask-growing strategy employed during training.

6.3.1 Generating Keypoints in a Scene

Keypoints can be efficiently collected using a 3D user interface, as described in the supp. mat, or keypoints can be inferred from the geometry of the scene, and can be therefore generated via action labels or language. An example of automatic KP generation can be seen in Fig. 6.6. Given a point cloud of the scene with semantic labels and a language description of a task, we can use simple heuristics to generate keypoints that can synthesize the described motion.

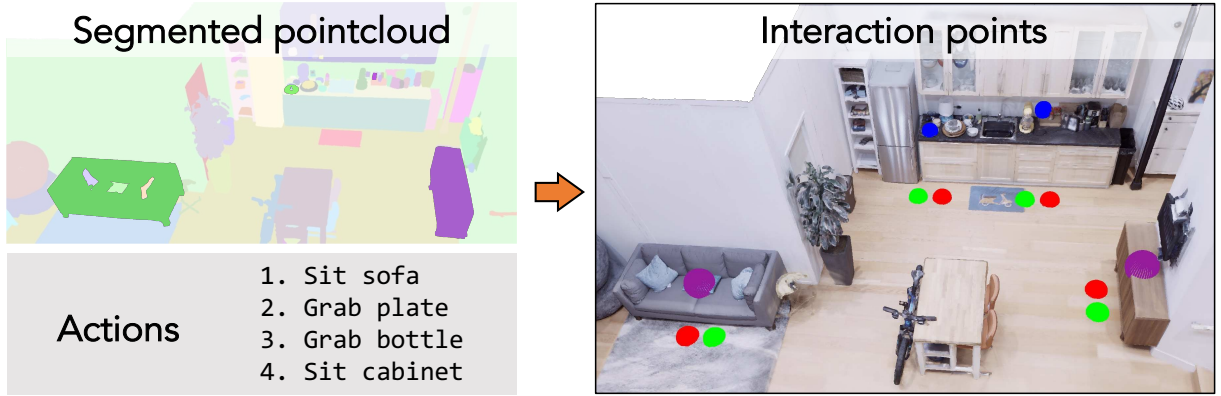


Figure 6.6: **Automatic keypoint placement in 3D scenes.** Using language instruction and semantic segmentation, keypoints can be automatically placed in a 3D scene.

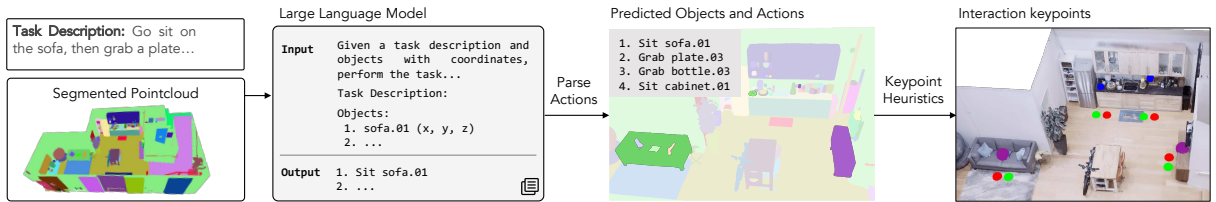


Figure 6.7: **Keypoint generation pipeline.** Given a pointcloud with semantic labels and an activity description, we generate keypoints that allow us to synthesize motion describing the activity. The red and green points correspond to the right and left feet respectively, the blue point corresponds to the hand and the purple point corresponds to the root.

6.3.1.1 User Interface for Manual Placement of Keypoints

We develop a real time user interface that allows for navigating 3D scenes much like a gaming interface which allows users to navigate through a 3D scene. It also allows users to click on points for which they need interaction in a 3D scene. For a visualization of this, please see the supp. video.

6.3.1.2 Language-Based Placement of Keypoints

We also design a method to automatically generate keypoints given a pointcloud with semantic labels and a set of actions or a language description of an activity. We assume that the semantic labels from the pointcloud are given. As a result, we can generate long range motion in 3D scenes that follow a language description. We describe below our approach.

Processing the scene pointcloud: We start by processing the scene pointcloud to be able to place keypoints. We first obtain the navigable area in the scene using the process described in Sec. 6.2.3. We then cluster points according to the object they belong to, using the semantic labels. Finally, we compute the height map of the scene,

Input Given a description of a task and objects with coordinates, execute the task using the objects and only the following actions: 1. Grab object 2. Sit object 3. Place object on object. Note that the coordinate y corresponds to the height. Do not use any other action. Description: Get the top most book and sit on the sofa to read, then place the book on the bookshelf. Objects: 1. Book.1 (x: 3.0, y: 1.0, z:0.4) 2. Book.2 (x: 3.0, y: 1.5, z:0.1) 3. Book.3 (x: 4.0, y: 0.5, z:0.1) 4. Sofa.1 (x: 4.0, y: 0.5, z:8.0) 5. Bookshelf.1 (x: 4.0, y: 0.5, z:8.0) .. Output 1. Grab Book.2 (highest book) 2. Sit Sofa.1 (highest book) 3. Place Book.2 on Bookshelf.1	Input Given a description of a task and objects with coordinates, execute the task using the objects and only the following actions: 1. Grab object 2. Sit object 3. Place object on object. Note that the coordinate y corresponds to the height. Do not use any other action. Description: Get all the plates and put them on the table, then sit on the chair and relax. Objects: 1. Plate.3 (x: 3.0, y: 0.7, z:0.8) 2. Plate.1 (x: 3.1, y: 0.7, z:0.9) 3. Table.1 (x: 13.1, y: 1.1, z:0.4) 4. Chair.1 (x: 11.0, y: 0.5, z:8.0) 5. Sofa.1 (x: 4.0, y: 0.5, z:8.0).. ... Output 1. Grab Plate.3 2. Place Plate.3 on Table.1 3. Grab Plate.1 4. Place Plate.1 on Table.1 5. Sit on Chair.1
--	--

Figure 6.8: **Action labels from language descriptions.** Examples of action labels generated from language descriptions using a pre-trained language model. The model is able to understand geometric constraints (tallest book) and generate instructions coherent with the language description.

following a similar process to that of the navigation map. We quantize the height of the scene between the floor and the ceiling into bins. We iterate over each height bin, going from the floor to the ceiling, and at every height bin, we build a map like the one in Fig. 6.4 using the points whose height falls into the bin range. We then aggregate this map over all the bins, by assigning to each area the value of the tallest bin that covered it.

Keypoints from actions: The first approach takes as input a sequence of actions. Each action is composed by a verb and an object instance, over which the action will be executed. We support three kinds of actions: *Sit*, *Grab* and *Place*. An example action sequence would be: *Grab controller.1*, *Grab chips.2*, *Sit sofa.1*. The numbers after the object are used to disambiguate between multiple instances of the same object class. For every action, we generate keypoints corresponding to the feet positions, the hand and the root, depending on the action label. If the action is *Grab*, we select the centroid of the points corresponding to the action object as the hand position. For the *Sit* action, we select the centroid as well, but update the height according to the height map calculated from the scene. For *Place*, we use the same method as *Sit*, but setting the hand position, instead of the root joint. Finally, we select the position of the feet, by sampling points in a circle around the selected object and making sure that they fall in a navigable area.

Keypoints from language: Given a method to generate keypoints from actions, we can use a pre-trained language model to generate keypoints from language descriptions, with a reasonable robustness. For that, we use a pre-trained large language model [Brown et al., 2020], and prompt it to generate sequences of actions from a

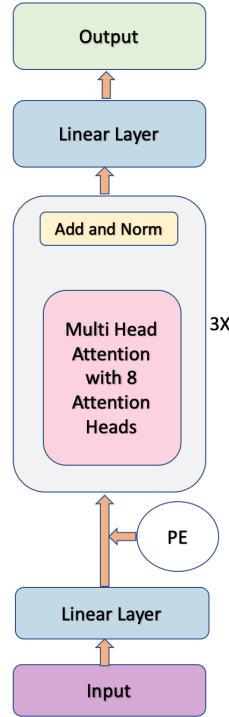


Figure 6.9: Transformer architecture for WalkNet and TransNet.

restricted set of actions and objects, obtained from the scene parsing. Fig. 6.8 shows the prompts we use.

After running the language model, we parse the output into sequences of actions and object labels, using the previously described method to convert those into key-points. We find that the language model is able to generate sequences of actions that match the description, and even perform some degree of spatial reasoning. An overview of the full system can be seen in Fig. 6.7.

6.3.2 Architecture Details

We present the architectural details of the transformers used in our method. As illustrated in Fig. 6.9, both *TransNet* and *WalkNet* are BERT style full-attention transformers where each node in the output receives information from all nodes in the input. First we embed the input vector to a 512 dimensional vector using a linear layer. To this embedded vector, we add a standard positional encoding. This is used as input to 3 standard attention layers stacked together, each of which comprises of 8 attention heads. The output of these layers is de-embedded using a linear layer and the output of this linear layers acts as the final output of the Transformer.

6.3.3 IK Details

To stabilize the IK step, we use several heuristics. Irrespective of the keypoints specified, we ensure the heel and the toes remain in the same plane.

When the user provides the location of the root joint, we ensure that the position of the head joint is at a height of 0.5 m above the location of the root joint and the two hands are placed at the same height as the root.

When the user specifies the location of either one or two hands, we ensure that the root joint is placed at a height of 0.9 m above the ground.

All these heuristics allow us to obtain more natural looking motion-anchors.

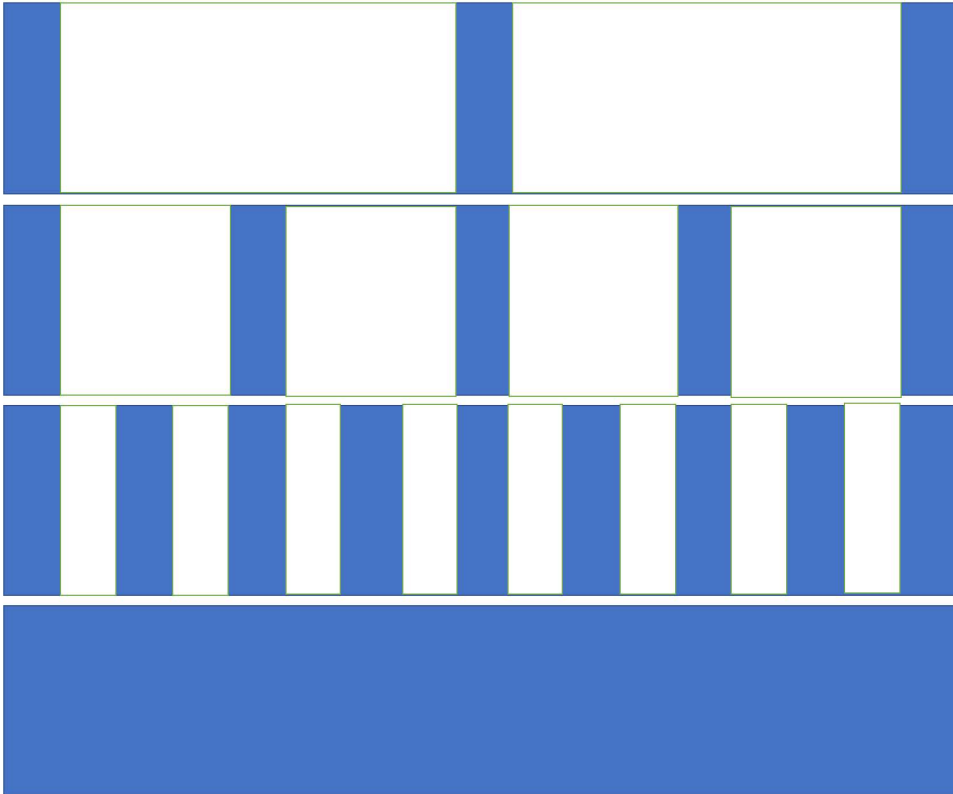


Figure 6.10: **Progressive growing of the mask.** In the first epoch, *TransNet* receives the whole sequence as input. In the second, alternative frames are dropped as illustrated by the white stripes. In the subsequent epoch, the masked out region is increased by one more frame. This goes on until all the middle $\frac{M}{2} - 1$ frames are masked out.

6.3.4 Progressive growing of the Mask for TransNet

We find that, if *TransNet* is trained to directly infill all the missing motion frames, it produces significant foot-skating. Please see the main paper for this experiment. We instead progressively grow the number of frames needed to be infilled by *TransNet*. In the first epoch of training, we set the mask to zero, i.e *TransNet* learns an identity

mapping. In the second epoch we drop one frame after each other. In each epoch the number of dropped frames is doubled until all $\frac{M}{2} - 1$ frames are masked out. For an illustration, please see Fig. 6.10

6.3.5 Different values of M used in *TransNet*

We train three variants of *TransNet* with 3 values of $M = 120, 150, 180$. To evaluate the best value of M , we place the same seed sequence in 20 locations in the canonical coordinate and place 20 random static pose at the origin of the coordinate frame - one for each seed sequence. Then we synthesize the missing poses using the three variants of M . We compute the average foot-skate metric the synthesized motion sequences. For $M=120$, the value is 1.04 cm/frame. For $M=150$, the value is 1.09 cm/frame. For $M=180$, the value is 1.12 cm/frame.

6.4 EXPERIMENTS

In this section we present implementation details of our method. Next, we compare our approach with existing methods. Our experiments show that we clearly outperform existing baselines. Next, we ablate our design choices and finally present qualitative results of our method.

6.4.1 Implementation Details

WalkNet and *TransNet* are BERT [Devlin et al., 2019] style full-attention transformers. Both consist of 3 attention layers - each composed of 8 attention heads. We use an embedding size of 512 for both transformers. For more details please see the supplementary material. For training both transformers, we set the learning rate to 10^{-5} . Both networks are trained using an L2 loss. We set $M = 120$ and $K = 30$. We experimented with three different values of M and found that $M = 120$ produces the least foot-skating. Please see the supplementary material for this experiments.

6.4.2 Datasets

Motion Data: To train *TransNet* and *Walknet* we use the large mocap dataset **AMASS** [Mahmood et al., 2019].

AMASS Data Normalization: AMASS contains motion sequences of variable framerate. We first resample all motion sequences to a fixed frame-rate of 30fps. Our model is person-specific, i.e we assume the shape parameters remain the same throughout motion synthesis. AMASS data is not person-specific. So we normalize every motion sequence by finding the difference between the height of the person in a particular sequence and that of constant-shape person used for motion synthesis.

Walk Data Augmentation: We use person specific walking data from the AMASS

	Ours	SAMP	Ours	GAMMA	Ours	Wang et al.
Motion Realistic (%) $\uparrow$	71.8	28.2	95.6	4.4	100	0
Motion best in Scene (%) $\uparrow$	76.8	23.2	100	0	100	0

Table 6.1: **Perceptual study comparison with baselines.** Comparisons between our method and existing baselines using a perceptual study.

	Language	Manual	WalkNet	MoGlow	TransNet	NeMF
Foot Skate (cm/f) $\downarrow$	0.93	0.92	0.91	1.88	1.1	1.54
User Study (%) $\uparrow$	53.8	46.2	75.7	24.3	66.8	33.2

Table 6.2: **Component-wise analysis.** We compare our method with different baselines across three design components: using language based or manually specified keypoints, the walking motion and the transition motion.

motion data to train WalkNet. To find walks in the AMASS data, we employ a very simple heuristic. We mark a frame as a walking frame if it satisfies these two properties 1) the root height is above a fixed threshold 2) the planar velocity (measured as the forward difference of the planar root positions) is above a very small threshold. This ensures that only frames where the person is not stationary and not bent are selected. We augment the walk motions in our dataset by twisting the trajectories around so that the same motion starts off at different locations in the canonical coordinate frame.

We assume a motion sequence in the canonical coordinate frame of N frames. We represent it as $\{(\mathbf{r}_i, \psi_i)\}_{i=1}^N$ where $\mathbf{r}$ represents the root position and ψ represents the root orientation. The last frame is assumed to be oriented towards the canonical axis of orientation.

We augment the motion sequence by rotating the velocity and orientation of each frame. We define the augmented motion sequence as $\{(\mathbf{r}_i^A, \psi_i^A)\}_{i=1}^N$. The algorithm for data augmentation is described in Alg. 2

We first define a function R that maps a scalar angle θ to the planar matrix that rotates a vector about the z-axis by the angle θ

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.6)$$

	Ours	SAMP	GAMMA	Wang et al.
Foot-skate ↓	0.91	1.34	0.94	4.53

Table 6.3: **Foot-skate metric comparison with baselines.** Comparisons between our method and existing baselines using the foot-skate metric.

Algorithm 2: Motion data augmentation.

Input: δ
 $\mathbf{r}_N^A = \mathbf{r}_N$;
 $\psi_N^A = \psi_N$;
for $i = N - 1 \dots 1$ **do**
 $\delta_i = \delta \times (N - i)$;
 $\mathbf{R}_i = R(\delta_i)$;
 $\mathbf{v}_i = \mathbf{r}_i - \mathbf{r}_{i+1}$;
 $\mathbf{r}_i^A = \mathbf{R}_i \mathbf{v}_i + \mathbf{r}_{i+1}^A$;
 $\psi_i^A = \mathbf{R}_i(\psi_i)$;
end for

Scene Datasets: We demonstrate that our method is able to generate realistic human motion in scenes from **Matterport3D**, **HPS**, **Replica** and **ScanNet** datasets. All these datasets have been reconstructed using RGB-D scanners or LIDAR scanners and contain scans with sizes ranging from 20 m^2 to 1000 m^2 . While Replica, Matterport scenes contain perfect geometry, ScanNet scenes do not. Our method is able to generalize across all these scenes.

6.4.3 Evaluation Metrics

We compare our method with existing baselines using perceptual studies and a foot skate metric. Additionally, we ablate various components of our method with the same foot skate metric.

Perceptual Study: We synthesize two motion sequences - one using our method and another using a baseline method and show the two synthesized sequences to participants in our perceptual study. The participant is asked to answer “Which motion looks most realistic?” and “Which motion satisfies scene constraints best?”. The study is conducted in such a manner that the participant is forced to choose one of two motions in front of them.

Foot Skating (FS): The foot-skate metric measures how much foot-skate occurs during a synthesized motion measured in cm/frame. For N frames, it is defined as:

$$s = \sum_{p=1}^N [v_p (2 - 2^{\frac{h_p}{H}}) \mathbf{1}_{h_p < H}]$$

where h_p is the height of the vertex and v_p is the velocity of a foot vertex on the right toe in frame p and $H = 2.5$ cm

6.4.4 Comparison with Baselines

As aforementioned, no method addresses the task of continual motion synthesis in arbitrary 3D scenes. For completeness we do our best to compare our approach with three existing methods: SAMP [Hassan et al., 2021a], GAMMA [Zhang and Tang, 2022], Wang et al. [Wang et al., 2022] which all generate animator guided motion by navigating A* paths in 3D scenes. Though, these methods use different forms of animator guidance - such as action labels, we modify them by incorporating the KP information used by our method. Note that except GAMMA, none of these baselines can be deployed in arbitrary 3D scenes without significant modifications, as described below.

SAMP: SAMP is written entirely in Unity and can only synthesize sitting and lying actions in synthetic scenes. Unlike SAMP our method requires no manual action annotation. The object of interaction and the action to perform are the animator guidance provided as input to SAMP. SAMP synthesizes motion by explicitly conditioning on the geometry of the object of interaction, and by navigating A* paths. For comparison with SAMP, we represent the object of interaction in one of our test scenes with a synthetic object in Unity. Using KPs, we represent the orientation of action in our test scene and use this orientation to port A* paths used in our test 3D scene into Unity and run the publicly available code of SAMP.

Since most of our test scenes do not have perfect segmentations available, we represent the object of interaction in a 3D test scene by one of the segmented objects available in the Unity interface of SAMP. We place this object at the origin of the Unity coordinate frame and orient it along the positive y axis. The keypoint that defines the root location in our 3D scene is used as the sitting location for the Unity object of interaction. Using keypoint information used by our method, we define the orientation of the object of interaction in the test 3D scene. Using this orientation, we find a rigid coordinate transformation between the test 3D scene and the Unity object of interaction. This allows us to move the A* waypoints defined in the scene coordinate frame into the unity coordinate frame. There we use the interface provided by SAMP to synthesize motion along the trajectory.

Once all motion is synthesized, we move it back into the 3D scene coordinate frame. We find that for long trajectories, SAMP produces a lot of foot-skating.

Note that SAMP cannot synthesize chained actions nor can it be deployed in arbitrary 3D scenes. For instance it cannot sit on stairs nor can it perform a grabbing action near a bookshelf. The comparison is included for completeness as SAMP also navigates A* paths.

Wang et al.: We run the pre-trained code of Wang et al. on scenes from HPS, Replica and Matterport Datasets. Instead of using action labels to generate anchor poses as done in the original paper, we replace this step with the motion anchors generated using our inverse kinematics step. Since Wang et al. [Wang et al., 2022] is trained using the PROX dataset and synthesizes navigational motion across A* paths by explicitly conditioning on scene geometry, it does not generalize at all to 3D scenes beyond these datasets.

Wang et al. assume the availability of scene SDFs which are unavailable for HPS,

Scannet and large Matterport scenes. These scene SDFs are required by Wang et al. to place the motion anchors in the 3D scene. Instead of using this step, we use the motion anchors generated by our IK step. Then, using these motion anchors we synthesize motion from the start to goal location using code provided by Wang et al. Since their method is trained only on PROX scenes, it does not generalize to large scenes from the HPS and Matterport datasets.

GAMMA: GAMMA only navigates 3D scenes and is unable to synthesize human-scene interaction. Similar to the navigation part of our method, it uses the start and end of a path as animator guidance. For the purpose of this comparison, we generate a set of paths in 3D scenes using A* and synthesize walking motion along this path using GAMMA and our method. GAMMA is unable to follow the exact waypoints of the path and as such produces significant interpenetrations with the 3D scene.

For visualizations of motion synthesized by these baselines, please see the supplementary video. We synthesize 5 motion sequences of a total duration of 300 seconds using each method in 5 different scenes for our perceptual study. In Tab. 6.1, we report the results of our perceptual study with 50 participants (see Sec. 6.4.3). Each column corresponds to the percentage of users who choose the method corresponding to the column heading. Our results are preferred by a vast majority of the participants. In Tab. 6.3, we report the numbers corresponding to foot skate metric.

6.4.5 Ablation Studies

Can *TransNet* be replaced with other inbetweeners? We compare *TransNet* with the SoTA inbetweening method NeMF [He et al., 2022] for the task of transitioning in and out of actions. For our task of infilling $\frac{M}{2} - 1$ frames in the canonical coordinate frame, *TransNet* produces more natural motion and less foot skating. We hypothesize that this occurs as NeMF is a general purpose inbetweener that can infill an arbitrary number of frames, whereas *TransNet* is a motion inbetweenner custom designed for the purpose of infilling $\frac{M}{2} - 1$ motion frames in the canonical coordinate frame. We conduct a new user study with 36 participants, asking users to rate the naturalness of 20 motion sequences by NeMF and *TransNet*. Results are reported in Tab 6.2.

We present two motion sequences to participants in our user study and label the motion sequences as ‘A’ and ‘B’. The interface is shown in Fig. 6.11. We conduct the user study such that a participant is always forced to pick one of the two motion sequences in front of them.

Can *WalkNet* be replaced with other path following methods? We provide comparisons with SAMP, Wang et al, and GAMMA which all navigate A* paths. As our experiments illustrate, our method outperforms these existing methods for navigation. For further completeness, we trained the SoTA walking method, MoGlow [Henter et al., 2020], on our walking data. When deployed on 150-200 meter long A* paths, it produces significant foot-skating after about 30 secs. We hypothesize that this occurs because MoGlow synthesizes motion in an egocentric coordinate frame, and hence the control signal provided by A* changes rapidly which leads MoGlow to synthesize motion with significant drift. We compare our method to MoGlow on



Figure 6.11: User study interface.

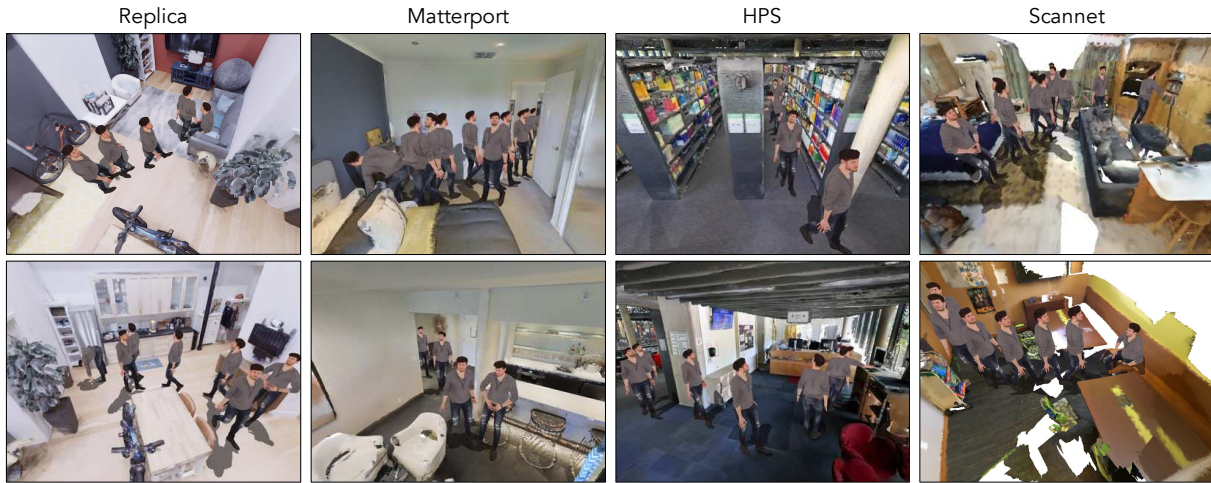


Figure 6.12: **Generalization across different scenes.** Our method generates motion that generalizes across different scenes. Here we show motion generation in scenes from 4 different datasets: Replica [Straub et al., 2019], Matterport [Chang et al., 2017], HPS [Guzov* et al., 2021] and Scannet [Dai et al., 2017].



Figure 6.13: **Diverse and controllable motion generation.** The keypoint representation allows us to generate diverse and highly controllable motion. We show here examples of different grabbing, sitting and newly defined motions.

these paths using a user study with 36 participants in Tab. 6.2, where our approach outperforms MoGlow.

How well does language based keypoint placement work? In this experiment, we

Architecture	Foot-skate ↓	Success Rate % ↑
Transformer	0.91	100
MoE	3.41	58.8
LSTM	6.89	10.22

Table 6.4: **Transformer architecture ablation.** Ablation study to evaluate the importance of the transformer architecture.

compare motion synthesized using manual keypoint placement with language based keypoint placement. We synthesize 5 motion sequences using keypoints generated by these two approaches and compare the synthesized sequences using a user study with 36 participants. When used for motion synthesis, these KPs produce similar quality as manual KP placement (Tab. 6.2).

How long does it take for a user to provide keypoints manually? We develop a user interface which allows users to navigate 3D scenes and to click on locations of interaction. We instruct 7 participants how to navigate 3D scenes with our user interface. On average it takes 245 seconds for users to learn the interface. We then ask each user to provide 5 sets of 3 action keypoints (the location of the root and the two feet or the location of one hand and two feet) for a total of 15 keypoints per scene in 5 different 3D scenes. On average it takes 125 seconds to select these points per scene.

How important is Origin Canonicalization? In this experiment, we define an Egocentric WalkNet, where we place a seed motion at the origin and orient it along the x axis. A target is provided as additional input to the EgoCentric Transformer. We randomly sample 20 locations in the canonical coordinate frame and provide them as input to the EgoCentric Transformer. Additionally, we synthesize motion from these 20 locations using our Origin canonicalized Transformer. *WalkNet* always reaches the origin while the EgoCentric Transformer fails to reach the target 69.2 percent of the time. Additionally, the EgoCentric Transformer produces 3.2 cm/frame foot-skating compared to ours which only produces 0.92 cm/frame.

How important is the transformer architecture? In this experiment we evaluate WalkNet - specifically we evaluate how essential is the transformer architecture for WalkNet. We randomly place seed motion sequences in the *goal-centric canonical coordinate frame* and synthesize motion using LSTMs, an MoE architecture and a transformer. In Tab. 6.4, we report the average foot-skate metric for these motion sequences and the success rate of convergence to origin.

How important is progressive growing of mask? In this experiment we evaluate TransNet. We train *TransNet* with and without progressively growing the mask. Then we randomly place seed motion sequences and random poses at the origin in the goal-centric canonical coordinate frame and synthesize motion with the two variants of *TransNet*. The various metrics are reported in Tab. 6.5. Without progressively growing the mask, *TransNet* produces significantly larger foot-skating.

Configuration	Foot-skate ↓
With Progressive Growing	0.98
Without Progressive Growing	1.46

Table 6.5: **Progressive mask growing ablation.** Ablation study to evaluate *TransNet* with and without progressive mask growing.

6.4.6 Qualitative Results

Please watch the supp. video for qualitative evaluation. In Fig. 6.12, we demonstrate examples of motion generated in scenes from 4 different datasets: Replica [Straub et al., 2019], Matterport [Chang et al., 2017], HPS [Guzov* et al., 2021] and ScanNet [Dai et al., 2017]. Moreover, representing the motion as Action Keypoints allows us to have high control and diversity over the generated motions. In Fig. 6.13 we show how this representation allows us to sit or pick objects at different heights (left column), or generate actions such as grabbing with two hands or stretching.

6.5 CONCLUSION

We presented the first method to synthesize continual human motion in scenes from the HPS, Matterport, ScanNet, and Replica datasets. Our core contribution is a novel method for long-range motion synthesis via iterative canonicalization and the use of keypoints to decouple scene reasoning from motion synthesis, and provide a flexible interface to synthesize motion. We demonstrated that our method works better than existing solutions that generate motion in 3D scenes.

While our approach presents an important step towards long-range motion synthesis in 3D scenes, it also has limitations: It assumes a horizontal floor and thus does not support scenes with uneven floors. It also assumes valid keypoint configurations: if the keypoints provided by the user do not conform to a valid pose, the pose produced by the IK step will not look realistic, producing unnatural motion. In the future we hope to remove this limitation by reducing the number of required keypoint inputs. We hope that the proposed approach drives new research towards continual human motion in arbitrary 3D scenes.

Part II

Humans in Scenes with Neural Scene Representations

In Part I, we addressed human appearance, capture, and motion synthesis using classical mesh and point cloud representations. While these representations provide the geometric machinery for robust animation pipelines, their rendering quality is fundamentally limited by the fidelity of the underlying surface approximation.

In this second part, we depart from classical representations entirely and adopt 3D Gaussian Splatting (3DGS) as the sole 3D representation for human–scene interaction. This shift brings photorealistic rendering at real-time rates, but introduces fundamental challenges: Gaussian fields provide none of the geometric quantities that classical pipelines rely on—no surfaces to walk on, no signed distance fields for collision, no navigable meshes for path planning, and no surface normals for shadow casting. Can these missing geometric capabilities be recovered entirely from Gaussian fields? The four chapters in this part demonstrate that they can. Ch. 7 reconstructs and animates separate human and object models in 3DGS scenes. Ch. 8 introduces the first method to synthesize human–scene interactions using 3DGS as the sole representation. Ch. 9 addresses shadow casting entirely in Gaussian space. Finally, Ch. 10 scales avatar reconstruction to in-the-wild YouTube video, unlocking internet-scale footage as a source of 3D human assets.

INTERACTSPLAT: GAUSSIAN SPLATTING FOR HUMAN–OBJECT SCENE ANIMATION

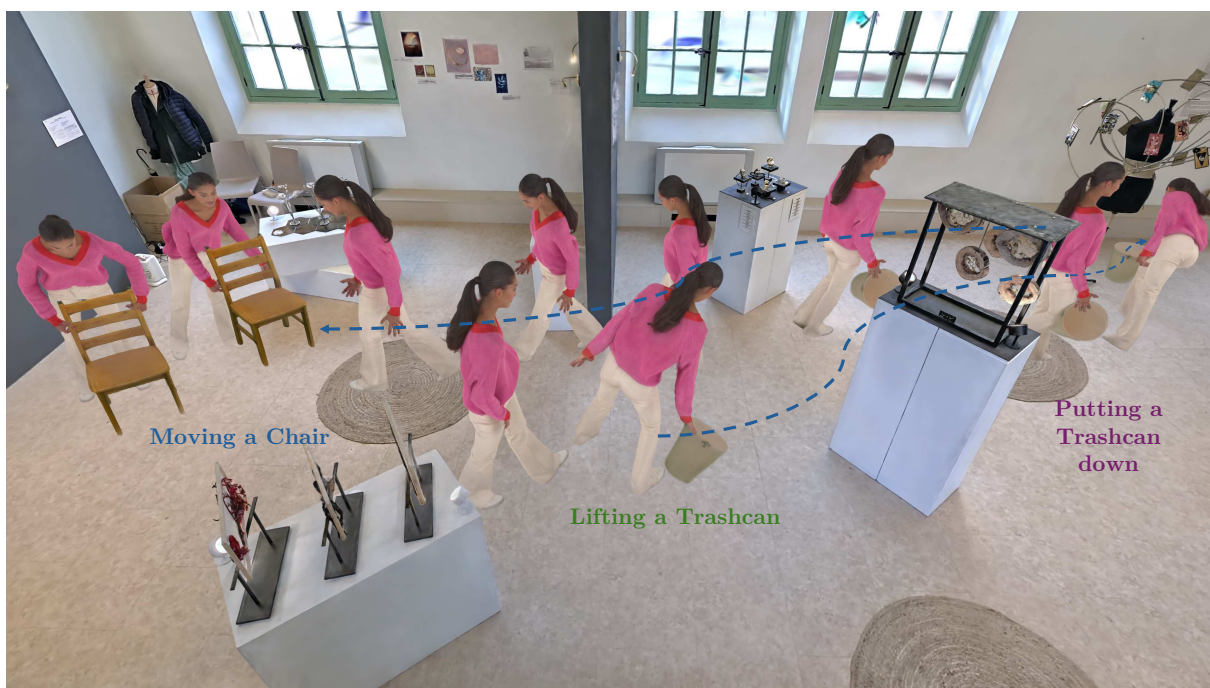


Figure 7.1: **InteractSplat**. Given multi-view RGB of human-object interactions, we reconstruct separate controllable 3DGS models of humans and objects and animate them in diverse 3D scenes—synthesizing long-horizon sequences where avatars navigate scenes, pick up objects, carry them, and set them down at new locations.

This chapter opens Part II of the thesis, which replaces the classical mesh and point cloud representations of Part I with 3D Gaussian Splatting. A long-standing goal of virtual reality is to create photorealistic renderings of humans moving around and interacting naturally with dynamic objects in large environments. To this end, we present InteractSplat—a method based on 3DGS for generating photorealistic, controllable renderings of humans performing chained interactions with multiple objects in large 3DGS scenes. This requires solving multiple problems in reconstruction, tracking, and animation of humans and objects. To address these problems, we make several contributions: we introduce a novel coarse-to-fine formulation for object reconstruction and tracking that learns object Gaussians on an underlying 2D surface manifold rather than in 3D volume, yielding sharper, fine-grained object details under

occlusion. We introduce dynamic 2D maps of 3DGS scenes that allow avatars to navigate scenes with dynamic objects. We further propose a contact constraint in Gaussian space that regularizes human-object relations and enables natural, physically plausible animation. Across three benchmarks—BEHAVE, NeuralDome, and DNA-Rendering—InteractSplat achieves high-quality reconstructions under occlusion and supports controllable synthesis of novel human-object interactions in diverse 3DGS scenes, including avatars lifting objects, moving furniture, and navigating dynamic 3D environments. Video results are available at <https://miraymen.github.io/interactsplat>.

7.1 INTRODUCTION

Photorealistic animatable human avatars have seen remarkable progress with Gaussian Splatting [Kerbl et al., 2023], enabling methods [Moreau et al., 2024, Wen et al., 2024, Kocabas et al., 2024, Qian et al., 2024, Li et al., 2024d, Pang et al., 2024] that reconstruct high-quality avatars driven by controllable pose signals. However, these methods reconstruct and animate humans *in isolation*—without objects, without occlusions, and in empty space. In reality, human motion rarely occurs in isolation: the environment constrains our motions and provides affordances, and we constantly interact with objects around us.

Two largely disconnected research directions address pieces of this problem. Photorealistic avatar methods reconstruct high-quality animatable humans [Moreau et al., 2024, Kocabas et al., 2024, Li et al., 2024d, Pang et al., 2024]—but in isolation, without objects. Conversely, human-object and human-scene interaction methods [Guzov* et al., 2021, Zhang et al., 2022, Hassan et al., 2019, Zhu et al., 2024a, Hassan et al., 2021a] model realistic motion in environments—but produce only skeletons or meshes, not photorealistic renderings. Recent photorealistic reconstruction methods either assume static objects [Kocabas et al., 2024, Xue et al., 2024], cannot synthesize novel interactions [Sun et al., 2021, Jiang et al., 2022c], or are restricted to hand-held objects or limited motions [Zhan et al., 2025b, Liu et al., 2023, Lee et al., 2025]. *No existing method bridges these two worlds: reconstructing separate photorealistic, controllable human and object models from their interactions, and animating them together in 3D scenes.*

We present InteractSplat, an end-to-end pipeline for photorealistic human-object interaction in 3D Gaussian scenes (Fig. 7.1). Starting from multi-view RGB of a human interacting with an object, we reconstruct separate controllable models of both, and animate them in diverse 3DGS environments—synthesizing long-horizon sequences where avatars navigate scenes, pick up objects, carry them, and set them down at new locations. Achieving this requires overcoming challenges on both fronts.

Reconstruction under mutual occlusion. Human-object interactions create occlusions where each entity hides parts of the other, causing existing single-entity reconstruction pipelines to fail. Furthermore, unlike for humans where SMPL [Loper et al., 2015, Pavlakos et al., 2019] provides geometry and tracking, no parametric template exists for objects—and objects are often small and heavily occluded, leading to degenerate solutions. We propose a composition and occlusion guided loss that avoids penalizing hidden regions, and a *coarse-to-fine* object reconstruction: learning a

template in feature space, tracking it across frames with photometric alignment, and using *canonical* Gaussian maps for high-fidelity textures (Secs. 7.2.1.1–7.2.1.5).

Animation in dynamic environments. Captured human–object interactions are typically short, single-interaction clips—insufficient for realistic long-horizon scenarios that involve multiple interaction types, locomotion, and collision avoidance. Moreover, objects move during interactions, changing the scene layout and invalidating static obstacle maps. We propose 2D Dynamic Maps—orthographic projections updated after every interaction—enabling collision-free path planning in changing environments. To synthesize long sequences from short clips, we chain retrieved interactions with RL-based locomotion and motion inbetweening (Secs. 7.2.2.1–7.2.2.2).

Contacts in Gaussian space. Naive composition of human, object, and scene Gaussians produces penetrations and floating artifacts. Unlike meshes, Gaussians lack explicit surfaces to snap to. We introduce contact constraints via soft nearest-neighbor distances that enforce plausible human–object, human–scene, and object–scene contacts directly in Gaussian space (Sec. 7.2.2.4).

We evaluate reconstruction on multiple datasets: DNA-Rendering [Cheng et al., 2023], NeuralDome [Zhang et al., 2023], and BEHAVE [Bhatnagar et al., 2022], showing improvements over existing methods. For animation, we demonstrate chained motions—avatars lifting objects and moving furniture—in large 3D scenes [SuperSplat] (Fig. 7.1). To our knowledge, such demonstrations have not been previously achieved. Our contributions are:

1. Joint reconstruction of humans and objects under mutual occlusion, via a composition and occlusion guided loss and a *coarse-to-fine* object reconstruction using *canonical* Gaussian maps for high-fidelity textures of dynamic objects.
2. Dynamic 2D maps of 3DGS scenes with changing components, enabling collision-free motion synthesis in dynamic environments.
3. Contact constraints in Gaussian space to ensure proper human–object, human–scene, and object–scene contacts when animating novel interactions.
4. An end-to-end pipeline for controllable generation of photorealistic humans interacting with dynamic objects across diverse 3D scenes.

7.2 METHOD

Given multi-view RGB images of humans interacting with objects, we first reconstruct separate animatable and controllable 3DGS models of both humans and objects (Sec. 7.2.1). We then animate these models in diverse 3D scenes, synthesizing collision-free navigation and human–object interactions (Sec. 7.2.2).

7.2.1 Reconstruction

Our method receives a set of multi-view RGB images of humans interacting with objects, captured from N_c cameras at T temporal frames, $\{\mathbf{I}_t^c\}_{c=1, t=1}^{N_c, T}$, along with

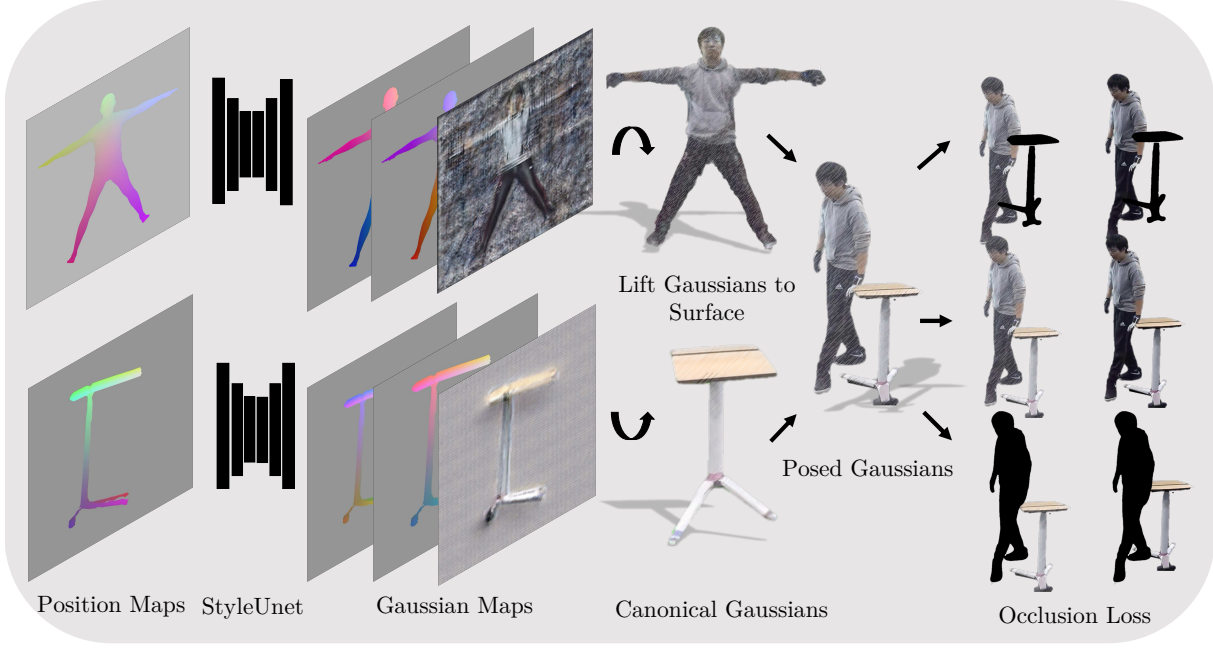


Figure 7.2: **Reconstruction pipeline overview.** Using position maps as input, we learn Gaussian parameters for Gaussians anchored to canonical human and object templates. Gaussian properties are predicted as 2D maps structured according to a UV unwrapping of the canonical templates. Each pixel corresponds to a single Gaussian. The canonical Gaussians are then posed using LBS for the human and a rigid transform for the object. We guide reconstruction with an occlusion-aware loss.

estimates of camera parameters $\{\gamma_{c,t}\}_{c=1,t=1}^{N_c,T}$ and human pose $\{\theta_t\}_{t=1}^T$. We aim to learn a deformable function $f(\theta, \phi; \gamma)$ that maps new human poses θ and object poses ϕ to RGB images rendered from novel viewpoints γ . We use a feature-based representation to construct coarse object templates (Sec. 7.2.1.1) and track them across temporal frames (Sec. 7.2.1.2). Using the estimated object template and the SMPL mesh, we define canonical Gaussian maps (Sec. 7.2.1.3) for two sets of Gaussians $\mathcal{G}^O$ and $\mathcal{G}^H$, which are deformed to posed space (Sec. 7.2.1.4) and learned using an occlusion guided photometric loss (Sec. 7.2.1.5).

7.2.1.1 Canonical Object Template

To model object motion, we first learn a coarse canonical 3D object template. Direct optimization of 3DGS [Kerbl et al., 2023] fails under naturally occurring human-object occlusions (Tab. 7.4), so we adopt triplane features [Mihajlovic et al., 2024, Chan et al., 2022] which act as a smoothness prior enabling convergence under occlusion. Specifically, we learn triplane features $\mathbf{F} \in \mathbb{R}^{3 \times H_p \times W_p \times l}$, where $H_p \times W_p$ is the spatial resolution of each plane and l the feature dimension. Each object Gaussian center μ_p is projected onto the three planes to obtain features $\mathbf{f}_p = \pi(\mu_p; \mathbf{F}) \in \mathbb{R}^{3l}$, where π denotes triplane projection via bilinear interpolation. Small MLPs then decode these features into Gaussian properties (color, covariance, scale, opacity). Crucially,



Figure 7.3: Occlusion patterns due to human presence.

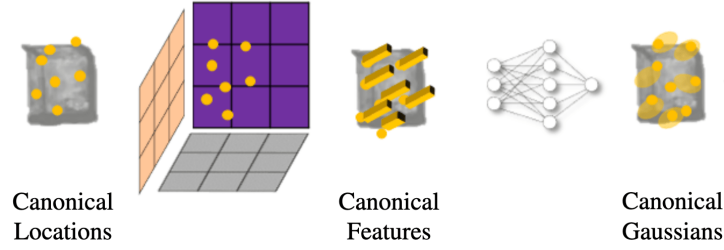


Figure 7.4: Learning Gaussians with features.

rather than optimizing Gaussian properties directly, we learn the feature grids and Gaussian positions μ_p , from which all other properties are derived. This indirect parameterization provides the regularization needed to reconstruct objects under heavy occlusion.

Why features? The original 3DGS algorithm is designed to work with images of complete objects in static scenes. We observe that 3DGS does not converge when used to reconstruct objects under occlusion using only object-visible pixels. This motivates using features, which act as a regularizer and a smoothness prior for reconstruction under occlusion—an observation also noted in prior work [Mihajlovic et al., 2024, Kocabas et al., 2024]. In our setting, even under dense camera setups, the object is occluded from a significant number of cameras. Fig. 7.3 shows observed pixels of one object across different cameras at a single timestep, illustrating typical occlusion patterns. Fig. 7.4 provides further illustration of how features interact with 3D Gaussian object locations. Alg. 3 in Sec. 7.3 summarizes the template learning procedure in pseudocode.

We select the frame t^* with minimal occlusion (*i.e.*, highest count of object pixels across cameras) and optimize

$$\mathcal{G}_*^O = \arg \min_{\mathcal{G}^O} \sum_{c=1}^{N_c} \|\mathcal{R}(\mathcal{G}^O; \gamma_{c,t^*}) - \mathbf{O}_{t^*}^c \cdot \mathbf{I}_{t^*}^c\|_2 ,$$

where $\mathcal{R}(\cdot; \gamma_{c,t})$ is the 3DGS rasterizer, $\mathbf{O}_t^c$ the object mask, and $\mathbf{I}_t^c$ the RGB image. This yields a coarse canonical template $\mathcal{G}_*^O$.

7.2.1.2 Object Tracking

Once we have the canonical template $\mathcal{G}_*^O$, we track it across timesteps by optimizing a rigid 6D pose ϕ_t :

$$\arg \min_{\phi_t} \sum_{c=1}^{N_c} \|\mathcal{R}(\mathcal{T}(\mathcal{G}_*^O; \phi_t); \gamma_{c,t}) - \mathbf{O}_t^c \cdot \mathbf{I}_t^c\|_2. \quad (7.1)$$

Here $\mathcal{T}(\mathcal{G}^O; \phi)$ is a rigid transformation on canonical object Gaussians. For every Gaussian, we transform its center μ and covariance Σ :

$$\mu' = \mathbf{R}_\phi \mu + \mathbf{t}_\phi, \quad \Sigma' = \mathbf{R}_\phi \Sigma \mathbf{R}_\phi^\top, \quad (7.2)$$

where $\mathbf{R}_\phi$ and $\mathbf{t}_\phi$ are the rotation and translation components of the 6D object pose ϕ . To ensure convergence, ϕ_t is initialized from ϕ_{t-1} .

7.2.1.3 Gaussian Maps

We use the SMPL body model to define a canonical human Gaussian template $\mathcal{G}_*^H$ by densely sampling points on the SMPL mesh. Given the canonical templates $\mathcal{G}_*^H$ and $\mathcal{G}_*^O$, we learn their Gaussian properties in image space using two StyleUNet [Wang et al., 2023] models S_H and S_O . This 2D formulation leverages the inductive biases of CNNs [Pang et al., 2024, Li et al., 2024d, Hu et al., 2024b] while maintaining a one-to-one correspondence between pixels and 3D Gaussians.

Human maps (pose-dependent). Following [Li et al., 2024d], we create pose-dependent position maps by projecting posed SMPL vertices onto front and back UV maps. S_H takes these maps as input (Fig. 7.2) and predicts per-pixel Gaussian properties as offsets from the canonical human template.

Canonical object maps. For the object, we project the coarse template $\mathcal{G}_*^O$ from front and back views to establish a one-to-one mapping between 2D pixels and 3D Gaussians (Fig. 7.5). We construct *canonical* Gaussian maps by storing the 3D location of each template Gaussian at its projected pixel position. Unlike for humans, these maps do not vary with time; the object is rigid and the same canonical maps are reused at every timestep. This design yields high-fidelity textures for dynamic objects under occlusion (Fig. 7.14). S_O processes these canonical maps and predicts offsets of Gaussian properties from the coarse template.

7.2.1.4 Deformation

StyleUNets (Sec. 7.2.1.3) predict offsets from canonical templates to produce updated Gaussians $\mathcal{G}^H$ and $\mathcal{G}^O$; additionally, S_H predicts per-Gaussian skinning weights $\mathbf{w}_{k,j}^H$. These are then deformed into posed space.

For humans, Linear Blend Skinning (LBS) (Fig. 7.2) is applied using the learned skinning weights. For every canonical human Gaussian, we compute:

$$\mathbf{R}_{k,t}^H = \sum_{j=1}^J \mathbf{w}_{k,j}^H \mathbf{R}_j^{\text{SMPL}}(\theta_t), \quad \mu'_{k,t} = \mathbf{R}_{k,t}^H \mu_k + \mathbf{t}_{k,t}^H, \quad \Sigma'_{k,t} = \mathbf{R}_{k,t}^H \Sigma_k (\mathbf{R}_{k,t}^H)^\top, \quad (7.3)$$

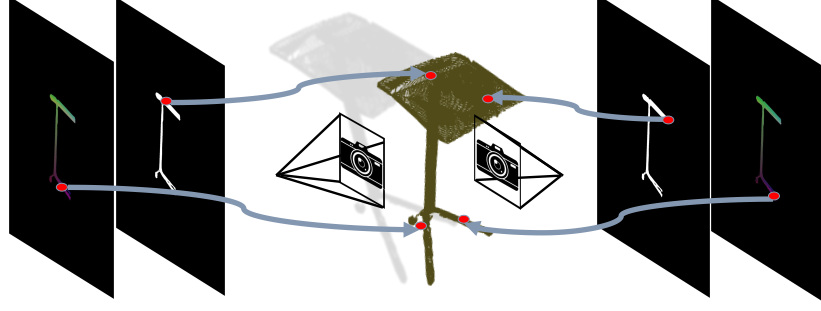


Figure 7.5: **Canonical object Gaussian maps.** Each pixel corresponds to a canonical Gaussian. The maps are time-invariant, yielding high-fidelity textures. Arrows indicate pixel-to-Gaussian correspondence.

where k indexes Gaussians, J is the number of SMPL skeleton joints, $\mathbf{R}_j^{\text{SMPL}}(\boldsymbol{\theta}_t)$ is the j -th joint rotation matrix under pose $\boldsymbol{\theta}_t$, and $\mathbf{t}_{k,t}^H$ is the translational component of the per-Gaussian LBS transform. We denote the full LBS deformation as $\mathcal{T}(\mathcal{G}^H; \boldsymbol{\theta}_t)$, where $\boldsymbol{\theta}_t$ subsumes both body joint rotations and root translation. For objects, a global rigid transform $\mathcal{T}(\mathcal{G}^O; \phi)$ is applied (Sec. 7.2.1.2). In both cases, only positions and covariances are deformed; colors and opacities remain as predicted by the StyleUNets.

7.2.1.5 Composition and Occlusion Guided Loss

Given input images $\mathbf{I}_t^c$ with human and object masks $\mathbf{H}_t^c, \mathbf{O}_t^c$, and human and object poses $\boldsymbol{\theta}_t$ and $\boldsymbol{\phi}_t$ (camera index c , timestep t), we render posed human and object Gaussians $\mathcal{G}_t^H = \mathcal{T}(\mathcal{G}^H; \boldsymbol{\theta}_t)$ and $\mathcal{G}_t^O = \mathcal{T}(\mathcal{G}^O; \boldsymbol{\phi}_t)$ to obtain rendered images:

$$\hat{\mathbf{I}}_H^{c,t} = \mathcal{R}(\mathcal{G}_t^H; \gamma_{c,t}), \quad \hat{\mathbf{I}}_O^{c,t} = \mathcal{R}(\mathcal{G}_t^O; \gamma_{c,t}), \quad \hat{\mathbf{I}}_A^{c,t} = \mathcal{R}(\mathcal{G}_t^H \cup \mathcal{G}_t^O; \gamma_{c,t}). \quad (7.4)$$

To avoid penalizing occluded regions, we ignore pixels where the other entity is visible. The per-frame, per-camera losses are:

$$\begin{aligned} \mathcal{L}_H^{c,t} &= \|\mathbf{I}_t^c \mathbf{H}_t^c (1 - \mathbf{O}_t^c) - \hat{\mathbf{I}}_H^{c,t} (1 - \mathbf{O}_t^c)\|_1, \quad \mathcal{L}_O^{c,t} = \|\mathbf{I}_t^c \mathbf{O}_t^c (1 - \mathbf{H}_t^c) - \hat{\mathbf{I}}_O^{c,t} (1 - \mathbf{H}_t^c)\|_1, \\ \mathcal{L}_A^{c,t} &= \|\mathbf{I}_t^c (\mathbf{H}_t^c + \mathbf{O}_t^c) - \hat{\mathbf{I}}_A^{c,t}\|_1. \end{aligned} \quad (7.5)$$

The overall image loss is

$$\mathcal{L}_1 = \sum_{c,t} (\mathcal{L}_H^{c,t} + \mathcal{L}_O^{c,t} + \mathcal{L}_A^{c,t}).$$

We additionally use a perceptual loss $\mathcal{L}_{\text{per}}$ [Zhang et al., 2018b] with the same masking strategy and a regularization loss $\mathcal{L}_{\text{reg}}$ that encourages predicted skinning weights to remain close to SMPL-derived weights. The final objective is:

$$\mathcal{L} = \lambda_{L1} \mathcal{L}_1 + \lambda_{\text{per}} \mathcal{L}_{\text{per}} + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}}, \quad (7.6)$$

where λ_{L1} , λ_{per} , λ_{reg} weight the respective terms (we set $\lambda_{L1} = 1.0$, $\lambda_{\text{per}} = 0.1$, $\lambda_{\text{reg}} = 0.005$). This formulation prevents erroneous supervision in occluded regions,

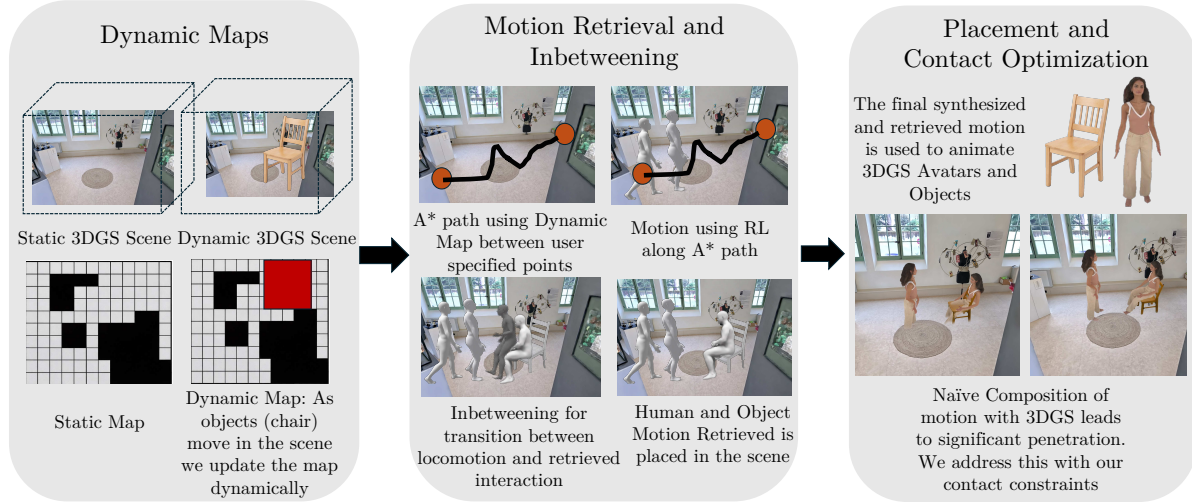


Figure 7.6: **Animation pipeline overview.** Given reconstructed human and object Gaussians, we animate them in diverse 3D scenes via dynamic 2D maps for collision-free navigation, retrieval-based motion synthesis with inbetweening, Gaussian placement and rendering, and contact refinement in Gaussian space.

while multi-view consistency ensures that ignored Gaussians are still supervised in other frames or views. Furthermore, our *canonical* Gaussian map formulation allows us to optimize the 6D object pose during training as well and hence refine the initially tracked object 6D parameters.

7.2.2 Animation

Given the reconstructed human and object Gaussians, we now describe how to animate them in 3D scenes (Fig. 7.6). We introduce 2D Dynamic Maps for collision-free navigation in changing scenes (Sec. 7.2.2.1), a retrieval-based motion synthesis pipeline (Sec. 7.2.2.2), Gaussian placement and rendering to convert skeletal motion into photo-realistic 3DGS renderings (Sec. 7.2.2.3), and contact constraints in Gaussian space to ensure plausible contacts (Sec. 7.2.2.4).

7.2.2.1 2D Dynamic Maps for Collision-Free Navigation

When navigating a 3DGS scene $\mathcal{G}^S$, the avatar must avoid both static obstacles and dynamic objects whose positions change as the human interacts with them. Following [Mir et al., 2025b], we obtain a top-down 2D map via orthographic projection and use it for path planning. Unlike [Mir et al., 2025b] which considers only static scenes, we extend this to *dynamic* environments by updating the map after every interaction.

Scene alignment via PCA. 3DGS scenes reconstructed from casual captures do not have a canonical orientation. To establish a gravity-aligned coordinate frame, we run PCA on the scene Gaussian centers $\{\mu_k^S\}_{k=1}^{N_S}$. The eigenvector $\mathbf{e}_1$ with the *smallest* eigenvalue identifies the up direction—since real-world scenes are wider than they are

tall, the axis of least variance is the vertical. We form the rotation $\mathbf{R}_{\text{align}} = [\mathbf{e}_3, \mathbf{e}_2, \mathbf{e}_1]^\top$ (columns ordered by descending eigenvalue) so that $+Z$ points upward. After rotation, we detect the floor level as the 2nd percentile of the Z coordinates and shift so that $Z=0$ corresponds to the floor.

Map construction. After alignment, we construct a top-down occupancy map by orthographic projection along Z . We discretize the XY plane into an $M \times M$ grid (we use $M=4096$). Each grid cell u is marked as *walkable* if the nearest projected Gaussian is farther than a threshold τ_{occ} ; otherwise it is an obstacle. This yields the static scene map $\mathcal{W}^S \in \{0, 1\}^{M \times M}$.

Dynamic updates. To obtain the dynamic map $\mathcal{W}_t$, we additionally project the object Gaussians. Given the current object pose ϕ_t , we transform the object Gaussians into the aligned frame via $\mathbf{R}_{\text{align}} \cdot (\mathbf{R}_{\phi_t} \mathbf{v} + \mathbf{t}_{\phi_t})$ and project them onto the XY plane. Grid cells covered by the projected object are marked as obstacles. We dilate the object footprint by a small kernel to provide a safety margin, then compose: $\mathcal{W}_t(u) = \mathcal{W}^S(u) \wedge \mathcal{W}_t^O(u)$, so a cell is walkable only if it is free in both maps. After each interaction moves the object, we recompute $\mathcal{W}_t^O$ with the updated object pose and re-plan the path using A^* .

7.2.2.2 Motion Retrieval and Inbetweening

To synthesize long sequences of an avatar navigating a scene and interacting with multiple objects, we adopt a retrieval-based approach. We use the tracked human poses $\{\theta_t\}$ and object poses $\{\phi_t\}$ obtained from our reconstruction step (Sec. 7.2.1) as interaction motions. The user specifies interaction locations; each retrieved sequence is then placed at the desired location $\mathbf{p}_i$ in the scene by applying a rigid transformation that aligns the interaction to the target position and the scene floor.

Given L interactions at scene locations $\mathbf{p}_1, \dots, \mathbf{p}_L$, we chain them into a continuous motion as follows.

RL-based locomotion. Between consecutive interaction locations, we generate collision-free locomotion using the RL-based policy of DIMOS [Zhao et al., 2023], which generates realistic walking motions along a given path. The input is a sequence of 2D waypoints obtained from A^* path planning on $\mathcal{W}_t$. We smooth the raw A^* path using B-spline fitting with collision-free enforcement to remove sharp turns. DIMOS then produces SMPL body pose sequences $\{\theta_t\}$ that follow the smoothed path while synthesizing natural, diverse locomotion. Crucially, unlike prior scene-aware locomotion methods that operate on static environments, our locomotion is planned on the *dynamic* maps that reflect the current object configuration.

Motion inbetweening. Locomotion segments and interaction segments generally do not start and end with matching poses. To produce smooth transitions, we use DartControl [Zhao et al., 2025a], a diffusion-based autoregressive motion model. Given $n_{\text{seed}}=3$ seed frames from the end of the preceding segment and a single target frame from the next segment, DartControl generates an inbetweening sequence τ that naturally transitions between the two. A text prompt (e.g., “walk” or “turn”) guides the style of the transition; the prompt is selected automatically based on the angle between the avatar’s current facing direction and the direction to the target location.

The full assembled motion for L interactions takes the form:

$$\underbrace{\text{loco}_0}_{\mathbf{p}_{\text{start}} \rightarrow \mathbf{p}_1} \rightarrow \tau_{\text{in}}^1 \rightarrow \mathcal{I}_1 \rightarrow \tau_{\text{out}}^1 \rightarrow \underbrace{\text{loco}_1}_{\mathbf{p}_1 \rightarrow \mathbf{p}_2} \rightarrow \cdots \rightarrow \mathcal{I}_L \rightarrow \tau_{\text{out}}^L \rightarrow \underbrace{\text{loco}_L}_{\mathbf{p}_L \rightarrow \mathbf{p}_{\text{end}}}, \quad (7.7)$$

where $\mathbf{p}_{\text{start}}$ and $\mathbf{p}_{\text{end}}$ are the user-specified start and end locations, $\mathcal{I}_i$ denotes the i -th retrieved interaction, τ_{in}^i and τ_{out}^i are inbetweening transitions, and loco_i is the locomotion segment between consecutive locations.

Object dynamics. During an interaction $\mathcal{I}_i$, the object follows retrieved poses ϕ_t . Between interactions, the object is either static at its last placed position, or—if the interaction involves carrying (e.g., a pickup)—it is attached to the avatar’s wrist joint and its pose is derived from the SMPL forward kinematics at each frame. After a *put-down* interaction, the object becomes static at its new location; the dynamic map $\mathcal{W}_t$ is updated accordingly (Sec. 7.2.2.1) for subsequent path planning.

7.2.2.3 Gaussian Placement and Rendering

The preceding steps produce SMPL pose sequences $\{\theta_t\}$ and object poses $\{\phi_t\}$, but these are skeletal parameters—not yet photorealistic 3DGS renderings. To obtain renderable Gaussians, we drive the reconstructed human and object models (Sec. 7.2.1) with the synthesized motion and compose them with the scene.

PCA pose projection. To improve generalization to unseen poses at animation time, following [Li et al., 2024d] we constrain novel poses to remain within the distribution of training poses using PCA. Let $\{P_t\}_{t=1}^T$ denote the set of training position maps, each vectorized as $\mathbf{p}_t \in \mathbb{R}^D$, where $D = H \times W \times C$ is the flattened position-map dimension. We perform PCA on the data matrix $\mathbf{X} = [\mathbf{p}_1, \dots, \mathbf{p}_T]$, obtaining the mean $\bar{P} \in \mathbb{R}^D$ and the top K principal components $\mathbf{Q}_K \in \mathbb{R}^{D \times K}$ with corresponding standard deviations $\{\sigma_i\}_{i=1}^K$. A novel test-time position map, vectorized as $\mathbf{p}$, is projected into this subspace by

$$\mathbf{z} = \mathbf{Q}_K^\top (\mathbf{p} - \bar{P}), \quad (7.8)$$

where each coefficient z_i is clipped to the interval $[-2\sigma_i, 2\sigma_i]$ to enforce plausibility. The low-dimensional reconstruction is then

$$\tilde{P} = \bar{P} + \mathbf{Q}_K \mathbf{z}, \quad (7.9)$$

which is reshaped back into a position map.

Human Gaussians. The reconstructed map $\tilde{P}$ is fed into the learned StyleUNet S_H , which predicts canonical human Gaussians $\mathcal{G}^H$. These are then deformed to world coordinates via LBS (Sec. 7.2.1.4), yielding posed human Gaussians $\mathcal{G}_t^{H,P} = \mathcal{T}(\mathcal{G}^H; \theta_t)$. This projection strategy ensures that novel poses from the animation pipeline are smoothly interpolated within the training distribution, improving robustness and preventing artifacts from out-of-distribution poses.

Object Gaussians. The object StyleUNet S_O predicts canonical object Gaussians $\mathcal{G}^O$ via the canonical Gaussian maps (Sec. 7.2.1.3). A rigid transformation defined by ϕ_t places them in the scene: $\mathcal{G}_t^{O,P} = \mathcal{T}(\mathcal{G}^O; \phi_t)$.

Interaction placement and floor alignment. The tracked interaction motions from our reconstruction step are in the coordinate frame of the capture setup (Y-up), which differs from the scene’s Z-up frame. We apply the rotation

$$\mathbf{R}_{\text{coord}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$$

to convert. For SMPL, the body model’s LBS rotates around the pelvis joint $\mathbf{J}_0$, not the origin. Naively applying $\mathbf{R}_{\text{coord}}$ to the translation introduces a systematic offset proportional to $(\mathbf{R}_{\text{coord}} - \mathbf{I}) \mathbf{J}_0$. To preserve exact human–object contact, we correct for this:

$$\mathbf{R}_{\text{new}} = \mathbf{R}_{\text{coord}} \mathbf{R}_{\text{old}}, \quad (7.10)$$

$$\mathbf{t}_{\text{new}} = \mathbf{R}_{\text{coord}} (\mathbf{t}_{\text{old}} + \mathbf{J}_0(\beta)) - \mathbf{J}_0(\beta), \quad (7.11)$$

where $\mathbf{R}$ denotes the global orientation, $\mathbf{t}$ the root translation, and $\mathbf{J}_0(\beta)$ is the pelvis joint location computed from the SMPL shape parameters β . After coordinate transformation, the interaction is placed at the target scene location $\mathbf{p}_i$. Because the RL-generated locomotion and the tracked interactions may have slightly different floor heights, we detect and correct vertical mismatches by running an SMPL forward pass on the boundary frames, computing the lowest body vertex Z-coordinate (foot contact), and applying a vertical offset ΔZ so that the feet land on the scene floor.

Scene composition. The posed human and object Gaussians are composited with the static scene Gaussians $\mathcal{G}^S$ and rendered via standard differentiable 3DGS rasterization from any desired viewpoint, producing photorealistic novel-view videos of the animated interaction within the reconstructed environment.

7.2.2.4 Contact Refinement in Gaussian Space

Naive composition of the posed human, object, and scene Gaussians from Sec. 7.2.2.3 often causes penetrations or missing contacts. We address this by optimizing small per-Gaussian displacements $\Delta_{k,t} \in \mathbb{R}^3$ on the human Gaussians. Let $\mu_{k,t}^H \in \mathbb{R}^3$ denote the center of human Gaussian k at time t , and $\mu_{p,t}^O \in \mathbb{R}^3$ the center of object Gaussian p at time t (both in posed/world space). However, unlike SMPL, Gaussian avatars lack correspondence with a fixed template. Therefore, we cannot manually define contact Gaussian primitives on the human avatar. Instead, to identify contact Gaussians, we define SMPL contact vertices V_c^{SMPL} (feet, hips, hands) and assign them to canonical Gaussians by nearest-neighbour search $i^* = \arg \min_k \|\mu_k^{H,C} - \mathbf{u}\|_2$ with $\mathbf{u} \in V_c^{\text{SMPL}}$, yielding the index set $\mathcal{I}_c$ (a subset of the human Gaussians). At detected contact frames t , we solve for displacements $\{\Delta_{k,t}\}$ with the objective

$$\begin{aligned} \mathcal{L}_t = & \lambda_c \sum_{k \in \mathcal{I}_c} \|\mu_{k,t}^H + \Delta_{k,t} - \mu_{p(k,t),t}^O\|_2^2 + \lambda_p \sum_{k \notin \mathcal{I}_c} h_r(d_\kappa(\mu_{k,t}^H + \Delta_{k,t}; \mathcal{G}_t^O))^2 \\ & + \lambda_r \sum_{k=1}^{N_H} \|\Delta_{k,t}\|_2^2 + \lambda_t \sum_{k=1}^{N_H} \|\Delta_{k,t} - \Delta_{k,t-1}\|_2^2, \end{aligned} \quad (7.12)$$

where $\mu_{p(k,t),t}^O$ is the nearest object Gaussian center to $\mu_{k,t}^H$ in posed space, i.e.,

$$p(k, t) = \arg \min_p \|\mu_{k,t}^H - \mu_{p,t}^O\|_2 ,$$

N_H and N_O are the number of human and object Gaussians, and

$$d_\kappa(\mathbf{x}; \mathcal{G}_t^O) = -\frac{1}{\kappa} \log \sum_{p=1}^{N_O} \exp(-\kappa \|\mathbf{x} - \mu_{p,t}^O\|_2)$$

is the soft nearest-neighbour distance to posed object Gaussians with temperature $\kappa > 0$ controlling the sharpness of the soft minimum. Further, $h_r(d) = \max(0, r - d)$ is a hinge margin with separation margin $r > 0$, and $\lambda_c, \lambda_p, \lambda_r, \lambda_t$ weight the contact, separation, regularization, and temporal smoothness terms.

Intuitively, this optimization forces contact Gaussians to snap onto the object surface, pushes non-contact Gaussians at least a margin r away from the posed object, penalizes large displacements, and encourages temporal smoothness. The refined human positions are $\mu_{k,t}^H + \Delta_{k,t}$, yielding a refined set of Gaussians $\tilde{\mathcal{G}}_t^H$. These are then composited with $\mathcal{G}_t^O$ and $\mathcal{G}^S$ and jointly rendered, producing photorealistic interactions with coherent contacts.

Human–scene and object–scene contacts. The same soft nearest-neighbour framework applies to contacts between the human and the scene (e.g., feet on the floor during locomotion, hips on a chair) and between the object and scene surfaces (e.g., a box resting on a table). For human-scene contacts, we replace $\mu_{p,t}^O$ with scene Gaussian centers μ_j^S in the objective above and identify foot and hip contact Gaussians via the same SMPL-to-Gaussian lifting. For object-scene contacts, we apply analogous displacements to the object Gaussians closest to scene surfaces. These additional constraints prevent floor penetration during walking and ensure placed objects rest stably on scene geometry.

7.3 IMPLEMENTATION DETAILS

We detail the StyleUNet architecture used for both human and object Gaussian map prediction, the feature-based object reconstruction procedure, and the initialization and training procedure.

7.3.1 Network Architecture

Following AnimatableGaussians [Li et al., 2024d], we employ the DualStyleUNet architecture—a U-Net augmented with StyleGAN2 [Karras et al., 2020] modulated convolutions—as the backbone for both human and object Gaussian map prediction. The architecture combines multi-scale skip connections with style-based modulation. An overview is shown in Fig. 7.7.

Encoder. The encoder processes 512×512 position maps (3 channels) through a series of downsampling blocks. An initial convolution maps the input from 3 to 128

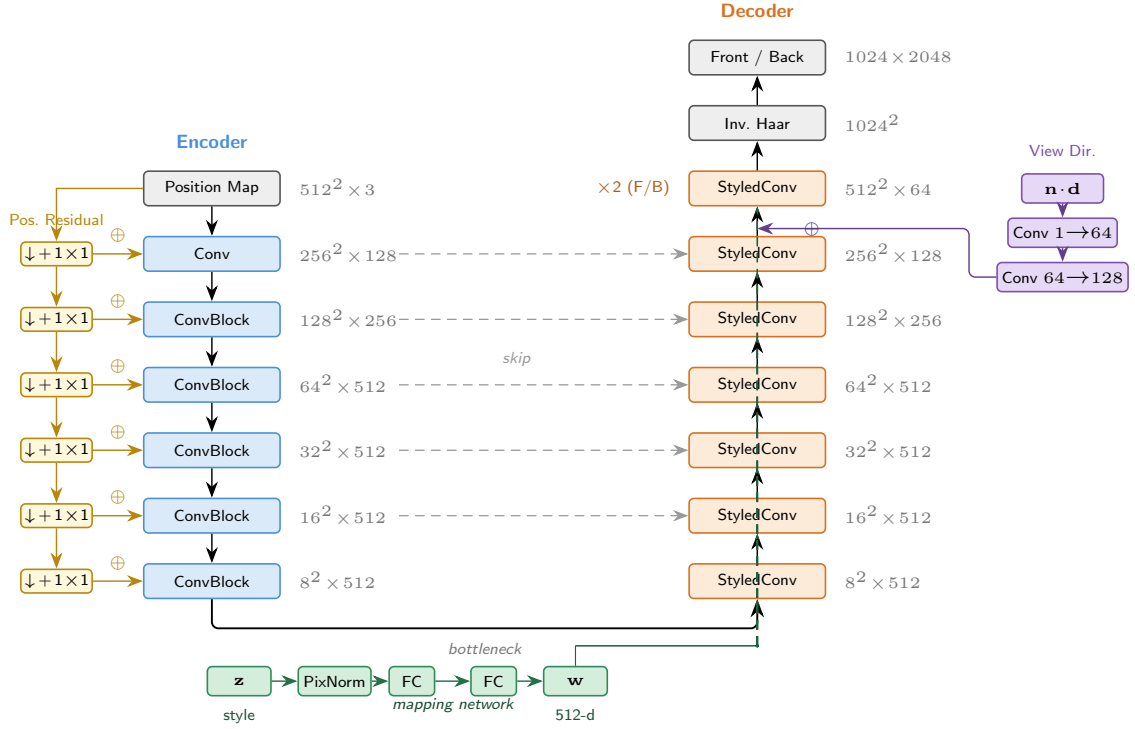


Figure 7.7: **StyleUNet architecture overview.** The encoder downsamples position maps with residual injection at each scale. The decoder uses StyleGAN2 modulated convolutions with skip connections from the encoder. A mapping network transforms style vector z into W space to modulate all decoder layers. View-direction features are injected at 256×256 . An inverse Haar transform converts the wavelet output to 1024×1024 . Two parallel branches (front/back) produce the 1024×2048 output.

channels at 256×256 resolution. Five subsequent ConvBlock stages progressively downsample to a 8×8 bottleneck with 512 channels. At each stage, the position map is also independently downsampled and projected via a 1×1 convolution, then added as a residual to the main feature path. This multi-scale residual injection ensures the encoder retains direct access to the raw positional information at every spatial resolution. The channel widths are: 128 at 256×256 , 256 at 128×128 , then 512 at all remaining resolutions (64×64 down to 8×8).

Decoder. Two parallel decoder branches (front and back UV maps) upsample from the 8×8 bottleneck to 512×512 in the Haar wavelet domain, which is then converted to 1024×1024 spatial resolution via an inverse Haar wavelet transform. Each branch consists of six upsampling stages. At each stage, a StyledConv with learnable upsampling doubles the spatial resolution, followed by a second StyledConv for further processing. Both convolutions use 3×3 modulated kernels with weight demodulation and per-pixel noise injection, following StyleGAN2. The decoder channel widths are: 512 channels from 16×16 to 64×64 , then 256 at 128×128 , 128 at 256×256 , and 64 at 512×512 .

Skip connections. At each decoder level, encoder features at the corresponding spatial resolution are concatenated with the decoder features along the channel dimension,



Figure 7.8: **Qualitative results of our method.** Reconstructed humans interacting with diverse objects in *novel human and object poses* from three datasets: *NeuralDome*, *BEHAVE*, *DNA-Rendering*.

then fused back to the original channel width via a 3×3 convolution.

Style modulation. A mapping network consisting of a pixel normalization layer followed by two equalized-learning-rate linear layers ($512 \rightarrow 512$, with $\text{lr_mul} = 0.01$) maps a 512-dimensional style vector to $\mathbf{W}$ space. The resulting style vector modulates convolution weights at each decoder layer. Style vectors are initialized as constant unit vectors ($\mathbf{1}/\sqrt{512}$) and kept fixed during inference.

Output. Each decoder branch produces a wavelet-domain representation with $C_{\text{out}} \times 4$ channels (four Haar subbands), which is converted to spatial domain at 1024×1024 resolution. The front and back maps are concatenated, yielding a 1024×2048 output map.

Network Instances. We instantiate two StyleUNets:

Human StyleUNet S_H . Input: pose-dependent position maps from projecting posed SMPL vertices onto front/back UV maps. Output: offsets from the canonical template via three heads—color ($C_{\text{out}} = 3$), position ($C_{\text{out}} = 3$, scaled by 0.05), and attributes ($C_{\text{out}} = 8$). S_H also predicts skinning weights $\mathbf{w}_{k,j}^H$ for LBS.

Object StyleUNet S_O . Input: time-invariant canonical object Gaussian maps from projecting the coarse template from front and back views. Since the object is rigid, the same canonical maps are reused for all timesteps. The output mirrors S_H (color, position, and attribute heads), predicting offsets from the coarse canonical object template.

View-direction conditioning. For the color head, a small CNN encodes a view-direction dot-product map (surface normal dotted with the viewing direction). It consists of two convolution layers ($1 \rightarrow 64$ and $64 \rightarrow 128$ channels, both 4×4 kernels, stride 2) separated by LeakyReLU. The resulting 128-channel features are injected into the decoder at 256×256 resolution, enabling view-dependent color prediction.

7.3.2 Feature-Based Object Reconstruction

As described in Sec. 7.2.1.1, we learn a coarse canonical object template using triplane features rather than direct 3DGS optimization. Alg. 3 summarizes this procedure in

pseudocode.

The triplane feature grid uses a spatial resolution of $H_p = W_p = 64$ per plane with a feature dimension of $l = 32$, yielding a compact representation of $3 \times 64 \times 64 \times 32$ learnable parameters. Each of the three axis-aligned planes captures complementary spatial information. The MLPs that decode triplane features into Gaussian properties consist of two hidden layers with 128 units and ReLU activations. We use separate MLP heads for geometry (center offset, scale, rotation) and appearance (color, opacity), allowing independent control over spatial and photometric properties.

During template learning, we initialize 3D candidate sites by uniformly sampling points within the bounding box of the object masks across all cameras. The sites are learnable parameters that move during optimization to cover the object surface. We apply small random jitter ($\sigma = 0.01$) to site positions during training to encourage exploration and prevent degenerate solutions where sites collapse to a small region. The template is optimized for approximately 30,000 iterations on the single frame t^* with minimal occlusion, using only pixels within the object mask for supervision.

Algorithm 3: Train 3D Gaussians from segmented multi-view images with features.

Input: Multi-view RGB images with object masks, initial 3D candidate sites (learnable), triplane feature grid (learnable), MLPs mapping features to Gaussian parameters, camera calibrations.

Output: Optimized feature grid, 3D Gaussian sites, and trained MLPs.

repeat

 Pick one camera view and a small batch of 3D candidate sites;

foreach *site in the batch* **do**

 Use its current 3D position; optionally apply jitter for exploration;

 Sample the triplane feature grid at this position (bilinear interpolation);

 Pass the feature through MLPs to predict Gaussian parameters: center offset, scale, rotation, color, and opacity;

 Collect all Gaussians and render from the chosen camera to produce a predicted RGB image and silhouette;

 Compute loss inside the object mask: photometric (mean absolute RGB difference) + mask (silhouette overlap);

 Combine terms with chosen weights to get total loss;

 Backpropagate and update the feature grid, 3D site locations, and MLP weights;

until *validation stops improving or maximum iterations reached;*

return *learned feature grid, optimized 3D sites, and trained MLPs.*

7.3.3 Training Details

Network Initialization. Since all StyleUNets learn offsets from canonical templates, proper initialization is critical to ensure that the network starts from the template



Figure 7.9: **Qualitative comparison with existing methods for animatable human reconstruction.** Existing methods are unable to reconstruct animatable objects.

geometry rather than from random predictions. We first train the human StyleUNet S_H for approximately 2,000 iterations on randomly sampled human position maps, with a target of zero offsets, forcing it to learn the identity mapping. Similarly, the object StyleUNet S_O is trained for approximately 2,000 iterations to map the pose-independent canonical position map to zero offsets. For the coarse canonical object template (Sec. 7.2.1.1), we use two separate MLPs: one mapping triplane features to geometry parameters (scale, covariance) and one mapping to appearance parameters (color, opacity). The triplane feature grid and MLP weights are jointly optimized during the template learning stage.

Training Procedure. We use Adam [Kingma and Ba, 2015] for all optimization with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate is set to 10^{-4} with a cosine annealing schedule:

$$\text{lr}(i) = \text{lr}_{\text{init}} \cdot \left[\frac{1}{2} (\cos(\pi \cdot i/N) + 1) (1 - \eta_{\min}) + \eta_{\min} \right], \quad (7.13)$$

where i is the current iteration, N is the total number of iterations, and $\eta_{\min} = 0.05$ ensures the learning rate never drops below 5% of the initial value. The final StyleUNets are trained for 800,000 iterations with a batch size of 1. At each iteration, we randomly sample one camera view and render a 512×512 patch for supervision.

7.4 EXPERIMENTS

We evaluate our method along two axes: *reconstruction* quality of animatable human-object models (Sec. 7.4.1), where we compare against existing methods on standard image metrics, and *animation* quality (Sec. 7.4.2), where we evaluate the synthesized interactions and scene-level motion.

7.4.1 Reconstruction Evaluation

Datasets. We use three datasets of human-object interaction: BEHAVE [Bhatnagar et al., 2022], captured with four cameras; DNA-Rendering [Cheng et al., 2023],

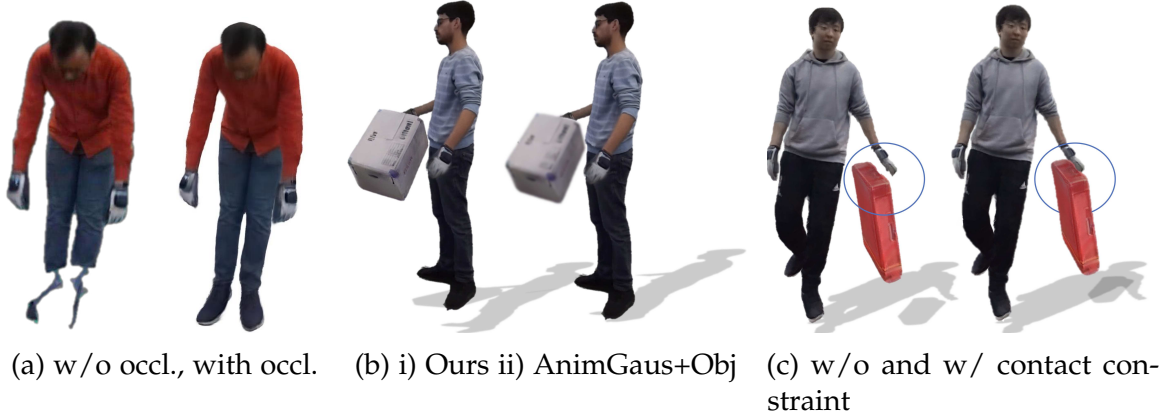


Figure 7.10: Ablation and modified baseline comparison.

Table 7.1: **Quantitative results on DNA-Rendering.** Evaluated on held-out images. We outperform existing baselines that reconstruct animatable 3D humans.

Subject: Metric:	Human			Object			Full		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
HUGS	22.13	0.7674	0.0437	17.54	0.6878	0.0452	21.13	0.7180	0.0425
ExAvatar	25.41	0.7743	0.0371	16.20	0.5752	0.0379	23.18	0.7632	0.0356
ToMiE	26.85	0.7700	0.0366	20.30	0.6581	0.0353	25.49	0.7790	0.0323
AnimGau	27.85	0.8900	0.0326	23.21	0.7550	0.0374	26.49	0.8656	0.0314
AnimGau+Obj	28.16	0.9090	0.0306	25.81	0.8430	0.0314	27.49	0.8956	0.0304
Ours	28.14	0.9174	0.0289	28.63	0.8973	0.0298	28.31	0.9242	0.0292

recorded with 60 cameras (48 train / 12 test); and **NeuralDome** [Zhang et al., 2023], a multi-view capture of human-object interactions. For BEHAVE we train on a subset of each sequence and test on held-out frames with the same cameras (only four available); for DNA-Rendering we evaluate on held-out novel views. In Fig. 7.8 we show qualitative results on all three datasets.

We additionally evaluate on the two-sequence NeuralDome benchmark. As shown in Tab. 7.3, our method consistently outperforms both AnimGaus and its modified object-aware variant. While AnimGau+Obj improves over the vanilla baseline, it still struggles with accurate object reconstruction. In contrast, our approach achieves the best scores across all metrics, following the same pattern observed on BEHAVE and DNA-Rendering.

7.4.1.1 Unmodified Baselines

We compare our approach with ExAvatar [Moon et al., 2024], AnimatableGaussians [Li et al., 2024d], HUGS [Kocabas et al., 2024], and ToMiE [Zhan et al., 2025b]. As these methods are primarily designed to reconstruct and animate humans recorded in isolation, they need to be modified for objects. In this experiment, to reconstruct

Table 7.2: **Quantitative results on BEHAVE.** Evaluated on held-out images with novel human and object poses.

Subject: Metric:	Human			Object			Full		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
HUGS	23.11	0.8274	0.0340	19.34	0.6378	0.0474	22.12	0.7434	0.0395
ExAvatar	25.41	0.8130	0.0341	18.35	0.6752	0.0390	22.18	0.7820	0.0366
ToMiE	23.85	0.7840	0.0346	18.51	0.6781	0.0394	21.49	0.7260	0.0374
AnimGau	26.85	0.8700	0.0266	21.50	0.7581	0.0344	24.99	0.8160	0.0294
AnimGau+Obj	27.65	0.8710	0.0287	24.50	0.8431	0.0334	26.11	0.8130	0.0299
Ours	27.64	0.8741	0.0285	26.39	0.8732	0.0325	26.92	0.8724	0.0292

Table 7.3: **Quantitative results on the NeuralDome benchmark.** Our method achieves the best performance across human, object, and full metrics.

Subject: Metric:	Human			Object			Full		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
AnimGaus	26.85	0.8700	0.0266	21.50	0.7581	0.0344	24.99	0.8160	0.0294
AnimGau+Obj	27.65	0.8710	0.0287	24.50	0.8431	0.0334	26.11	0.8130	0.0299
Ours	27.94	0.8741	0.0258	26.39	0.8732	0.0312	26.92	0.8724	0.0281

humans and objects, we modify the segmentation masks provided in the datasets to include the object along with the humans. All compared methods reconstruct a human using 3DGS, with a canonical space deformed via LBS to posed space and supervised against ground truth pixels. Since we force all these methods to reconstruct the object as well, they are forced to modify the canonical Gaussians to explain pixels detached from the human body. This results in artifacts. The object either disappears or is reconstructed as a blob-like surface, as shown in Fig. 7.9. All metrics are computed for *novel poses* on held-out test images (Tab. 7.2, Tab. 7.1), with separate human and object scores. In Fig. 7.11 we show additional renderings from all unmodified baselines alongside our method and the ground truth on the BEHAVE dataset.

7.4.1.2 Comparison with DynamicGaussians

Unlike our method, DynamicGaussians [Luiten et al., 2024] and follow-up works are not controllable—they only replay the original sequence and cannot synthesize novel human-object interactions. We provide a qualitative comparison in Fig. 7.12. Notably, for replaying the original sequence, our method produces sharper renderings than DynamicGaussians, but is much slower to train.

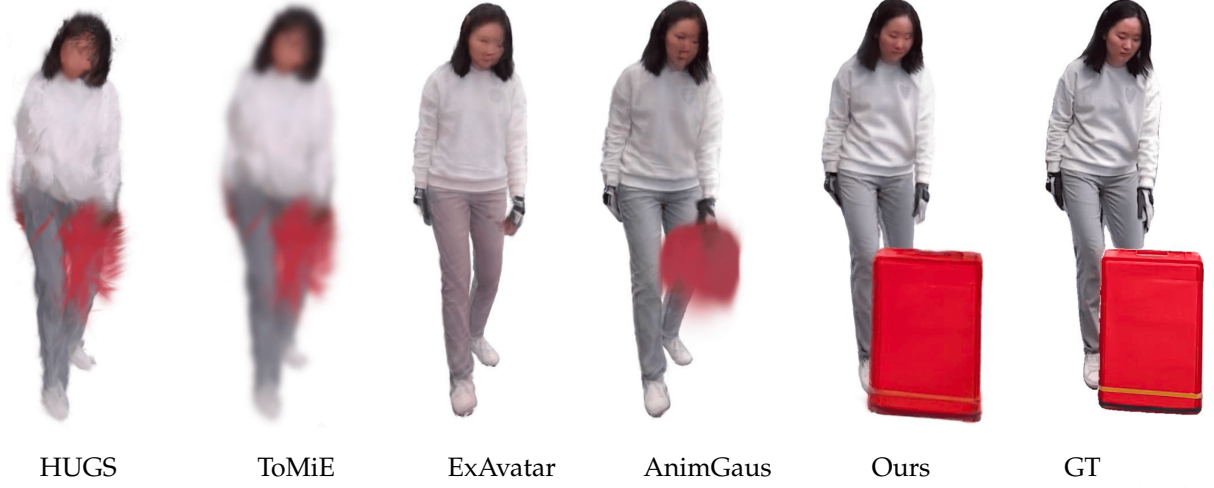


Figure 7.11: **Comparison with unmodified baselines on BEHAVE.** Existing methods are unable to reconstruct animatable objects.



Figure 7.12: **Comparison with DynamicGaussians.** Left: Ours, Right: DynamicGaussians (segmented).

7.4.1.3 Modified Baseline

We also evaluate **AnimGau+Obj**, which uses AnimatableGaussians with separate human and object reconstruction using our tracked template, but without our Gaussian Map formulation. Human quality is comparable to ours, but the object is clearly blurry without Gaussian Maps (Fig. 7.10b, Tab. 7.2, Tab. 7.1).

7.4.1.4 Ablation Studies

In Tab. 7.4 and Fig. 7.10, we evaluate our design choices. Each method component improves the results. **(1) Contact Constraint.** We evaluate the contact constraint on the same human+object using GT images but for novel poses (**w/o Contact** in Tab. 7.4). The improvement for same human+object is minimal but pronounced for novel human-object interaction (Fig. 7.10c). **(2) Obj Gaussian Map.** We remove the canonical Gaussian maps and generate final renderings using the coarse object template. Canonical Gaussian maps enable reconstruction of high-frequency details.

Table 7.4: **Ablation study on DNA-Rendering, BEHAVE, and NeuralDome.** Each component in our method adds to the performance and our method cumulatively yields best results. Each line in the table corresponds to an experiment detailed in Sec. 7.4.1.4.

Subject: Metric:	DNA-Rendering			BEHAVE			NeuralDome		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
w/o Features	26.11	0.9674	0.0410	27.54	0.9678	0.0464	25.00	0.9218	0.0595
w/o Obj Map	30.41	0.9743	0.0241	33.20	0.9752	0.0290	26.18	0.9232	0.0356
w/o Occ Loss	29.12	0.9727	0.0266	32.94	0.9695	0.0360	26.63	0.9141	0.0418
w/o Contact	32.34	0.9664	0.0209	32.63	0.9673	0.0259	27.44	0.9674	0.0289
Full (Ours)	32.64	0.9774	0.0209	33.63	0.9773	0.0258	27.64	0.9574	0.0285

(Tab. 7.4 (w/o Obj Map) and Fig. 7.14). **(3) Features.** We optimize the locations of the 3DGS parameters directly. Naive 3DGS fails due to significant occlusion patterns in the captured videos (w/o Features in Tab. 7.4). Without features, the template reconstruction does not converge. **(4) Occlusion Loss.** We switch off the occlusion-aware loss, forcing occluded regions to be part of the background, which degrades reconstruction quality (Fig. 7.10a, w/o Occ Loss in Tab. 7.4).

In Fig. 7.13, we show further examples comparing our Gaussian map formulation against learning Gaussian properties in a 3D feature space. Our design yields sharper results with finer details.



Figure 7.13: **Gaussian maps vs. 3D feature space. (Zoom in.)** Left: learning Gaussian properties as 2D Gaussian maps for dynamic objects under occlusion. Right: Gaussian properties in 3D feature space.

7.4.2 Animation Evaluation

We evaluate the animation pipeline (Sec. 7.2.2) qualitatively and quantitatively. In Fig. 7.8, we show reconstructed avatars animated to interact with novel objects. In Fig. 7.16, we demonstrate chained interactions placed in diverse 3D environments [SuperSplat]. We quantify animation quality with automatic contact metrics and a perceptual user study, ablating each component of the pipeline (Tab. 7.5).



Figure 7.14: **Gaussian map ablation.** Naive 3DGS fails due to occlusion.



Figure 7.15: **Without dynamic maps,** the avatar penetrates placed objects.

7.4.2.1 Metrics

Let $\mu_{k,t}^H$ denote the center of human Gaussian k at frame t , and let $Y \in \{O, S\}$ denote the target entity (object or scene). We write $\mu_{p,t}^Y$ for the center of the nearest target Gaussian p and $\mathbf{s}_p^Y$ for its scale vector. **Penetration Depth** measures how deeply human Gaussians penetrate into object or scene Gaussians. For each human Gaussian whose distance to its nearest object/scene Gaussian is smaller than that Gaussian’s effective radius $\|\mathbf{s}_p^Y\|_2$, we compute

$$d_k = \max(0, \|\mathbf{s}_p^Y\|_2 - \|\mu_{k,t}^H - \mu_{p,t}^Y\|_2)$$

and report the average over all penetrating Gaussians and frames. **Contact Distance** measures how close designated contact Gaussians (feet, hands, hips, identified via the SMPL lifting in Sec. 7.2.2.4) are to their target surface at detected contact frames:

$$d_c = \frac{1}{|\mathcal{I}_c|} \sum_{k \in \mathcal{I}_c} \min_p \|\mu_{k,t}^H - \mu_{p,t}^Y\|_2,$$

where $\mathcal{I}_c$ is the set of contact Gaussian indices. For the **User Study**, participants are shown pairs of rendered videos — our full method versus an ablated variant — and asked to select the more realistic one; we report the preference rate for our full method. Our user study had 13 naive raters who were shown 7 comparison videos.

7.4.2.2 Ablation

We ablate the following animation components: **(1) Dynamic Maps.** We replace the dynamic maps with a static map (**w/o Dynamic Maps**). Without dynamic maps, the avatar penetrates placed objects (Fig. 7.15). **(2) Inbetweening.** We remove inbetweening (**w/o Inbetweening**). **(3) Contact Refinement.** We disable Gaussian contact refinement (**w/o Contact**).

Tab. 7.5 confirms that each animation component contributes to the final result. Dynamic maps have the largest effect: without them, the avatar freely penetrates relocated objects (penetration depth increases from 1.8 mm to 6.9 mm), and 85.6% of users prefer the full pipeline. Contact refinement reduces contact distance from 23.7 mm to 8.2 mm, producing visually tighter grasps and foot placements. Inbetweening improves



Figure 7.16: **Scene-level animation and interaction with dynamic objects.**

Table 7.5: **Animation evaluation.** User study shows the percentage of participants preferring our method over each ablation in pairwise comparisons.

	Pen. Depth ↓ (mm)	Contact Dist. ↓ (mm)	User Study	
			Full	Ablation
w/o Contact	2.1	23.7	79.2%	20.8%
w/o Dynamic Maps	6.9	14.8	85.6%	14.4%
w/o Inbetweening	1.9	8.6	72.4%	27.6%
Full (Ours)	1.8	8.2		

perceptual quality (preferred by 72.4% of users) by smoothing transitions between locomotion and interaction segments. Together, these results confirm that collision avoidance, contact enforcement, and smooth motion blending are all necessary for convincing animated interactions in 3DGS scenes.

7.4.3 Additional Qualitative Results

Fig. 7.17 shows additional novel-view renderings across diverse human-object interaction sequences, demonstrating our method’s ability to handle objects of varying geometry and appearance.



Figure 7.17: **Additional qualitative results.** We show novel-view renderings of diverse human-object interaction sequences reconstructed by our method, demonstrating its ability to handle a wide variety of objects and interaction types.

7.5 CONCLUSION

We have presented InteractSplat, a 3DGS-based method for photorealistic, controllable rendering of humans interacting with dynamic objects in diverse 3D scenes. Unlike prior work that reconstructs humans in isolation and animates them in empty space, InteractSplat decouples human and object reconstruction, enabling independent pose control and, for the first time, synthesis of novel human-object interactions placed in varied 3D environments. To achieve this, we introduced several contributions: a coarse-to-fine object reconstruction approach using canonical Gaussian maps that yields high-fidelity textures for dynamic objects under heavy occlusion; a composition and occlusion-guided loss for joint human-object reconstruction; dynamic 2D maps of 3DGS scenes that enable collision-free avatar navigation in environments with moving objects; and contact constraints in Gaussian space that ensure plausible human-object, human-scene, and object-scene contacts during animation. Experiments on BEHAVE, NeuralDome, and DNA-Rendering demonstrate significant improvements in reconstruction quality over existing methods, and we showcase chained interactions—avatars lifting objects, moving furniture, and navigating dynamic 3D environments—that have not been demonstrated before in the context of photorealistic neural rendering. Our method currently models only one object per interaction and assumes rigid object motion, which excludes deformable objects. The reconstruction requires a multi-view capture setup and relies on pre-estimated SMPL poses and segmentation masks. Future work includes extending to monocular RGB input and non-rigid object deformations.

SALA: SCENE-ALIGNED ANIMATION OF HUMAN AVATARS IN DIVERSE 3D ENVIRONMENTS WITH GAUSSIAN SPLATTING



Figure 8.1: **Animating humans in 3DGS scenes.** We introduce a unified Gaussian-based representation for humans and environments, enabling synthesis of photoreal human–scene interactions entirely within the Gaussian splatting framework.

While Ch. 7 addressed human–object interaction in 3DGS scenes, this chapter tackles the broader problem of animating humans in diverse 3D environments using 3DGS as the sole underlying representation. Existing animation pipelines, including those based on the classical representations used in Part I, rely on meshes or point clouds to represent scenes and humans—these representations are versatile and provide explicit surfaces, signed distance fields, and walkable regions for motion synthesis and contact modeling. However, they cannot produce photorealistic renderings. Unlike these pipelines, our approach introduces 3DGS as the 3D representation for animating humans in scenes. This poses fundamental challenges: 3DGS provides none of the geometric quantities that existing methods rely on. We address these challenges with four contributions: (i) extracting navigable structures from raw Gaussian fields via opacity culling and PCA-alignment and orthographic projection, enabling pathfinding directly on Gaussians; (ii) adapting RL-based locomotion to 3DGS scenes with an egocentric walkability grid derived from scene Gaussians; (iii) a differentiable SDF proxy defined as nearest-neighbor distance to scene Gaussians, enabling fine-grained interaction synthesis via latent-space optimization with collision avoidance; and (iv) a

differentiable contact refinement that enforces human–scene contact through per-Gaussian offsets and soft nearest-neighbor distances, bypassing the intractable learned pose-to-Gaussian mapping. Together, these enable scene-aligned motion synthesis and photorealistic free-viewpoint rendering of human–scene interactions—without meshes or paired human–scene data. We evaluate on ScanNet++ and SuperSplat scenes with avatars from sparse and dense multi-view capture. Finally, we demonstrate that our framework enables geometry-consistent free-viewpoint rendering of casually captured mobile-phone videos edited with new animated humans—opening the door to applications such as gaming and AR/VR content creation in environments captured with consumer cameras. Video results are available at <https://miraymen.github.io/sala>.

8.1 INTRODUCTION

Human animation in 3D scenes is essential for applications ranging from video gaming and computer-generated imagery (CGI) to robotics. Recent research has made significant progress on generating humans in 3D scenes [Hassan et al., 2019, Jiang et al., 2024a, Hassan et al., 2021a, Zhao et al., 2023, Hwang et al., 2025], where humans are typically represented as either 3D skeletons or meshes, while background scenes are represented as meshes or point clouds. These representations are compact and versatile, capable of modeling a wide variety of surfaces. However, a fundamental limitation persists: there is almost always a domain gap between rendered results and real images due to limitations in lighting, materials, and geometric fidelity.

In parallel, neural scene representations have emerged, beginning with Neural Radiance Fields (NeRF) [Mildenhall et al., 2020] and more recently 3D Gaussian Splatting (3DGS) [Kerbl et al., 2023], enabling photorealistic rendering of objects, humans, and full 3D scenes from novel viewpoints or in novel poses. Yet, despite their success in rendering quality, neural representations have seen little to no adoption in human–scene animation pipelines, which continue to rely on mesh and point cloud-based frameworks.

Gaussian Splatting as a 3D representation for human–scene animation offers natural advantages over existing mesh-based representations. First, rendering human–scene interactions with Gaussian Splatting is more photorealistic. Second, 3DGS allows reconstructing scenes from only a casually captured mobile-phone video [Ling et al., 2024], thus enabling applications such as geometry-consistent free-viewpoint rendering of videos with new humans, personalized content creation, and gaming or AR/VR experiences in environments captured with consumer cameras. Such applications are difficult with mesh-based representations, as estimating meshes or point clouds from monocular scene videos alone remains challenging [Wang et al., 2025a]. This motivates the central question addressed in this chapter: *Can neural scene representations—specifically Gaussian Splatting—be effectively used as a 3D representation for human animation in 3D scenes?* (Fig. 8.1).

Answering this question requires overcoming challenges specific to the 3DGS representation that do not arise with meshes. A first obstacle is data: existing human–

scene datasets [Hassan et al., 2019, Jiang et al., 2024a, Bhatnagar et al., 2022] provide motion annotations exclusively in mesh form. Extending 3DGS to animation pipelines would ordinarily require paired motion-scene datasets in 3DGS, which do not exist. Collecting such data at scale is costly, and conversion from mesh-based annotations into Gaussians is non-trivial. We sidestep this by decoupling rendering from motion synthesis: humans and scenes are reconstructed as Gaussians independently, and motion is synthesized in a shared coordinate frame without requiring paired data (Sec. 8.2).

Beyond data, four technical challenges remain (Fig. 8.2). **(1) No navigable scene geometry.** Existing scene-aware motion methods [Hassan et al., 2021a, Jiang et al., 2024a, Hwang et al., 2025, Zhao et al., 2023] query mesh surfaces, signed distance fields, or walkable floor planes represented using navigation meshes—none of which exist in a 3DGS reconstruction. We address this by extracting obstacle maps via *opacity culling* and PCA-aligned orthographic projection, enabling A* pathfinding directly on raw Gaussian fields (Sec. 8.3.1). **(2) No local obstacle context for locomotion.** Given the paths from (1), we use an RL locomotion policy to generate motion along them while avoiding obstacles. However, such policies are trained on clean synthetic mesh environments and need adaptation for 3DGS, where reconstructions are noisy, incomplete, contain floaters, and lack topology. We introduce an *egocentric walkability grid* extracted from opacity-filtered scene Gaussians, providing the local obstacle context the policy requires at each step. Because the underlying motion model is text-conditioned, varying the text cue (e.g., “jump”, “twirl”) produces diverse locomotion styles beyond walking (Sec. 8.3.2). **(3) No differentiable distance for collision avoidance.** Synthesizing fine-grained interactions (sitting, grasping) requires a differentiable distance-to-scene function—meshes provide this via SDFs, but Gaussians do not. We define a *differentiable SDF proxy* as nearest-neighbor distance to scene Gaussians, and couple it with deterministic latent-space optimization, enforcing scene-aware constraints directly in Gaussian space (Sec. 8.3.3). **(4) No surfaces for contact and intractable pose optimization.** Unlike meshes, where contact can be enforced by projecting vertices onto surfaces or by optimizing SMPL pose parameters via LBS, 3DGS provides no explicit surface to snap to. The learned pose-to-Gaussian mapping—which, unlike LBS for meshes, involves a forward pass through a neural network—is intractable to optimize through. We address this by optimizing small per-Gaussian translation offsets using a soft nearest-neighbor distance to scene Gaussians, enforcing contact without explicit surfaces (Sec. 8.3.4).

We present results on scenes from the ScanNet++ dataset [Yeshwanth et al., 2023] and the publicly available SuperSplat 3DGS library [SuperSplat], and on avatars from the ActorsHQ [Işık et al., 2023], BEHAVE [Bhatnagar et al., 2022], and Avatar-ReX [Zheng et al., 2023] datasets—demonstrating that our method works with avatars reconstructed from as few as four cameras. Finally, we demonstrate geometry-consistent free-viewpoint rendering of casually captured mobile-phone videos edited with new animated humans on several videos from DL3DV [Ling et al., 2024]—showcasing a unique advantage of the 3DGS representation: the ability to animate humans in scenes reconstructed from everyday footage, opening the door to gaming and AR/VR content creation in environments captured with consumer cameras. To

summarize, our contributions are:

1. We introduce the 3D Gaussian Splatting representation to the classical computer graphics problem of animating humans in 3D environments.
2. Navigable structures from raw Gaussians via opacity culling and PCA-aligned projection, enabling A* pathfinding without meshes.
3. Gaussian-aligned locomotion via an egocentric walkability grid, adapting RL policies to noisy Gaussian reconstructions.
4. A differentiable SDF proxy for collision-aware interaction synthesis (sitting, grasping) via latent-space optimization.
5. Differentiable contact refinement in Gaussian space via soft nearest-neighbor distances, without explicit surfaces or pose optimization.

8.2 BACKGROUND: GAUSSIAN RECONSTRUCTION

Our method represents both scenes and humans as sets of 3D Gaussians that are reconstructed independently and later composed for rendering. This decoupling is key: it eliminates the need for paired human–scene data in 3DGS form, since motion is synthesized in a shared SMPL coordinate frame and only mapped to Gaussians at render time. We summarize both reconstructions below.

Scene Gaussians. Given a casually captured mobile-phone video or multi-view video of a static scene, we model the environment as a set of N_S anisotropic 3D Gaussians $\mathcal{G}^S = \{(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \alpha_i, \mathbf{c}_i)\}_{i=1}^{N_S}$, where each primitive has center $\boldsymbol{\mu}_i \in \mathbb{R}^3$, covariance $\boldsymbol{\Sigma}_i \in \mathbb{R}^{3 \times 3}$, opacity $\alpha_i \in (0, 1)$, and color $\mathbf{c}_i$. These Gaussians are projected to the image plane and alpha-composited front-to-back following the standard 3DGS pipeline [Kerbl et al., 2023]. Parameters are optimized with a photometric loss across frames.

Human Gaussians. We learn pose-conditioned human Gaussian representations from multi-view images following [Li et al., 2024d]. A per-identity StyleUNet f_ϕ maps SMPL pose maps $P_t = \mathcal{M}(\beta, \theta_t)$ to canonical Gaussians $\mathcal{G}_t^{H,C} = f_\phi(P_t)$ anchored on the SMPL surface via a front–back UV atlas, where $\mathcal{M}$ denotes the pose-map generation function (the SMPL-deformed template rendered under orthographic projection; see Sec. 8.2.2) and θ_t is the body pose at frame t . These are deformed to posed space via linear blend skinning (LBS) with SMPL joint transformations, yielding $\mathcal{G}_t^{H,P} = \{(\boldsymbol{\mu}_k^P, \boldsymbol{\Sigma}_k^P, \alpha_k, \mathbf{c}_k)\}_{k=1}^{N_H}$, where N_H is the number of human Gaussians. At test time, poses from our motion module (Sec. 8.3.2) are projected onto the top- K PCA components of training pose maps before being mapped to Gaussians, following [Li et al., 2024d]. Crucially, this learned mapping $\theta_t \mapsto \mathcal{G}_t^{H,P}$ passes through the network f_ϕ and LBS, making direct pose optimization for scene contact intractable—a challenge addressed in Sec. 8.3.4.

We now detail how LBS is adapted to deform Gaussian primitives (as opposed to mesh vertices, cf. Sec. 3.1) and describe the avatar learning framework. For the

standard 3DGS rendering pipeline (covariance parameterization, splatting, and alpha compositing), we refer the reader to Sec. 3.3.

8.2.1 Linear Blend Skinning for Posing

We deform canonical, SMPL-anchored [Pavlakos et al., 2019, Loper et al., 2015] Gaussians into posed space using linear blend skinning (LBS).

Canonical Gaussians and anchoring. Each canonical Gaussian is $(\boldsymbol{\mu}_k^C, \boldsymbol{\Sigma}_k^C, \alpha_k, \mathbf{c}_k)$, with center $\boldsymbol{\mu}_k^C \in \mathbb{R}^3$, covariance $\boldsymbol{\Sigma}_k^C \in \mathbb{R}^{3 \times 3}$, opacity $\alpha_k \in (0, 1)$, and color $\mathbf{c}_k$. Gaussians are anchored to the SMPL surface by nearest-neighbor association to mesh vertices.

Skinning weights. Let J be the number of SMPL joints (as in Sec. 3.1). For Gaussian k , we use skinning weights $w_{k,j} \in [0, 1]$, $\sum_{j=1}^J w_{k,j} = 1$, inherited from its associated SMPL vertex.

Joint transformations. Given pose parameters $\boldsymbol{\theta}_t$, SMPL provides per-joint rigid motions $\mathbf{R}_j(\boldsymbol{\theta}_t) \in \text{SO}(3)$ and $\mathbf{d}_j(\boldsymbol{\theta}_t) \in \mathbb{R}^3$, which are the rotation and translation components of the bone transformation $\mathbf{G}_j(\boldsymbol{\theta}_t, \mathbf{J})$ (Eq. (3.1)), relative to the canonical rest pose.

Position deformation. The posed center of Gaussian k at time t is the LBS blend of bone transforms:

$$\boldsymbol{\mu}_k^P(\boldsymbol{\theta}_t) = \sum_{j=1}^J w_{k,j} (\mathbf{R}_j(\boldsymbol{\theta}_t) \boldsymbol{\mu}_k^C + \mathbf{d}_j(\boldsymbol{\theta}_t)). \quad (8.1)$$

Covariance deformation. We approximate the posed covariance by applying the same blended rotational action:

$$\boldsymbol{\Sigma}_k^P(\boldsymbol{\theta}_t) \approx \left(\sum_{j=1}^J w_{k,j} \mathbf{R}_j(\boldsymbol{\theta}_t) \right) \boldsymbol{\Sigma}_k^C \left(\sum_{j=1}^J w_{k,j} \mathbf{R}_j(\boldsymbol{\theta}_t) \right)^\top. \quad (8.2)$$

This first-order treatment ignores shear and higher-order effects, but is accurate in practice and mirrors vertex LBS behavior.

Posed Gaussians and rendering. The posed human Gaussians are

$$\mathcal{G}_t^{H,P} = \{(\boldsymbol{\mu}_k^P, \boldsymbol{\Sigma}_k^P, \alpha_k, \mathbf{c}_k)\}_{k=1}^{N_H}, \quad (8.3)$$

which we render with the standard 3D Gaussian Splatting rasterizer. This defines a differentiable mapping $\mathcal{H} : \boldsymbol{\theta}_t \mapsto \mathcal{G}_t^{H,P}$, used during training and inference in combination with the pose-conditioned canonical prediction f_ϕ .

8.2.2 3DGS Avatar Learning Pipeline

The avatars follow the Animatable Gaussians framework [Li et al., 2024d]: a pose-dependent neural network maps SMPL body poses to 3D Gaussian primitives, which are then articulated via LBS. We summarize each stage below.

8.2.2.1 Template Reconstruction

Given multi-view video of a person, a character-specific canonical template is first reconstructed. Rather than relying on the naked SMPL mesh—which cannot represent loose garments such as dresses—a canonical signed distance field (SDF) and color field are optimized from a near-A-pose frame and a mesh is extracted via Marching Cubes [Li et al., 2024d]. SMPL skinning weights are diffused from the body surface to the template surface, enabling the template to be deformed via LBS while faithfully capturing the geometry of worn clothing.

8.2.2.2 Pose Maps and Gaussian Maps

The key idea behind Animatable Gaussians is to parameterize 3D Gaussians as 2D maps on the template surface, enabling powerful 2D convolutional architectures to predict pose-dependent Gaussian attributes.

Pose maps. For each pose θ_t , the canonical template is deformed via LBS and rendered under front and back orthographic projections, producing two position maps $\mathcal{P}_f(\theta_t)$ and $\mathcal{P}_b(\theta_t)$; together these constitute the pose map $P_t = \mathcal{M}(\beta, \theta_t)$. Each pixel stores the 3D coordinates of the corresponding deformed template vertex, encoding the current body configuration as a structured 2D image. Together, the front and back views form an approximate UV atlas of the body surface: because orthographic front and back projections cover nearly the entire mesh, every surface point is captured in at least one of the two maps, providing a dense surface-to-image correspondence analogous to a classical UV parameterization but without the seam artifacts and distortion of a full unwrapping.

Gaussian maps. A StyleUNet f_ϕ —a U-Net augmented with StyleGAN-based modulation layers [Li et al., 2024d]—maps the pose maps to *Gaussian maps*:

$$\mathcal{M}_f(\theta_t), \mathcal{M}_b(\theta_t) \leftarrow f_\phi(\mathcal{P}_f(\theta_t), \mathcal{P}_b(\theta_t), \mathcal{V}),$$

where $\mathcal{V}$ encodes the view direction relative to the template normal. Each pixel in a Gaussian map encodes a complete 3D Gaussian primitive: a position offset $\Delta\mu$ relative to the template surface, per-axis scales $\mathbf{s}$, a rotation quaternion $\mathbf{q}$, opacity α , and view-dependent color $\mathbf{c}$. The collection of all foreground pixels across the front and back maps yields the full set of canonical Gaussians $\mathcal{G}_t^{H,C}$.

Training. The StyleUNet is trained end-to-end from multi-view video (typically 16–48 cameras). For each training frame, the predicted canonical Gaussians are posed via LBS and rendered via standard 3DGS rasterization. Parameters are optimized with a photometric reconstruction loss (L1 + perceptual) between rendered and ground-truth images.

8.2.2.3 Pose Projection Strategy

To improve generalization to unseen poses, following [Li et al., 2024d], we constrain test-time poses to remain within the distribution of training poses using Principal Component Analysis (PCA). Let $\{P_t\}_{t=1}^T$ denote the set of training pose maps. We vectorize each P_t into $\mathbf{p}_t \in \mathbb{R}^D$, where $D = HWC$ is the flattened pose-map dimension, and stack them into a data matrix $\mathbf{X} = [\mathbf{p}_1, \dots, \mathbf{p}_T]$.

We perform PCA on $\mathbf{X}$, obtaining the mean $\bar{\mathbf{P}} \in \mathbb{R}^D$ and the top K principal components $\mathbf{Q}_K \in \mathbb{R}^{D \times K}$ with corresponding standard deviations $\{\sigma_i\}_{i=1}^K$. A novel test-time pose map, vectorized as $\mathbf{p}$, is projected into this subspace by

$$\mathbf{z}_y = \mathbf{Q}_K^\top (\mathbf{p} - \bar{\mathbf{P}}), \quad (8.4)$$

where each coefficient $z_{y,i}$ is clipped to the interval $[-2\sigma_i, 2\sigma_i]$ to enforce plausibility. The low-dimensional reconstruction is then

$$\tilde{P}_y = \bar{\mathbf{P}} + \mathbf{Q}_K \mathbf{z}_y, \quad (8.5)$$

which is reshaped back into a pose map of size $H \times W \times C$. The reconstructed map $\tilde{P}_y$ is fed into the StyleUNet f_ϕ and subsequently deformed with LBS to yield posed Gaussians $\mathcal{G}_y^{H,P}$. This projection strategy ensures that novel poses are smoothly interpolated within the training distribution, improving robustness and generalization.

8.3 METHOD

Building on the Gaussian reconstruction of Sec. 8.2, we present a method that enables virtual humans represented as Gaussian splats to navigate and interact in complex 3DGS environments. Our method assumes that the user provides start and goal positions, interaction locations, and an action label for each interaction (e.g., “sit”, “grab”, “jump”). Realizing this requires addressing challenges unique to 3DGS that do not arise with mesh-based representations.

Technical challenges. First, 3DGS provides no explicit surfaces, floor planes, or signed distance fields, so standard scene-aware motion methods that query mesh geometry cannot be directly applied. Second, RL locomotion policies trained on clean synthetic meshes must operate on noisy, incomplete Gaussian reconstructions with floaters and no topology. Third, synthesizing fine-grained interactions requires a differentiable distance-to-scene function for collision avoidance—meshes provide this via SDFs, but Gaussians do not. Fourth, enforcing human–scene contact cannot rely on surface projection or direct pose optimization: unlike mesh avatars, where

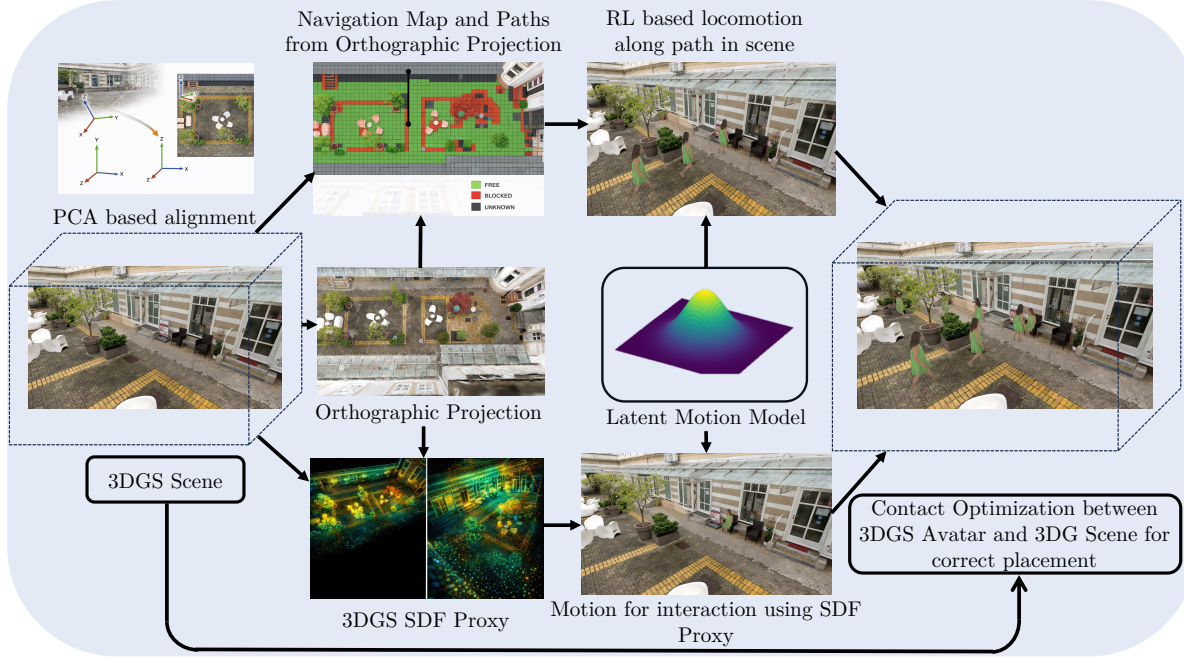


Figure 8.2: **Method overview.** Given a 3DGS scene, we (1) extract navigable structures via PCA-based alignment and orthographic projection to build navigation maps and A* paths (Sec. 8.3.1); (2) synthesize locomotion along the path using an RL policy adapted to 3DGS (Sec. 8.3.2); (3) generate fine-grained interactions (e.g., sitting) via latent-space optimization using a differentiable SDF proxy derived from scene Gaussians (Sec. 8.3.3); and (4) refine contact between the 3DGS avatar and scene for correct placement (Sec. 8.3.4). All stages operate directly on Gaussians without meshes.

LBS provides a direct linear mapping from pose to vertices, 3DGS avatars require a forward pass through a learned network that maps pose to Gaussians, making back-propagation intractable for contact optimization.

Our approach. We address these challenges with four components. **(1) Navigable structures from 3DGS.** We extract obstacle maps from raw Gaussian fields via opacity culling and PCA-aligned projection, enabling A* pathfinding directly on Gaussians without meshes (Sec. 8.3.1). **(2) Gaussian-aligned locomotion.** We adapt RL-based locomotion to 3DGS scenes, introducing an egocentric walkability grid extracted from scene Gaussians to provide the local obstacle context the RL policy requires at each step. The text-conditioned motion backbone enables diverse locomotion styles (walking, jumping, twirling) via the text cue (Sec. 8.3.2). **(3) Interaction synthesis via differentiable SDF proxy.** We define a differentiable SDF proxy as nearest-neighbor distance to scene Gaussians and use it as the collision signal for deterministic latent-space optimization, synthesizing fine-grained actions (sitting, grasping) directly in Gaussian space (Sec. 8.3.3). **(4) Differentiable contact refinement in 3DGS.** We optimize per-Gaussian translation offsets to enforce correct contact between human and scene Gaussians (Sec. 8.3.4, Fig. 8.5), using soft nearest-neighbor distances in lieu of explicit surfaces. These composed Gaussians can be rendered from any camera viewpoint to produce photorealistic videos of humans interacting with diverse scenes—

including scenes reconstructed from casually captured mobile-phone footage, enabling applications such as gaming and AR/VR content creation in environments captured with consumer cameras. Fig. 8.2 provides an overview.

8.3.1 From 3DGS to Navigable and Proxy Structures

A prerequisite for scene-aware motion synthesis is a navigable scene representation: obstacle maps, pathfinding graphs, and distance fields. For meshes, these are readily available via navigation meshes and signed distance fields. 3DGS provides none of these—only a cloud of Gaussian primitives with positions, opacities, and scales. We introduce a pipeline that extracts navigation-ready representations from raw Gaussian fields, enabling deployment of RL-based locomotion policies trained on mesh environments.

Gravity-aligned top-down projection. Given scene Gaussians $\mathcal{G}^S$, we first establish a gravity-aligned coordinate frame. We compute PCA of the Gaussian centers $\{\mu_i\}_{i=1}^{N_S}$ to identify the dominant axes of the scene. The smallest principal component is taken as the vertical (gravity) direction, and the remaining two define the ground plane. This yields a rotation that aligns the scene to a canonical top-down view without requiring camera metadata or manual annotation.

Opacity culling. A naive orthographic projection of all Gaussians produces a cluttered map dominated by floaters—low-opacity Gaussians that arise from reconstruction noise and do not correspond to physical surfaces. We filter these by thresholding: only Gaussians with opacity $\alpha_i > \tau_\alpha$ are retained, where τ_α is an opacity threshold. Without it, phantom obstacles block viable paths and the RL policy’s goal-reaching rate drops from 95.2% to 42.6% (Sec. 8.5).

Obstacle map and pathfinding. After culling, we render the retained Gaussians into a binary top-down obstacle map $\mathcal{O} \in \{0, 1\}^{W_m \times H_m}$ via orthographic projection, where $W_m \times H_m$ is the map resolution and cells with projected Gaussian density above a threshold are marked as obstacles. We then run A* pathfinding on $\mathcal{O}$ to compute collision-free waypoint sequences between start and goal positions.

Differentiable SDF proxy. Since 3DGS has no signed distance field, we approximate one via differentiable nearest-neighbor distance to scene Gaussians: for a query point $\mathbf{x}$ (e.g., a sampled SMPL body point), we compute $\rho(\mathbf{x}) = \min_i \|\mathbf{x} - \mu_i\| - r_i$, where $r_i = \max(s_{x,i}, s_{y,i}, s_{z,i})$ is the effective radius of scene Gaussian i , defined as the largest component of its scale vector $\mathbf{s}_i$. Positive values indicate free space; negative values indicate penetration. This proxy provides the collision loss $\mathcal{L}_{\text{coll}}$ used in interaction synthesis (Sec. 8.3.3). For contact refinement (Sec. 8.3.4), we introduce a soft variant of this distance that provides smoother gradients suitable for pulling Gaussians onto surfaces.

8.3.2 Gaussian-Aligned Locomotion

We introduce a Gaussian-aligned locomotion module that synthesizes controllable human navigation directly in 3DGS scenes. The central challenge is that existing

scene-aware motion methods [Zhao et al., 2023, Hassan et al., 2021a] rely on mesh geometry: they query floor planes for pathfinding, signed distance fields for collision, and surface normals for contact—none of which exist in a 3DGS reconstruction. Our solution derives reliable scene cues from the navigable structures extracted in Sec. 8.3.1, enabling RL-based locomotion in Gaussian scenes without meshes.

Design overview. We adopt a strong latent motion backbone trained on large-scale mocap data and build an RL locomotion policy on top of it that navigates between waypoints while avoiding scene Gaussians. The policy is trained in synthetic environments but deployed in noisy 3DGS scenes. This decomposition is effective in 3DGS settings: much of the raw scene detail can be abstracted to a set of *paths* for navigation, thus enabling scene-aware motion synthesis without human–scene paired data. The policy operates in the latent space of a learned motion model [Zhao et al., 2025a].

Latent motion backbone. We adopt a latent motion prior following [Zhao et al., 2025a], trained on AMASS [Punnakkal et al., 2021, Mahmood et al., 2019]. A transformer VAE learns a compact motion-primitive space: given motion history $\mathbf{H}$ and a future segment $\mathbf{X}$, an encoder produces a latent code $\mathbf{z}$, and a decoder reconstructs motion as $\hat{\mathbf{X}} = \mathcal{D}(\mathbf{H}, \mathbf{z})$. A diffusion prior is placed in this latent space, with a denoiser $\mathcal{F}$ that predicts the clean code from noisy input:

$$\hat{\mathbf{z}}_0 = \mathcal{F}(\mathbf{z}_\tau, \tau, \mathbf{H}, \mathbf{e}), \quad (8.6)$$

where τ is the diffusion timestep and $\mathbf{e}$ an optional text embedding. At inference, we sample $\mathbf{z}_{\tau_{\max}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, denoise via Eq. (8.6) in ~ 10 steps, and decode $\hat{\mathbf{X}} = \mathcal{D}(\mathbf{H}, \hat{\mathbf{z}}_0)$ to produce a short motion clip. The history $\mathbf{H}$ is then updated autoregressively.

Scene-adapted RL locomotion in 3DGS. Using the navigable structures from Sec. 8.3.1, we cast locomotion as an MDP whose *action space* is the latent space of the motion model. At each step i , the policy π_ω observes:

$$s_i = (\mathbf{H}_i, \mathbf{g}_i, \mathcal{W}_i, \mathbf{e}_i), \quad (8.7)$$

comprising motion history, a goal cue, the egocentric walkability grid (defined below), and a text embedding. The text cue defaults to “walk” but can be set to other motion styles (e.g., “jump”, “twirl”) to synthesize diverse locomotion beyond standard walking. The policy outputs a start-noise $\mathbf{z}_{\text{RL},i}^{(\tau_{\max})}$, which frozen $\mathcal{F}$ and $\mathcal{D}$ map to a motion clip via Eq. (8.6), ensuring stable and diverse rollouts. Rewards encourage waypoint progress, obstacle avoidance, and kinematic plausibility.

Egocentric walkability grid. The RL policy consumes an egocentric occupancy grid $\mathcal{W} \in \{0, 1\}^{N_g \times N_g}$ centered on the agent’s pelvis, where N_g is the grid resolution. For each cell u , we compute its nearest-neighbor distance $d(u)$ to opacity-filtered Gaussian centers and classify:

$$\mathcal{W}(u) = \begin{cases} 1 \text{ (walkable)} & \text{if } d(u) > \tau_{\text{occ}}, \\ 0 \text{ (blocked)} & \text{otherwise,} \end{cases} \quad (8.8)$$

where τ_{occ} is a distance threshold. This grid provides the local obstacle context that the RL policy requires at each step, replacing the navigation-mesh queries used during training in synthetic environments.

8.3.2.1 Latent Motion Backbone Details

The transformer VAE from [Zhao et al., 2025a] learns a compact motion-primitive space from AMASS [Punnakkal et al., 2021, Mahmood et al., 2019]. Given motion history $\mathbf{H}$ and a future motion segment $\mathbf{X}$, the encoder $\mathcal{E}$ outputs a Gaussian posterior

$$q_\phi(\mathbf{z} \mid \mathbf{H}, \mathbf{X}) = \mathcal{N}(\boldsymbol{\nu}, \sigma^2 \mathbf{I}),$$

where $\boldsymbol{\nu}$ is the posterior mean and σ is the posterior standard deviation, with reparameterized sample $\mathbf{z} = \boldsymbol{\nu} + \sigma \odot \boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The decoder $\mathcal{D}$ reconstructs motion as $\hat{\mathbf{X}} = \mathcal{D}(\mathbf{H}, \mathbf{z})$. The diffusion prior operates in this latent space with forward process

$$q(\mathbf{z}_\tau \mid \mathbf{z}_{\tau-1}) = \mathcal{N}(\sqrt{1 - \beta_\tau} \mathbf{z}_{\tau-1}, \beta_\tau \mathbf{I}),$$

where β_τ is the noise schedule at timestep τ . The denoiser $\mathcal{F}$ predicts the clean code $\hat{\mathbf{z}}_0 = \mathcal{F}(\mathbf{z}_\tau, \tau, \mathbf{H}, \mathbf{e})$, where $\mathbf{e}$ is an optional text embedding.

8.3.2.2 Policy and Actions

We train locomotion using PPO [Schulman et al., 2017] with an actor–critic architecture; both networks are 4-layer MLPs (width 512) with residual connections. The policy’s output corresponds to the diffusion start-noise $\mathbf{z}_{\text{RL},i}^{(\tau_{\max})}$ in the latent space of the motion model, parameterized as

$$\mathbf{z}_{\text{RL},i}^{(\tau_{\max})} = 4 \tanh(\mathbf{z}_{\text{raw},i}^{(\tau_{\max})}) \in [-4, 4]^{d_z}, \quad (8.9)$$

where d_z is the latent dimension and $\mathbf{z}_{\text{raw},i}^{(\tau_{\max})}$ is the unconstrained vector output of the policy network before squashing. This bounded parameterization stabilizes exploration and ensures consistency with the pretrained denoiser–decoder $(\mathcal{F}, \mathcal{D})$, which maps $\mathbf{z}_{\text{RL},i}^{(\tau_{\max})}$ into short motion clips.

8.3.2.3 Observations

At each step i , the policy observes the state $s_i = (\mathbf{H}_i, \mathbf{g}_i, \mathcal{W}_i, \mathbf{e}_i)$ as defined in Eq. (8.7): motion history $\mathbf{H}_i$, a goal cue $\mathbf{g}_i$ (clamped to a 5 m egocentric horizon and 120° field of view), a body-centric *egocentric walkability grid* $\mathcal{W}_i \in \{0, 1\}^{N_g \times N_g}$ ($N_g=16$) covering a 1.6×1.6 m region around the pelvis, and a text cue $\mathbf{e}_i$ (defaulting to “walk” but configurable to other motion styles such as “jump” or “twirl”). Each cell encodes walkability: $\mathcal{W}_i(u) = 1$ if the cell is traversable, and $\mathcal{W}_i(u) = 0$ if blocked by obstacles. During training in synthetic mesh-based rooms, occupancy is computed from the navigation mesh and object collisions; during deployment in 3DGS, it is derived from projected, opacity-thresholded Gaussians as described in Sec. 8.3.1.

8.3.2.4 Reward Design

Let $\mathbf{p}_i \in \mathbb{R}^3$ denote the pelvis position at step i and $\mathbf{g} \in \mathbb{R}^3$ the goal position in world coordinates. With planar distance $d_i = \|(\mathbf{p}_i - \mathbf{g})_{xy}\|_2$, we shape progress and success as

$$r_{\text{dist}}^{\text{prog}} = d_{i-1} - d_i, \quad r_{\text{succ}} = \mathbf{1}[d_i < 0.3]. \quad (8.10)$$

Orientation alignment is encouraged via

$$r_{\text{ori}} = \frac{1}{2}(\langle \mathbf{p}_i - \mathbf{p}_{i-1}, \mathbf{g} - \mathbf{p}_{i-1} \rangle + 1), \quad (8.11)$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product of the normalized direction vectors. Kinematic plausibility is promoted by penalizing foot skating and floor contact errors:

$$r_{\text{skate}} = -v_f^{(2-2h/\tau_h)}, \quad r_{\text{floor}} = -(h_f - \tau_f)_+, \quad (8.12)$$

where v_f is the horizontal foot displacement magnitude, h is the foot height above the ground, $\tau_h=0.03$ m is a height threshold, h_f is the height of the lowest foot vertex, $\tau_f=0.03$ m is a floor contact tolerance, and $(\cdot)_+$ clips negative values to zero. The exponent interpolates the skating penalty: when the foot is on the ground ($h=0$) the penalty is $-v_f^2$, and it vanishes as $h \rightarrow \tau_h$. Obstacle avoidance is realized by discouraging overlap between the walkability grid $\mathcal{W}_i$ and the body footprint $\mathcal{B}_{xy}(\mathbf{X})$:

$$r_{\text{pene}} = \exp\left(-|\mathcal{W}_{i,0} \cap \mathcal{B}_{xy}(\mathbf{X})|\right), \quad (8.13)$$

where $\mathcal{W}_{i,0}$ are the non-walkable cells and $|\cdot|$ counts the number of overlapping cells.

The total reward is

$$r_i = w_{\text{dist}} r_{\text{dist}}^{\text{prog}} + w_{\text{succ}} r_{\text{succ}} + w_{\text{ori}} r_{\text{ori}} + w_{\text{skate}} r_{\text{skate}} + w_{\text{floor}} r_{\text{floor}} + w_{\text{pene}} r_{\text{pene}}. \quad (8.14)$$

8.3.2.5 Training Environments

We procedurally generate rectangular rooms (edge 2–7 m) filled with ShapeNet furniture (chairs, beds, sofas, desks, tables) with z-up alignment and expanded margins to preserve traversability. Navigation meshes yield collision-free start/goal pairs and waypoint paths, and randomized initial poses/heading encourage robustness. These synthetic environments, together with the reward terms in Eqs. (8.10)–(8.14), teach policies to reach goals while avoiding obstacles and maintaining physically plausible motion.

8.3.2.6 Autoregressive Rollout

The rollout algorithm, originally introduced in [Zhao et al., 2025a], is used in our transition synthesis framework. It generates motion sequences by iteratively sampling and denoising latent codes:

Algorithm 4: Autoregressive rollout generation using latent motion primitive model.

Input: Primitive decoder $\mathcal{D}$, latent denoiser $\mathcal{F}$, motion seed $\mathbf{H}_{\text{seed}}$, text prompts $\mathbf{e} = [e_1, \dots, e_N]$, diffusion steps τ_{max} , classifier-free guidance scale w , sampler S

Input: (Optional) Latent noises $\mathbf{Z}_{\tau_{\text{max}}} = [z_{\tau_{\text{max}}}^1, \dots, z_{\tau_{\text{max}}}^N]$

Output: Motion sequence $\mathbf{M}$

$\mathbf{H} \leftarrow \mathbf{H}_{\text{seed}}, \mathbf{M} \leftarrow \mathbf{H}_{\text{seed}};$

for $i \leftarrow 1$ **to** N **do**

;

if *latent noise not provided* **then**

Sample $z_{\tau_{\text{max}}}^i \sim \mathcal{N}(\mathbf{0}, \mathbf{I});$

else

Use provided $z_{\tau_{\text{max}}}^i;$

$\hat{z}_0^i \leftarrow S(\mathcal{F}, z_{\tau_{\text{max}}}^i, \tau_{\text{max}}, \mathbf{H}, e_i, w);$

;

$\hat{X} \leftarrow \mathcal{D}(\mathbf{H}, \hat{z}_0^i);$

;

$\mathbf{M} \leftarrow \text{CONCAT}(\mathbf{M}, \hat{X});$

$\mathbf{H} \leftarrow \text{CANONICALIZE}(\hat{X}_{F-H+1:F});$

;

// number of motion primitives

// DDIM/DDPM denoising

// decode latent to motion primitive

// update history

return $\mathbf{M};$

8.3.3 Interaction Synthesis via SDF Proxy

Once the RL locomotion policy (Sec. 8.3.2) brings the agent near an action point, control switches to *deterministic latent-space optimization* for fine-grained actions such as sitting or grasping. These transitions require a differentiable distance-to-scene function to penalize body–scene penetration; we use the SDF proxy $\rho(\cdot)$ from Sec. 8.3.1.

Latent optimization for transitions in 3DGS. Following [Zhao et al., 2025a], we adopt a deterministic DDIM sampler that maps a terminal noise variable $\mathbf{Z}_{\text{opt}}$ to a motion snippet $\mathbf{M} = (\mathbf{M}_1, \dots, \mathbf{M}_f)$ of f frames, conditioned on a seed history $\mathbf{H}_{\text{seed}}$ and a text cue $\mathbf{e}$ (e.g., “sit”). Starting from the locomotion end state $\mathbf{M}_{\text{end}}^{\text{loc}}$, we optimize $\mathbf{Z}_{\text{opt}}$ via gradient descent on:

$$\mathcal{L} = \mathcal{L}_{\text{reach}} + \lambda_{\text{stop}} \mathcal{L}_{\text{stop}} + \lambda_{\text{start}} \mathcal{L}_{\text{start}} + \lambda_{\text{coll}} \mathcal{L}_{\text{coll}} + \lambda_{\text{smooth}} \mathcal{L}_{\text{smooth}}, \quad (8.15)$$

where the individual terms enforce goal reaching, stopping, continuity with locomotion, collision avoidance, and temporal smoothness:

$$\mathcal{L}_{\text{reach}} = \|\mathbf{x}_f(a^*) - \mathbf{g}\|_2^2, \quad \mathcal{L}_{\text{stop}} = \|\mathbf{v}_f(a^*)\|_2^2, \quad (8.16)$$

$$\mathcal{L}_{\text{start}} = \|\mathbf{M}_1 - \mathbf{M}_{\text{end}}^{\text{loc}}\|_2^2, \quad \mathcal{L}_{\text{coll}} = \sum_{b \in \mathcal{B}_f} [-\rho(b)]_+^2, \quad (8.17)$$

$$\mathcal{L}_{\text{smooth}} = \frac{1}{f-1} \sum_{t=2}^f \|\mathbf{M}_t - \mathbf{M}_{t-1}\|_2^2. \quad (8.18)$$

Here a^* is a task-specific anchor joint (e.g., pelvis for sitting, hand for grasping), $\mathbf{x}_f$ and $\mathbf{v}_f$ are its position and velocity at the final frame, $\mathbf{g} \in \mathbb{R}^3$ is the user-specified 3D goal point—which can be at any height, enabling grasping at different shelf levels—and $\mathcal{B}_f$ are body surface points sampled from SMPL at frame f . After completing the action, the same formulation synthesizes the exit transition (e.g., sit $\rightarrow$ walk), after which locomotion resumes. Although our method can synthesize actions that mimic “picking up” or “grabbing,” it cannot lift actual objects, as we assume the scene remains static.

8.3.4 Differentiable Contact Refinement in 3DGS

After compositing posed human Gaussians $\mathcal{G}_t^{H,P}$ with scene Gaussians $\mathcal{G}^S$, floor penetration and inconsistent contacts frequently arise (Fig. 8.5). This occurs due to mismatch in body shape and height between the SMPL mesh used for motion generation and the 3DGS avatar, since motion synthesis assumes a fixed body shape across sequences and focuses solely on pose generation. As discussed in Sec. 8.1, direct SMPL pose optimization is intractable for Gaussians—the pose-to-Gaussian mapping passes through a learned neural network rather than the linear LBS used for meshes—and no explicit surface exists for contact projection. We therefore optimize small per-Gaussian *translation offsets* $\mathbf{T}_{k,t}$, refining a subset of human Gaussians to achieve (i) contact where appropriate and (ii) separation elsewhere.

Contact detection and indexing. We detect contact frames by thresholding vertical velocity and acceleration at each SMPL joint, producing a per-joint binary indicator $\delta_{c,t}$. To determine *which* Gaussians to refine, we lift SMPL contact vertices V_c^{SMPL} (e.g., feet, hip) to canonical human Gaussians via nearest-neighbour search:

$$k^* = \arg \min_k \|\boldsymbol{\mu}_k^C - \mathbf{m}\|_2, \quad \mathbf{m} \in V_c^{\text{SMPL}}.$$

The resulting index set $\mathcal{I}_c$ is identity-agnostic: it adapts to any avatar regardless of Gaussian count.

Scene proximity in Gaussian space. While the hard SDF proxy ρ (Sec. 8.3.1) suffices for collision avoidance during interaction synthesis, contact refinement requires smoother gradients to pull Gaussians onto surfaces. We therefore define a soft nearest-neighbour distance

$$d_\kappa(\mathbf{x}) = -\frac{1}{\kappa} \log \left(\sum_{i=1}^{N_S} \exp(-\kappa \|\mathbf{x} - \boldsymbol{\mu}_i\|) \right),$$

where $\boldsymbol{\mu}_i$ are scene Gaussian centers and κ controls softness. This provides stable gradients for contact/separation without requiring explicit meshes.

Refinement objective. For a contact Gaussian $k \in \mathcal{I}_c$ at frame t with indicator $\delta_{c,t}$, we optimize a translation $\mathbf{T}_{k,t}$ and update $\tilde{\boldsymbol{\mu}}_{k,t}^P = \boldsymbol{\mu}_{k,t}^P + \mathbf{T}_{k,t}$ by minimizing

$$\mathbf{T}_{k,t}^* = \arg \min_{\mathbf{T}} \lambda_{\text{snap}} \|\boldsymbol{\mu}_{k,t}^P + \mathbf{T} - \boldsymbol{\mu}_{i(k,t)}\|_2^2 + \lambda_d \psi(\boldsymbol{\mu}_{k,t}^P + \mathbf{T}, \delta_{c,t}) + \lambda_r \|\mathbf{T}\|_2^2,$$

where $\mu_{i(k,t)}$ is the nearest scene Gaussian center and

$$\psi(\mathbf{x}, \delta) = \begin{cases} d_\kappa(\mathbf{x})^2, & \delta = 1 \text{ (enforce contact)} \\ h_r(d_\kappa(\mathbf{x}))^2, & \delta = 0 \text{ (enforce separation)} \end{cases} \quad \text{with } h_r(d) = \max(0, r - d).$$

Here $r > 0$ is the hinge margin, and $\lambda_{\text{snap}}, \lambda_d, \lambda_r$ are scalar weights for snapping, contact/separation penalty, and regularization, respectively. For temporal coherence, we add $\lambda_{\text{temp}} \sum_t \|\mathbf{T}_{k,t} - \mathbf{T}_{k,t-1}\|_2^2$, where λ_{temp} is a temporal smoothness weight. Intuitively, the objective snaps designated contact Gaussians toward nearby scene surfaces when contact is detected, pushes them away otherwise, penalizes large displacements, and smooths motion over time.

The refined human Gaussians $\tilde{\mathcal{G}}_t^{H,P}$ are composed with $\mathcal{G}^S$ and rendered with the standard 3DGS rasterizer to produce the final human–scene interactions.

8.3.5 Contact Detection

From the synthesized SMPL motion, let $\mathbf{p}_{c,t}$ denote the position of contact joint c at frame t , with velocity $\mathbf{v}_{c,t} = \mathbf{p}_{c,t} - \mathbf{p}_{c,t-1}$ and acceleration $\mathbf{a}_{c,t} = \mathbf{v}_{c,t} - \mathbf{v}_{c,t-1}$. A frame is marked as a contact frame if the joint has small vertical velocity and downward acceleration:

$$\delta_{c,t} = (|v_{c,t}^y| < \tau_v) \wedge (a_{c,t}^y < \tau_a),$$

where y denotes the vertical axis, τ_v is the velocity threshold, and τ_a is the acceleration threshold.

8.4 IMPLEMENTATION DETAILS

We describe the coordinate frame alignment and height normalization procedures used to composite avatars into reconstructed scenes, followed by a summary of all pipeline hyperparameters.

8.4.1 Coordinate Frame and Height Normalization

Scene Gaussians $\mathcal{G}^S$ and human Gaussians $\mathcal{G}^H$ are reconstructed independently and therefore live in different coordinate frames with potentially different scales and orientations. Before compositing, we align them into a shared metric coordinate frame.

8.4.1.1 Scene Coordinate Frame Alignment

3DGS scene reconstructions inherit an arbitrary coordinate frame from the underlying structure-from-motion initialization. Neither orientation nor metric scale is guaranteed. We normalize the scene coordinate frame in two steps.

Gravity alignment via PCA. We compute PCA of the scene Gaussian centers $\{\mu_i\}_{i=1}^{N_S}$ and assign the smallest principal component as the vertical (gravity) axis. Let $\mathbf{R}_{\text{PCA}} \in \mathbb{R}^{3 \times 3}$ denote the resulting rotation. We apply $\mathbf{R}_{\text{PCA}}$ to all scene Gaussian centers and covariances:

$$\mu'_i = \mathbf{R}_{\text{PCA}} \mu_i, \quad \Sigma'_i = \mathbf{R}_{\text{PCA}} \Sigma_i \mathbf{R}_{\text{PCA}}^\top.$$

After this step, the y -axis is aligned with gravity and the xz -plane defines the horizontal ground plane.

Ground plane estimation. We estimate the ground plane height y_{floor} from the aligned scene Gaussians. Among Gaussians retained after opacity culling ($\alpha_i > \tau_\alpha$), we take the lower percentile of vertical positions:

$$y_{\text{floor}} = \text{percentile}_p(\{y'_i : \alpha_i > \tau_\alpha\}),$$

where p is a low percentile (we use $p=5$) to be robust to sub-floor outliers. All scene Gaussian centers are then shifted so that the ground sits at $y=0$: $\mu''_i = \mu'_i - (0, y_{\text{floor}}, 0)^\top$.

8.4.1.2 Avatar Height Normalization

SMPL-based avatars are defined in metric space (meters) via the body model, but scene Gaussians may be at an arbitrary scale—particularly for monocular reconstructions where absolute scale is unobservable.

Reference height from scene. We estimate a scene-specific metric scale factor by measuring the vertical extent of representative scene structures. When known reference objects are available (e.g., doors, standard furniture heights), we compute the scale factor as $\eta = h_{\text{ref}}/h_{\text{scene}}$, where h_{ref} is the known metric height and h_{scene} is the measured height in the scene’s coordinate system. For scenes from ScanNet++ [Yeshwanth et al., 2023], which are reconstructed at metric scale, $\eta=1$. For scenes from other sources (e.g., SuperSplat [SuperSplat], DL3DV [Ling et al., 2024]), we estimate η from scene context.

Avatar placement. Given the normalized scene and scale factor η , the avatar’s root translation at each frame t is set so that the lowest SMPL vertex (typically the foot sole) coincides with the ground plane:

$$\mathbf{t}_t^y = y_{\text{floor}} + \Delta y_t,$$

where $\mathbf{t}_t^y$ is the vertical component of the SMPL root translation and Δy_t is the height of the lowest vertex above the pelvis origin in the current pose. This ensures the avatar’s feet rest on the estimated ground plane prior to the contact refinement stage (Sec. 8.3.4), which further adjusts individual Gaussians for precise contact.

8.4.2 Hyperparameters

Tab. 8.1 lists the hyperparameters for all pipeline stages.

Table 8.1: **Hyperparameters for all pipeline stages.** Top: RL locomotion reward weights (Eq. (8.14)). Middle: interaction latent-space optimization weights (Eq. (8.15)). Bottom: contact refinement weights (Sec. 8.3.4).

Stage / Symbol	Description	Value
<i>RL locomotion rewards</i>		
w_{dist}	distance progress	1.0
w_{succ}	goal success	1.0
w_{ori}	orientation alignment	0.1
w_{skate}	foot skating penalty	100.0
w_{floor}	floor contact penalty	100.0
w_{pene}	scene penetration	1.0
<i>Interaction optimization</i>		
λ_{stop}	velocity at goal	1.0
λ_{start}	continuity with locomotion	1.0
λ_{coll}	collision avoidance	0.1
λ_{smooth}	temporal smoothness	0.1
learning rate	Adam step size	0.01
optimization steps	gradient iterations	100
guidance scale w	classifier-free guidance	5.0
diffusion steps	DDIM sampling steps	10
<i>Contact refinement</i>		
τ_v	contact velocity threshold (m/s)	0.01
τ_a	contact acceleration threshold (m/s ²)	0.5
λ_{snap}	snap to nearest scene Gaussian	1.0
λ_d	contact / separation penalty	0.1
λ_r	translation regularization	0.01
λ_{temp}	temporal smoothness	0.1
r	hinge margin (m)	0.05
κ	soft-NN temperature	100

The hyperparameters in Tab. 8.1 were selected via grid search on a held-out ScanNet++ validation scene and kept fixed across all experiments. The RL reward weights w_{skate} and w_{floor} are set substantially higher than the remaining terms because foot skating and floor penetration are the most perceptually salient artifacts in human-scene animation—even small violations are immediately noticeable to viewers. The interaction optimization uses a modest number of gradient steps (100) with classifier-free guidance, which we found sufficient for convergence given the smooth latent space of the diffusion prior. For the contact refinement stage, the soft nearest-neighbor temperature $\kappa=100$ produces a sharp approximation to the true nearest-neighbor distance while remaining differentiable, and the hinge margin $r=0.05$ m prevents the optimization from pulling distant Gaussians toward the scene surface when no contact is intended.

8.5 EXPERIMENTS

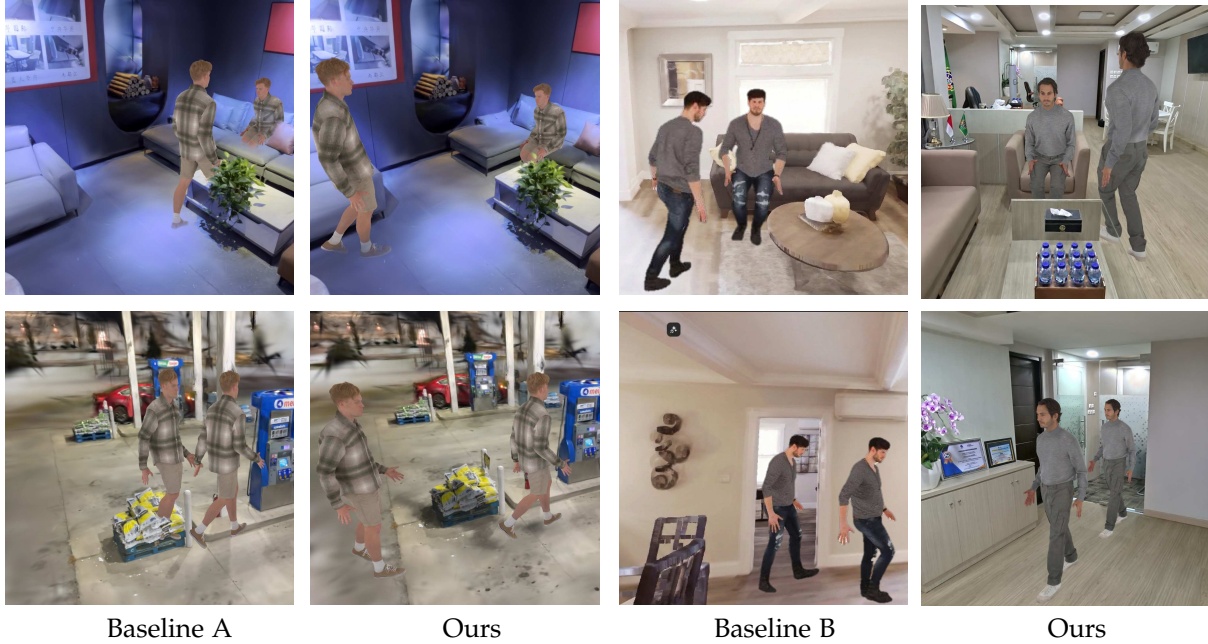


Figure 8.3: **Visual comparison with baselines.** Left pair (motion evaluation): Baseline A uses a VGGT [Wang et al., 2025a] mesh from monocular video, producing collisions due to artifacts; our motion synthesis method entirely relies on 3DGS and correctly generates plausible motion in 3D scenes. Right pair (rendering evaluation): Baseline B uses the highest-quality mesh pipeline (Replica [Straub et al., 2019] + RenderPeople [RenderPeople, 2025]); our 3DGS result achieves more photorealistic appearance. Humans prefer our renderings in both settings (Tab. 8.2).

We evaluate our method along two axes: *rendering quality*—does 3DGS produce more photorealistic human–scene compositions than meshes (Sec. 8.5.1)?—and *motion quality*—does our Gaussian-aligned pipeline (Secs. 8.3.1–8.3.4) synthesize plausible, collision-free motion (Sec. 8.5.2)?

8.5.1 Rendering Evaluation

We compare against two baselines using human judgments and VLM assessments.

Baselines. *Baseline A (within-scene, monocular).* Each DL3DV [Ling et al., 2024] scene is reconstructed as both 3DGS and a mesh via VGGT [Wang et al., 2025a]. Motion is synthesized on the mesh using [Zhao et al., 2023], then naively composited with the 3DGS scene for rendering—without our walkability, interaction, or refinement modules. This tests whether naive Gaussian compositing suffices for monocular videos. *Baseline B (cross-scene, HQ vs HQ).* We pit the highest-quality mesh rendering—Replica [Straub et al., 2019] scenes with a RenderPeople [RenderPeople, 2025] scan animated by [Zhao et al., 2023]—against our best 3DGS rendering (SuperSplat scenes,



Figure 8.4: **Effect of the egocentric walkability grid (Sec. 8.3.2).** **Left:** without the walkability map, the RL policy lacks local obstacle awareness and generates motion that passes through the desk. **Right:** with the walkability grid extracted from scene Gaussians, the avatar correctly avoids the obstacle.

Table 8.2: **Rendering preference (win rate, %).** Higher = our 3DGS preferred.

Evaluator	Bas. B	Bas. A
	HQ vs HQ	Mono.
Human study	82.1	72.9
GPT-5	75.2	71.8
Gemini 2.5	69.1	65.9

AvatarReX [Zheng et al., 2023] avatars). This experiment compares the highest-quality rendering achievable with mesh-based methods against the highest-quality rendering achievable with 3DGS-based methods. Within-scene comparison is not feasible for Baseline B because the two modalities require fundamentally different capture setups: mesh-based avatars (RenderPeople) are reconstructed from ~ 250 DSLR cameras in a single frame, whereas 3DGS avatars require ~ 120 s of multi-view video from as few as 4–16 cameras; similarly, Replica scenes are reconstructed from dense depth sensors, while SuperSplat scenes use different capture modalities. Since neither the original Replica images nor the RenderPeople capture data can be used to reconstruct 3DGS counterparts, comparisons are necessarily cross-scene.

Protocols. *Protocol I (human study):* 21 participants perform pairwise forced-choice judgments on 5 video pairs per baseline, selecting the more photorealistic rendering. *Protocol II (VLM study):* GPT-5 [OpenAI, 2025] and Gemini 2.5 [Comanici et al., 2025] judge 10 matched frame pairs per video for photorealism. For the pairwise VLM study, the VLM is instructed to ignore artistic style and focus on physical plausibility: “Which image appears more photorealistic? Consider geometry (straight lines, depth cues), materials (BRDF, speculars), lighting/shadows, and absence of artifacts (flicker, halos, floaters). Respond with A, B, or Tie.”

Results. Tab. 8.2 reports win rates across both protocols. On the HQ-vs-HQ comparison (Baseline B), human participants prefer our 3DGS renderings 82.1% of the time over the best available mesh pipeline—confirming the central premise that *neural scene representations yield superior rendering quality for human–scene animation* (Fig. 8.3).

Table 8.3: **Ablation study on ScanNet++**. Motion synthesized using 3DGS; metrics evaluated in mesh space.

Method	Skating↓	Penetr.↓	Jerk↓	Goal↑
Ours (full)	0.89	0.012	0.47	95.2
w/o opacity culling	1.45	0.042	0.78	42.6
w/o walkability map	1.37	0.021	0.71	84.5
w/o contact refinement	1.11	0.016	0.55	91.3

On the within-scene monocular comparison (Baseline A), our full pipeline is preferred 72.9% of the time; the mesh baseline suffers from noisy VGGT reconstructions that produce blocked paths and penetration artifacts. VLM judgments corroborate both trends. Notably, all three evaluators—human raters, GPT-5, and Gemini 2.5—rank the two baselines in the same relative order, with Baseline B (HQ mesh pipeline) being the stronger competitor. This consistency across human and automated judges suggests that the preference signal is robust and not an artifact of a particular evaluation protocol. The gap between human and VLM scores on Baseline B (82.1% vs. 69.1–75.2%) indicates that human raters are more sensitive to the subtle rendering advantages of 3DGS—such as view-dependent specular highlights and fine geometric detail—than current VLMs, which tend to focus on coarse structural plausibility.

8.5.2 Motion Quality and Ablations

Having established the rendering advantage of 3DGS over meshes, we now evaluate whether our pipeline produces plausible, collision-free motion despite operating entirely on Gaussian representations. We first define the motion quality metrics, then present an ablation study that isolates the contribution of each component—opacity culling, the walkability grid, and contact refinement—and finally compare motion quality against Baseline A.

8.5.2.1 Motion Metrics

We evaluate using the following metrics:

1. **Foot Contact Score.** Following prior work, we compute the scaled foot skating when in contact with the floor [Zhang et al., 2018a]: The foot velocities are added if the foot height h is within a maximum threshold of $H = 2.5$ cm. Each velocity magnitude v in the horizontal plane is further weighted by an exponential interpolation

$$s = v \left(2^{-2\frac{h}{H}} \right),$$

to estimate the amount of skating s during motion, where the exponent is clamped between 0 and 1.



Figure 8.5: **Effect of SDF proxy (Sec. 8.3.3).** **Left:** without the SDF proxy, interactions generated in 3D scenes penetrate through the 3DGS scene. **Right:** with our SDF proxy, the avatar’s hips rest on the chair and feet maintain floor contact.

2. **Interaction Penetration.** For each frame, the penetration score is defined as

$$s_{\text{inter_pene}} = \sum_{v_i \in V} |(\Psi_O(v_i))_-|, \quad (8.19)$$

where Ψ_O is the signed distance field (SDF) of the scene objects, $(\cdot)_-$ clips all positive values to zero, and V denotes body vertices. We report both average penetration over time and maximum penetration within a sequence.

3. **Jerk.** To capture temporal smoothness, we compute the *jerk* as the third derivative of the body joint trajectories, averaged over time and joints.
4. **Goal Reaching.** We measure the success rate of whether the synthesized motion successfully reaches the designated goal position—the goal is reached if the final pose is within a threshold of the user-specified goal location.

Ablation study. To isolate each component’s contribution, we evaluate on ScanNet++ [Yeshwanth et al., 2023] where both 3DGS and mesh reconstructions are available. Motion is synthesized using 3DGS; metrics are evaluated against mesh geometry. Tab. 8.3 reports these four metrics. Opacity culling has the largest impact: without it, floaters create phantom obstacles that reduce goal reaching from 95.2% to 42.6% and more than triple penetration (Fig. 8.8). Removing the walkability map degrades goal reaching to 84.5% and increases penetration, as the RL policy lacks local obstacle awareness. Fig. 8.4 visualizes the effect: without the egocentric walkability grid, the agent attempts infeasible paths through obstacles. This failure is particularly pronounced in cluttered environments such as offices and living rooms, where narrow gaps between furniture require fine-grained spatial reasoning that the global A* path alone cannot provide. The egocentric grid, extracted at each step from the surrounding scene Gaussians, gives the policy a local occupancy signal that complements the coarse global path and allows the avatar to make reactive adjustments to avoid nearby obstacles. Without contact refinement, penetration and skating increase moderately, confirming its role in enforcing plausible human–scene contact (see also Fig. 8.5). The ablation reveals a clear hierarchy among our components: opacity culling is a prerequisite for all downstream stages, as unfiltered floaters corrupt both the obstacle map and the walkability grid, cascading into navigation failures. The walkability grid provides the



Figure 8.6: **Free-viewpoint rendering of monocular videos with inserted animated humans.** Each scene is a DL3DV [Ling et al., 2024] video reconstructed as 3DGS; the human avatar is animated by our pipeline and composited in Gaussian space, enabling consistent rendering from novel viewpoints. **Left:** indoor showroom. **Right:** outdoor park.

RL policy with the local geometric context it needs to plan feasible trajectories around furniture and walls, while contact refinement handles the finer-grained problem of ensuring physically plausible foot placement and body–surface contact at interaction sites. Together, these three components form a complementary pipeline where each stage addresses a distinct failure mode of operating on raw Gaussian fields.

Motion comparison with Baseline A. We additionally compare motion quality against Baseline A. Both methods are evaluated against VGGT meshes: our motion is synthesized from 3DGS, while Baseline A uses VGGT meshes directly. As shown in Tab. 8.4, our method achieves substantially better motion despite operating on Gaussians—Baseline A suffers from noisy meshes reconstructed from monocular video, leading to $3\times$ higher skating and $6\times$ higher penetration. This result demonstrates that the geometric structures extracted from 3DGS—obstacle maps, walkability grids, and SDF proxies—are more reliable for motion synthesis than meshes reconstructed from monocular video via feed-forward methods.

8.5.3 Qualitative Results

Fig. 8.7 shows results across diverse scenes (SuperSplat library, ScanNet++) and avatar sources (ActorsHQ [Işık et al., 2023], BEHAVE [Bhatnagar et al., 2022] with only 4 cameras, DNA-Rendering [Cheng et al., 2023], AvatarReX [Zheng et al., 2023]). The results demonstrate that our pipeline generalizes across different reconstruction qualities and capture conditions, from sparse 4-camera setups to dense 48-camera rigs, producing plausible human–scene interactions in all cases. In Fig. 8.6, we demonstrate free-viewpoint rendering of monocular RGB videos edited with geometry-consistent animated humans—a capability uniquely enabled by 3DGS scene reconstruction. These results highlight the practical applicability of our framework, enabling content creators to insert animated characters into casually captured videos and render them from arbitrary viewpoints. Additional qualitative results across further indoor and outdoor 3DGS environments are shown in Fig. 8.10.

Table 8.4: **Motion comparison: monocular setting.** Our 3DGS-based motion vs Baseline A (VGGT mesh); evaluated against VGGT meshes. Despite operating entirely on Gaussians without explicit mesh geometry, our method achieves substantially lower skating, penetration, and jerk while reaching goals more reliably.

Method	Skating↓	Penetr.↓	Jerk↓	Goal↑
Ours (3DGS)	0.92	0.014	0.49	93.8
Baseline A (VGGT)	2.84	0.089	0.83	58.6

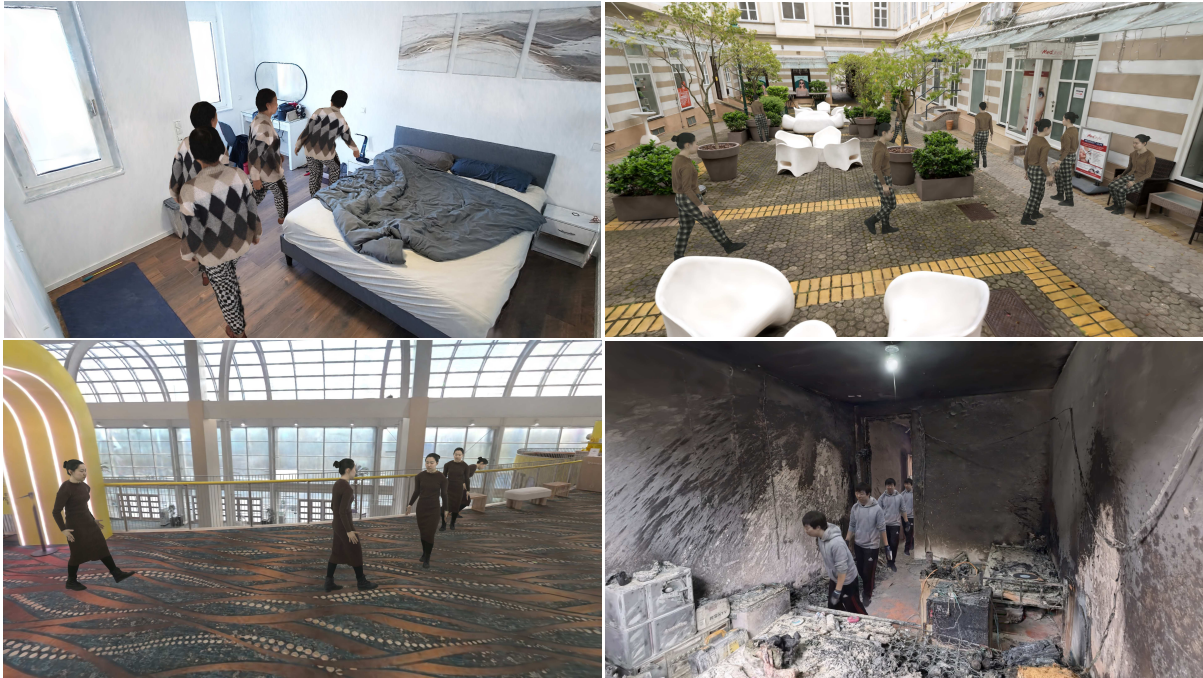


Figure 8.7: **Qualitative results across diverse environments and avatar sources.** Scenes span the SuperSplat library and ScanNet++ [Yeshwanth et al., 2023]; avatars are reconstructed from multi-view captures ranging from 4 cameras (BEHAVE [Bhatnagar et al., 2022]) to 48 cameras (ActorsHQ [Işık et al., 2023]). Our method generalizes across reconstruction quality and capture conditions.

8.5.4 Failure Modes of Baseline A

Fig. 8.9 shows that the underlying mesh recovered from VGGT contains blocked paths and significant noisy structures. This results in (1) navigation paths that penetrate through the scene, (2) interaction poses with significant penetration into the scene, and (3) unnatural motion. Note that for this baseline we do not apply any refinement in Gaussian space but use naive composition for rendering. These failure cases underscore that mesh quality from monocular feed-forward methods remains insufficient for reliable motion synthesis, motivating our approach of deriving geometric structures directly from 3DGS representations which, while implicit, benefit from multi-view consistency enforced during scene reconstruction.

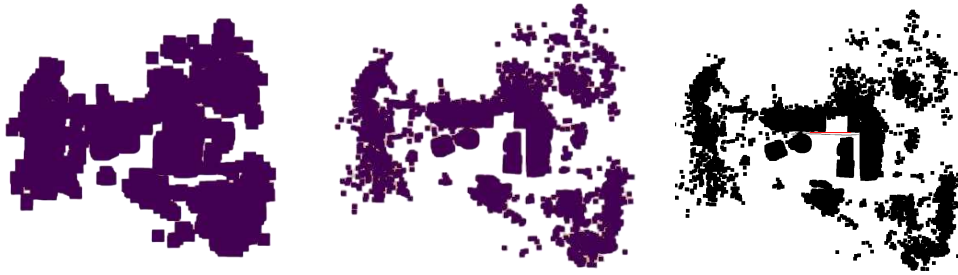


Figure 8.8: **Effect of opacity culling on obstacle maps.** (a) Naive orthographic projection of Gaussians leads to blocky structures. (b) Orthographic projection with opacity culling. (c) Path in filtered map.



Figure 8.9: **VGGT mesh reconstruction from monocular video.** The underlying mesh contains blocked paths and significant noisy structures.

8.6 CONCLUSION

We have presented, to the best of our knowledge, the first method to synthesize human interactions in diverse 3D environments using 3D Gaussian Splatting as the underlying 3D representation. Our results suggest that neural rendering is mature enough to function as a practical component in 3D human–scene animation pipelines, bridging previously disjoint lines of work in human–scene animation and neural rendering. *Crucially, our pipeline operates on scenes reconstructed from casually captured mobile-phone video*, enabling geometry-consistent free-viewpoint video editing and opening the door to gaming and AR/VR content creation in environments captured with consumer cameras. We believe this opens new research directions at the intersection of human animation and neural rendering.

Limitations. Despite this progress, our pipeline has several limitations. First, com-

plex and rapidly changing illumination can cause rendering artifacts and imperfect relighting. Second, we do not enforce full physics-based constraints, which can yield interactions that look plausible yet violate contact, stability, or momentum conservation. Third, the range of interaction types is limited; highly dexterous manipulation and multi-contact behaviors remain challenging. Fourth, we assume access to multi-view videos of a human performing diverse actions.

Outlook. Addressing these issues suggests several promising research directions: integrating stronger lighting estimation and inverse rendering, incorporating differentiable or learned physics priors for contact and dynamics, expanding the interaction vocabulary to richer, longer, and multi-person scenarios, and investigating how avatars that generalize to out-of-distribution poses can be reconstructed from monocular videos. We hope this work provides a foundation for scalable, video-native human–scene animation pipelines and catalyzes further advances in data, models, and evaluation for interactive 3D human animation.



Figure 8.10: **Additional qualitative results across diverse 3DGS environments. Top:** animated human navigating an indoor living room scene while avoiding furniture. **Bottom:** animated human navigating an outdoor courtyard, demonstrating generalization to outdoor environments with obstacles such as planters, benches, and chairs. Both results are rendered entirely within the Gaussian Splatting framework.

RAGA: REAL TIME RAY TRACED GAUSSIAN SHADOW CASTING FOR 3DGS AVATAR-SCENE INTERACTION



Figure 9.1: **RAGA casts physically plausible shadows for animated 3DGS avatars composited into 3DGS scenes.** Our ray-traced Gaussian shadow casting produces soft, temporally stable shadows entirely in Gaussian space, without any mesh reconstruction.

Chs. 7 and 8 demonstrated that 3DGS can support reconstruction, animation, and contact modeling for humans in scenes. However, compositing avatars into scenes without consistent shadows breaks visual coherence. This chapter addresses this gap. We study the problem of physically plausible shadow casting when animating 3DGS avatars, either individually or in multi-avatar and object-interaction scenarios, within existing 3DGS scenes. In contrast to prior methods that rely on binary hit tests and mesh-based shadow casters, our method performs shadow computation entirely in Gaussian space, without requiring any mesh reconstruction. We introduce RAGA, a Ray-Traced Gaussian Shadow Casting formulation based on exact ray-Gaussian line integrals. For each occluding Gaussian, we integrate the opacity profile along the shadow ray and normalize by the theoretical maximum integral, producing a weight that captures *how* the ray traverses the occluder rather than merely whether an intersection occurred. To reduce temporal variance from clothing deformations in animated avatars, we further introduce an avatar proxy representation that stabilizes

shadow casting while preserving visual fidelity. We implement RAGA using custom CUDA kernels integrated with the NVIDIA OptiX framework; as such, our shadow tracer runs at rates of about 50 FPS. We evaluate on single-avatar, multi-avatar, and avatar–object interaction scenarios across multiple datasets, demonstrating substantially improved shadow realism, temporal stability, and scene coherence.

9.1 INTRODUCTION

Recent advances in 3D Gaussian Splatting (3DGS) [Kerbl et al., 2023] have demonstrated that, beyond high-quality scene reconstruction, Gaussian-based representations can support compelling dynamic content, including animatable 3D human avatars embedded into complex real-world scenes [Mir et al., 2025b,a]. This opens the possibility of using 3DGS as a representation in content creation and simulation pipelines—for example, virtual production, CG compositing, and simulators for robotics and autonomous driving, which have traditionally relied on mesh-based assets.

Despite this progress, a fundamental limitation persists: physically plausible shadows are largely absent when animating 3DGS avatars interacting with static scenes or objects. As a result, inserted humans often appear visually detached from their environment, undermining realism and scene coherence—especially in dynamic settings where shadows provide critical spatial and temporal cues.

No existing method supports shadow casting at composition time—when an avatar is inserted into a new scene—in the volumetric setting where both caster (avatar) and receiver (scene) are 3D Gaussians. Tab. 9.1 summarizes these gaps.

Classical Shadow Mapping (Tab. 9.1, top block) [Lokovic and Veach, 2000, Williams, 1978, Reeves et al., 1987] is not directly applicable in this setting because, unlike meshes, 3DGS is a volumetric, opacity-based representation that lacks a well-defined inside–outside structure or explicit surface geometry. Existing shadow mapping pipelines rely on surface intersections and depth tests, assumptions that do not hold for 3DGS.

Inverse rendering methods (Tab. 9.1, middle block) [Liang et al., 2024, Jiang et al., 2024c, Gu et al., 2025, Bi et al., 2024, Chen et al., 2025a, Du et al., 2025, Zheng et al., 2026, Wu et al., 2025c] decompose scenes into materials and lighting, recovering per-scene self-shadows through baked occlusion volumes [Liang et al., 2024], ray-traced visibility [Gu et al., 2025], or learned shadow parameters [Du et al., 2025, Zheng et al., 2026]. However, these are per-scene optimizations on static geometry: they cannot handle dynamic avatars, and compositing a new avatar into a scene does not update the baked or learned shadow representation.

Ray-based splatting methods (Tab. 9.1, bottom block) [Moenne-Loccoz et al., 2024, Wu et al., 2025b, Byrski et al., 2025] perform ray–Gaussian intersection for rendering but rely on binary hit tests for shadow casting, ignoring how much of a Gaussian the ray actually traverses. Moreover, these methods assume mesh-based shadow casters; they do not address the fully volumetric setting where both caster and receiver are 3D Gaussians. Reconstructing meshes from 3DGS dynamic avatars is itself non-trivial [Ye et al., 2024] and often reduces visual fidelity.

RAGA. In this work, we address this gap by introducing a principled, fully

This normalization is the key design choice: the resulting weight captures *how* the ray traverses each occluder. A ray that merely grazes a Gaussian accumulates little opacity and receives a low weight; a ray passing through a long Gaussian along its major axis receives a high weight; and crucially, the same Gaussian crossed along its narrow direction receives a proportionally lower weight, correctly reflecting the reduced volumetric obstruction.

Table 9.1: **Capability comparison.**

	Composable	3DGS cast.	3DGS recv.	Dyn. 3DGS avat.	Temp. stab.
Shadow maps [Williams, 1978]	✓	×	×	×	✓
Deep shadow [Lokovic and Veach, 2000]	✓	×	×	×	✓
GS-IR [Liang et al., 2024]	×	×	✓	×	×
IRGS [Gu et al., 2025]	×	✓	✓	×	×
GS ³ [Bi et al., 2024]	×	✓	✓	×	×
3DGRT [†] [Moenne-Loccoz et al., 2024]	✓	×	✓	×	×
RaySplat [†] [Byrski et al., 2025]	✓	×	✓	×	×
Ours (RAGA)	✓	✓	✓	✓	✓

[†] Can be adapted for use with 3DGS shadow casters.

Gaussian-based solution to shadow casting for animated 3DGS avatars and objects (Fig. 9.1). We propose RAGA, a Ray-Traced Gaussian Shadow Casting formulation that computes shadows using only Gaussian particles as rendering primitives, without requiring any mesh reconstruction or hybrid representation.

Our method constructs a ray-traced shadow map by shooting rays from scene Gaussians toward light sources and modeling light attenuation induced by occluding Gaussians. For each occluder, we compute exact entry and exit points where the shadow ray intersects the Gaussian’s support, integrate its opacity profile between those bounds—which admits a closed-form solution via the error function—and normalize by the theoretical maximum integral for that Gaussian.

Adaptation of ray-based methods such as 3DGRT [Moenne-Loccoz et al., 2024, Wu et al., 2025b], or RaySplat [Byrski et al., 2025] for shadow casting with 3D Gaussian shadow casters fails: their binary hit tests treat all intersections equally regardless of traversal depth, and 3DGRT’s icosahedron proxy for each Gaussian introduces shadow artifacts due to the geometric mismatch with the smooth volumetric Gaussian. Our exact line-integral formulation eliminates these issues.

Temporal stability. To further reduce temporal variance caused by high-frequency clothing deformations in animated avatars, we introduce an avatar proxy representation that approximates pose-dependent changes in avatar Gaussians with isotropic Gaussians that preserve most of the geometric structure and reduce temporal variation in the avatar Gaussians. This proxy significantly reduces temporal jitter for moving avatars interacting with the environment.

We implement RAGA using custom CUDA kernels integrated with the NVIDIA OptiX framework, leveraging hardware ray tracing cores on modern GPUs; our shadow tracer alone runs at interactive rates of about 50 FPS. We evaluate our method on multiple datasets, including single-avatar, multi-avatar, and avatar–object interaction scenarios, demonstrating substantial improvements in shadow realism, stability, and scene integration. To summarize, our contributions are as follows:

1. We introduce a fully Gaussian-space shadow casting framework for 3DGS avatars animated in 3DGS scenes, eliminating the need for mesh reconstruction.
2. We propose RAGA, a ray-traced shadow casting formulation based on exact ray–Gaussian line integrals and normalized attenuation, replacing the approximate

proxy-geometry intersections used in prior work.

3. We introduce an avatar proxy strategy that stabilizes shadows under dynamic deformations.
4. We extensively evaluate on diverse avatar and interaction scenarios, demonstrating improved realism and temporal stability over existing approaches.

9.2 METHOD

Given a 3DGS scene and one or more animated 3DGS avatars, our goal is to compute physically plausible shadows cast by the avatars onto the scene at each frame, operating entirely in Gaussian space without mesh reconstruction. Fig. 9.2 provides an overview.

Our core idea is to formulate each Gaussian’s shadow contribution as a *normalized line integral* along the shadow ray (Fig. 9.2 f). For each scene Gaussian (receiver), we cast a ray toward the light source and, for every occluding avatar Gaussian the ray encounters, we: (i) compute exact entry and exit points where the ray intersects the Gaussian’s bounded support—its χ^2 -confidence ellipsoid (Fig. 9.2 b, Sec. 9.2.2)—rather than relying on an icosahedron proxy that can produce incorrect hits (Fig. 9.2 a); (ii) integrate the Gaussian’s opacity profile between those bounds, which admits a closed-form solution via the error function (Sec. 9.2.3); and (iii) normalize the result by the theoretical maximum integral—the value obtained when a ray passes through the center of the Gaussian along its longest axis (Sec. 9.2.4). This stands in contrast to simply evaluating the peak Gaussian response (Fig. 9.2 c), which is direction-agnostic and fails for anisotropic Gaussians (Fig. 9.2 d, e).

This normalization is the key design choice: the resulting factor $\eta \in [0, 1]$ captures *how* the ray traverses the Gaussian. A ray that merely grazes an occluder produces a small line integral and thus a low factor (Fig. 9.2 d). A ray passing through a long, thin Gaussian along its major axis accumulates substantial opacity and receives a high factor. Crucially, a ray crossing the *same* Gaussian along its narrow direction receives a low factor, correctly reflecting the reduced volumetric obstruction. The factor η then modulates the Gaussian’s opacity α when updating the shadow transmittance (Sec. 9.2.5). All ray–Gaussian queries are accelerated via a BVH built over conservative bounding boxes, enabling hardware ray tracing through NVIDIA OptiX (Sec. 9.2.6). Finally, to suppress temporal flickering from the pose-dependent Gaussian generator, we introduce an avatar proxy that ensures temporally coherent shadow casting (Sec. 9.2.8). All formal propositions and proofs supporting the derivations below are provided in Sec. 9.6.

9.2.1 Ray–Gaussian Shadowing

We represent each occluder as an anisotropic 3D Gaussian with center $\mu \in \mathbb{R}^3$, rotation $\mathbf{R} \in \text{SO}(3)$, and per-axis scales $\mathbf{s} = (s_x, s_y, s_z)$. The covariance matrix is (cf. Eq. (3.9))

$$\Sigma = \mathbf{R} \text{diag}(s_x^2, s_y^2, s_z^2) \mathbf{R}^\top, \quad (9.1)$$

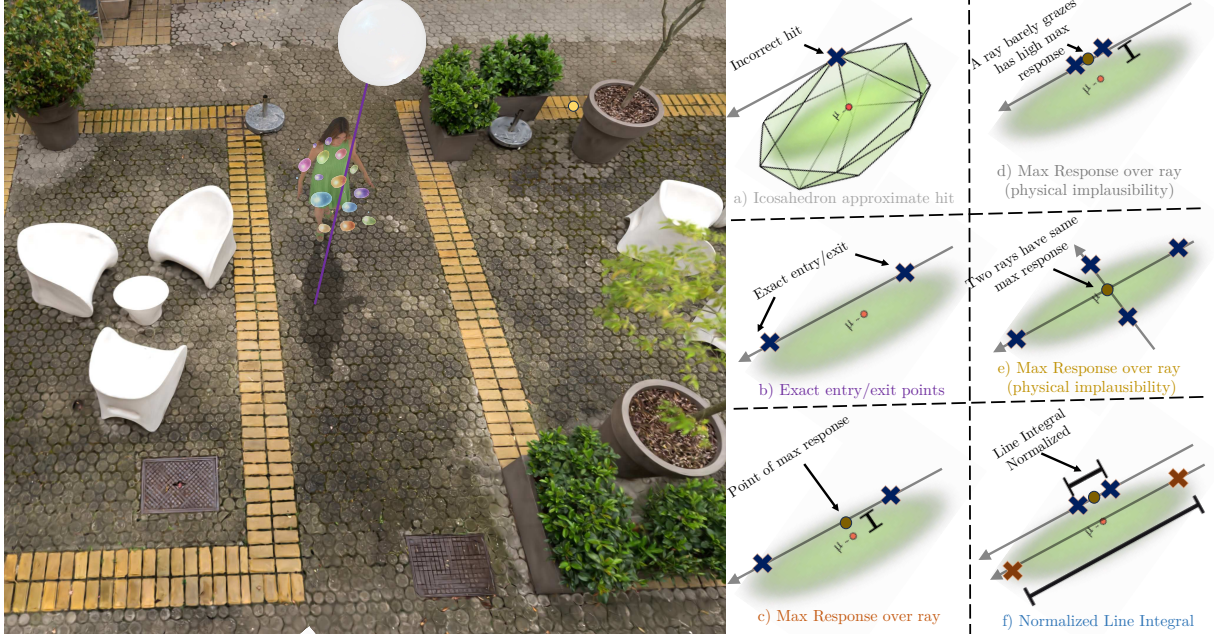


Figure 9.2: **Method overview.** Left: a shadow ray is cast from a scene Gaussian toward the light source through an animated 3DGS avatar by accumulating ray transmittance over all Gaussians in the avatar. Right: comparison of ray–Gaussian intersection strategies. (a) The icosahedron proxy used by 3DGRT [Moenne-Loccoz et al., 2024] produces incorrect hits outside the Gaussian support. (b) Our exact quadratic test (Sec. 9.2.2) finds precise entry/exit points. (c) Evaluating the maximum Gaussian response along the ray ignores traversal depth (Sec. 9.2.2). (d) The point of maximum response does not reflect total volumetric obstruction (Sec. 9.2.2). (e) Two rays with different traversals receive the same max response, and a grazing ray is overweighted (Sec. 9.2.2). (f) Our normalized line integral (Sec. 9.2.3, Eq. (9.13)) correctly captures how much of each Gaussian the ray traverses.

and the unnormalized Gaussian density is

$$g(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right), \quad (9.2)$$

where $g(\boldsymbol{\mu}) = 1$ (Lemma 9.1). This is the unnormalized Gaussian density $\mathcal{G}_k(\mathbf{x})$ from Eq. (3.8), written without subscript when the occluder index is clear from context.

Given a receiver point $\mathbf{o}$ and a point light position $\boldsymbol{\ell}$, we cast a shadow ray

$$\mathbf{x}(\tau) = \mathbf{o} + \tau \mathbf{v}, \quad \tau \in [0, 1], \quad \mathbf{v} = \boldsymbol{\ell} - \mathbf{o}. \quad (9.3)$$

Gaussian-space transform. To simplify the intersection test, we first define the whitening transform $\mathbf{L} = \text{diag}(s_x^{-1}, s_y^{-1}, s_z^{-1}) \mathbf{R}^\top$, so that $\mathbf{y} = \mathbf{L}(\mathbf{x} - \boldsymbol{\mu})$ satisfies $\|\mathbf{y}\|_2^2 = (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})$. The ray becomes

$$\mathbf{y}(\tau) = \mathbf{o}' + \tau \mathbf{v}', \quad \mathbf{o}' = \mathbf{L}(\mathbf{o} - \boldsymbol{\mu}), \quad \mathbf{v}' = \mathbf{L}\mathbf{v}. \quad (9.4)$$

9.2.2 Exact Ellipsoid Intersection and Entry/Exit

We restrict contributions to the χ^2 -confidence ellipsoid (we use $\chi^2 = 9$, corresponding to a 3σ cutoff):

$$\|\mathbf{y}\|_2^2 \leq \chi^2, \quad (9.5)$$

which yields a quadratic constraint along the ray (Prop. 9.1):

$$\|\mathbf{o}' + \tau \mathbf{v}'\|_2^2 = A\tau^2 + 2B\tau + C, \quad (9.6)$$

with coefficients

$$A = \|\mathbf{v}'\|_2^2, \quad B = \mathbf{o}'^\top \mathbf{v}', \quad C = \|\mathbf{o}'\|_2^2. \quad (9.7)$$

The maximum-response point on the ray (minimum of Eq. (9.6)) and the minimum Mahalanobis distance squared are (Prop. 9.2)

$$\tau_c = -\frac{B}{A}, \quad d_\perp^2 = C - \frac{B^2}{A}. \quad (9.8)$$

If $d_\perp^2 > \chi^2$, the ray misses the ellipsoid. Otherwise, defining $\Delta = \chi^2 - d_\perp^2$, the entry and exit times are (Cor. 9.1)

$$\tau_{\text{in}} = \tau_c - \sqrt{\frac{\Delta}{A}}, \quad \tau_{\text{out}} = \tau_c + \sqrt{\frac{\Delta}{A}}. \quad (9.9)$$

We clamp to the ray segment: $\tilde{\tau}_{\text{in}} = \max(\tau_{\text{in}}, 0)$, $\tilde{\tau}_{\text{out}} = \min(\tau_{\text{out}}, 1)$, and discard if $\tilde{\tau}_{\text{in}} \geq \tilde{\tau}_{\text{out}}$.

Why not evaluate at the peak? A simpler alternative to integration would be to use the Gaussian's response at the closest point τ_c (Eq. (9.9)), i.e. $g(\mathbf{x}(\tau_c)) = \exp(-\frac{1}{2}d_\perp^2)$ (Fig. 9.2 c). However, this value depends only on the perpendicular distance $d_\perp$ to the Gaussian center and is entirely *direction-agnostic*: two rays at the same distance but traversing vastly different distances through the Gaussian receive identical weights (Fig. 9.2 e). Worse, a ray barely grazing a large Gaussian along its major axis can have a small $d_\perp$ and thus a high peak response, despite minimal volumetric obstruction (Fig. 9.2 d). The line integral (Sec. 9.2.3) resolves both issues by accumulating opacity over the full traversal length, naturally producing lower weights for shorter paths through the Gaussian.

9.2.3 Truncated Line Integral (Closed Form)

Along the ray, Eq. (9.2) reduces to a 1D Gaussian in τ :

$$g(\mathbf{x}(\tau)) = \exp\left(-\frac{1}{2}\|\mathbf{o}' + \tau \mathbf{v}'\|_2^2\right) = \exp\left(-\frac{1}{2}d_\perp^2\right) \exp\left(-\frac{1}{2}A(\tau - \tau_c)^2\right). \quad (9.10)$$

We integrate over the ellipsoid intersection interval, $I_{\text{line}} = \int_{\tilde{\tau}_{\text{in}}}^{\tilde{\tau}_{\text{out}}} g(\mathbf{x}(\tau)) d\tau$, which admits a closed-form solution via the error function (Props. 9.3–9.4):

$$I_{\text{line}} = \exp\left(-\frac{1}{2}d_\perp^2\right) \sqrt{\frac{\pi}{2A}} \left[\text{erf}\left(\sqrt{\frac{A}{2}}(\tilde{\tau}_{\text{out}} - \tau_c)\right) - \text{erf}\left(\sqrt{\frac{A}{2}}(\tilde{\tau}_{\text{in}} - \tau_c)\right) \right]. \quad (9.11)$$

9.2.4 Normalization by Maximum Line Integral

To make the integral dimensionless and comparable across Gaussians of varying size, we normalize by the maximum attainable line integral within the same χ^2 bound—a center pass ($d_\perp^2 = 0$) through the longest axis $s_{\max} = \max(\mathbf{s})$ (proof in Sec. 9.6.5):

$$I_{\max} = \sqrt{2\pi} \frac{s_{\max}}{\|\mathbf{v}\|_2} \operatorname{erf}\left(\sqrt{\frac{\chi^2}{2}}\right). \quad (9.12)$$

The normalized thickness factor $\eta \in [0, 1]$ (Prop. 9.5) is then

$$\eta = \operatorname{clamp}\left(\frac{I_{\text{line}}}{I_{\max}}, 0, 1\right), \quad (9.13)$$

which is near 0 for grazing intersections and approaches 1 for a center pass along the major axis.

9.2.5 From Opacity to Shadow Transmittance

Each occluder Gaussian carries an opacity parameter $\alpha \in (0, 1)$. We update the shadow transmittance T multiplicatively over all Gaussians intersecting the ray (found via BVH traversal and the exact test above). The normalized thickness factor η modulates each Gaussian’s effective attenuation.

We consider three transmittance update rules:

- **Linear:** uses η as fractional coverage:

$$T \leftarrow T (1 - \alpha \eta). \quad (9.14)$$

- **Exponentiated:** treats η as a fraction of optical thickness, equivalent to $\alpha_{\text{eff}} = 1 - (1 - \alpha)^\eta$ (Prop. 9.6):

$$T \leftarrow T (1 - \alpha)^\eta. \quad (9.15)$$

- **Beer–Lambert:** maps opacity to extinction $\kappa(\alpha) = -\ln(1 - \alpha)$ (Prop. 9.7):

$$T \leftarrow T \exp(-\kappa(\alpha) I_{\text{line}}). \quad (9.16)$$

We use the exponentiated model in all experiments (ablation in Sec. 9.4.5).

9.2.6 Acceleration via BVH over Conservative AABBs

To accelerate intersection queries, we build a BVH over per-Gaussian axis-aligned bounding boxes that conservatively enclose the χ^2 ellipsoid in world space. The ellipsoid projection radius along world axis $d \in \{x, y, z\}$ is

$$e_d = \sqrt{\chi^2 (s_x^2 \mathbf{R}_{d1}^2 + s_y^2 \mathbf{R}_{d2}^2 + s_z^2 \mathbf{R}_{d3}^2)}, \quad (9.17)$$

yielding $\text{AABB} = [\boldsymbol{\mu} - \mathbf{e}, \boldsymbol{\mu} + \mathbf{e}]$. Only BVH candidates invoke the exact quadratic test Eqs. (9.6)–(9.9) and integral Eq. (9.11), enabling efficient soft shadow estimation in dense 3DGS scenes. We implement the BVH using NVIDIA OptiX, leveraging hardware RT cores for near real-time performance (Sec. 9.3.3).

9.2.7 Shadow Rendering

After computing the transmittance $T_k \in [0, 1]$ for each scene Gaussian k , we modulate its color during 3DGS rasterization. The shadowed color of Gaussian k is $\mathbf{c}'_k = T_k \mathbf{c}_k$, where $T_k = 1$ indicates full illumination and $T_k \rightarrow 0$ indicates full shadow. The modified colors $\mathbf{c}'_k$ are then splatted using standard α -blending, producing the final shadowed rendering.

9.2.8 Avatar Proxy for Temporally Stable Shadows

While the pose-dependent 3DGS avatar model produces high-fidelity renderings, using the raw primitives for shadow casting introduces temporal instability. The pose-to-geometry generator $\mathcal{H}(\boldsymbol{\theta}_t)$ —a neural network that maps body pose $\boldsymbol{\theta}_t$ to 3D Gaussian primitives—produces outputs that fluctuate in topology, opacity, and anisotropic orientation to satisfy view-dependent photometric losses. When used as shadow occluders, these fluctuations cause the shadow silhouette to flicker and oscillate rapidly between frames.

To address this, we decouple the render-ready geometry from the shadow-casting geometry by constructing a **stable avatar proxy** (Fig. 9.3)—a regularized derivative of the avatar that guarantees temporal coherence.

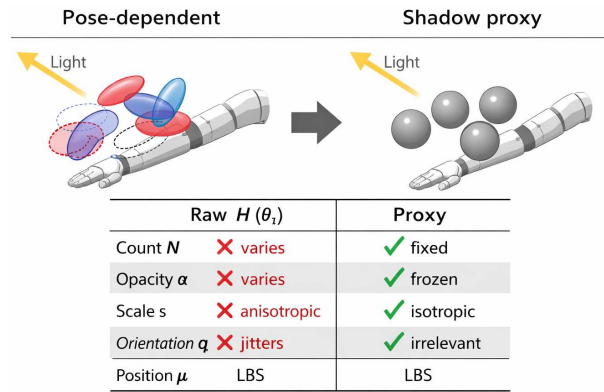


Figure 9.3: **Avatar proxy.** We replace the pose-dependent anisotropic Gaussians (left) with a stable isotropic proxy (right) that eliminates temporal flickering in shadows.

Canonical topology locking. To eliminate topological noise, we freeze the generative process at a canonical reference pose $\boldsymbol{\theta}_{\text{ref}}$ (e.g., a T-pose), caching the configuration once:

$$\mathcal{P}_{\text{proxy}} = \{(\boldsymbol{\mu}_k^{(0)}, \mathbf{s}_k^{(0)}, \alpha_k^{(0)})\}_{k=1}^N = \mathcal{H}(\boldsymbol{\theta}_{\text{ref}}). \quad (9.18)$$

By fixing the number of primitives N and their initial attributes, we ensure a persistent set of occluders, preventing artifacts caused by primitives appearing and disappearing between frames.

Isotropic regularization. To enforce rotational invariance, we override the anisotropic scale with a conservative isotropic radius $r_k = \max(s_{k,x}^{(0)}, s_{k,y}^{(0)}, s_{k,z}^{(0)})$, transforming the covariance into $\tilde{\Sigma}_k = r_k^2 \mathbf{I}$. Since a sphere’s silhouette is rotation-invariant, the shadow projection becomes independent of the primitive’s orientation.

Simplified kinematics. During inference, we articulate the proxy using Linear Blend Skinning (LBS) applied strictly to positions. Given the canonical center $\mu_k^{(0)}$ and joint transforms $\mathbf{G}_j(\theta_t)$ (Eq. (3.1)):

$$\mu_k^{(t)} = \sum_{j=1}^J w_{k,j} \mathbf{G}_j(\theta_t) \mu_k^{(0)}, \quad (9.19)$$

where $w_{k,j} \geq 0$ are normalized skinning weights ($\sum_{j=1}^J w_{k,j} = 1$). Crucially, we do **not** update the orientation of the proxies. Since the primitives are isotropic spheres, their rotation is irrelevant to the ray intersection test, guaranteeing that the shadow signal is continuous with respect to the pose θ_t .

9.3 IMPLEMENTATION DETAILS

We describe the 3DGS avatar model used in our experiments, the GPU implementation of our shadow tracer, and analyze its runtime performance.

9.3.1 3DGS Avatar Model

Our shadow casting method operates on pose-dependent 3DGS avatars produced by a generator $\mathcal{H}(\theta_t)$. We use the Animatable Gaussians framework [Li et al., 2024d], which maps SMPL body poses to 3D Gaussian primitives via a learned neural network. The pipeline consists of three stages.

Template reconstruction. Given multi-view video of a person, a character-specific canonical template is reconstructed by optimizing a canonical signed distance field (SDF) and color field from a near-A-pose frame, then extracting a mesh via Marching Cubes. SMPL skinning weights are diffused from the body surface to the template surface, enabling the template to represent loose garments while remaining deformable via LBS.

Pose maps and Gaussian maps. For each pose θ_t , the canonical template is deformed via LBS and rendered under front and back orthographic projections, producing position maps $\mathcal{P}_f(\theta_t)$ and $\mathcal{P}_b(\theta_t)$ that encode the current body configuration as structured 2D images. A StyleUNet f_ϕ —a U-Net augmented with StyleGAN-based modulation layers—maps these position maps to Gaussian maps:

$$\mathcal{M}_f(\theta_t), \mathcal{M}_b(\theta_t) \leftarrow f_\phi(\mathcal{P}_f(\theta_t), \mathcal{P}_b(\theta_t), \mathcal{V}),$$

where $\mathcal{V}$ encodes the view direction relative to the template normal. Each pixel in a Gaussian map encodes a complete 3D Gaussian primitive: a position offset $\Delta\mu$

relative to the template surface, per-axis scales $\mathbf{s}$, a rotation quaternion $\mathbf{q}$, opacity α , and view-dependent color $\mathbf{c}$. The collection of all foreground pixels across front and back maps yields the full set of canonical Gaussians $\mathcal{G}_t^{H,C}$.

Articulation via LBS. Canonical Gaussians are articulated into posed space using linear blend skinning. Each Gaussian k inherits skinning weights $w_{k,j}$ from its nearest template vertex. Given pose θ_t , the posed Gaussian center is $\mu_k^{(t)} = \sum_{j=1}^J w_{k,j} \mathbf{G}_j(\theta_t) \mu_k^{(0)}$, where $\mathbf{G}_j(\theta_t)$ are the per-joint rigid transformations from the SMPL skeleton (Eq. (3.1)). The covariance is similarly transformed using the blended rotation. The StyleUNet is trained end-to-end from multi-view video with a photometric reconstruction loss (L1 + perceptual).

This pipeline defines the differentiable mapping $\mathcal{H} : \theta_t \mapsto \mathcal{G}_t^H$ used throughout our method. As discussed in Sec. 9.2.8, the photometric optimization causes per-frame fluctuations in Gaussian attributes that are imperceptible during rendering but produce temporally unstable shadows, motivating our avatar proxy.

9.3.2 Shadow Tracer Implementation

We implement our shadow tracer as custom CUDA kernels integrated with the NVIDIA OptiX framework for hardware-accelerated ray tracing. The BVH over conservative AABBs (Sec. 9.2.6) is built using OptiX’s built-in BVH builder, which leverages RT cores on NVIDIA GPUs for efficient traversal.

9.3.3 Runtime Analysis

We analyze the runtime of RAGA as a function of avatar complexity on an NVIDIA H100 GPU with $\sim 500\text{K}$ scene Gaussians. Fig. 9.6 reports per-frame shadow computation time as the number of avatar Gaussians varies from 20K to 400K. Only our shadow tracer, implemented as a CUDA kernel, runs at interactive rates of about 50 FPS; the end-to-end pipeline (including 3DGS rasterization) achieves near real-time performance even for dense avatars thanks to OptiX hardware ray tracing and BVH acceleration.

9.4 EXPERIMENTS

We evaluate RAGA along two axes: *shadow quality*—how accurately do our ray-traced Gaussian shadows approximate mesh-based ground truth (Sec. 9.4.2)?; *temporal stability*—does the avatar proxy reduce flickering compared to raw Gaussian shadow casting (Sec. 9.4.3)?; in both evaluations we also compare with two modified ray-traced methods, 3DGRT [Moenne-Loccoz et al., 2024] and RaySplat [Byrski et al., 2025]. We further report *ablations* on each pipeline component (Sec. 9.4.5) and a timing analysis (Sec. 9.3.3).

Table 9.2: **Shadow quality vs. pseudo-GT on ScanNet++**. Averaged over 5 scenes. Mesh Shadow is an oracle.

Method	SAE↓	SM-IoU↑	BF↑
Mesh Shadow (oracle)	0.000	1.000	1.000
Ours (RAGA)	0.031	0.847	0.812
3DGRT (mod.)	0.058	0.741	0.693
RaySplat (mod.)	0.062	0.728	0.679

Table 9.3: **Temporal consistency (TSC↓)**. Averaged over 5 sequences. Lower = more stable.

Method	TSC↓
Ours (full proxy)	0.0018
Ours w/o proxy	0.0061
3DGRT (mod.)	0.0114
RaySplat (mod.)	0.0121

9.4.1 Experimental Setup

Pseudo-GT Protocol. We obtain pseudo-GT shadow maps $S^+(p)$ via classical mesh-based rendering in two settings:

- **ScanNet++ [Yeshwanth et al., 2023] scenes:** Since ScanNet++ provides both a 3DGS and a mesh reconstruction, we replace the avatar with the posed SMPL mesh and cast shadows onto the scene mesh using classical shadow mapping. We render the shadow attenuation onto a transparent surface and composite it with the 3DGS scene rendering. We use 5 ScanNet++ scenes for quantitative evaluation.
- **General 3DGS scenes (e.g. SuperSplat [SuperSplat]):** For scenes without a mesh reconstruction, we construct an approximate floor plane from the 3DGS point cloud, replace the avatar with the posed SMPL mesh, and cast shadows onto the floor proxy via classical rendering. The floor is set to invisible and the resulting shadow map is composited with the 3DGS rendering.

Why not use mesh-based shadow casting? While the pseudo-GT protocol above uses meshes to establish a reference, mesh-based shadows are not a practical solution for general 3DGS scenes:

- **Scene mesh extraction is lossy.** Obtaining a high-quality mesh from a 3DGS scene is not straightforward and leads to significant loss of detail [Ye et al., 2024]. While we use a floor proxy for pseudo-GT, this discards all wall- and furniture-based shadow receivers—ScanNet++ is an exception where dense meshes happen to be available.
- **Mesh proxies lose detail.** Mesh-based shadow casting requires replacing occluders with mesh proxies such as SMPL for humans, which discards clothing and hair detail from the shadow silhouette. While our avatar proxy also smooths fine detail, it retains significantly more geometric fidelity than the bare SMPL mesh because it operates on the actual Gaussian point cloud rather than a parametric body model.

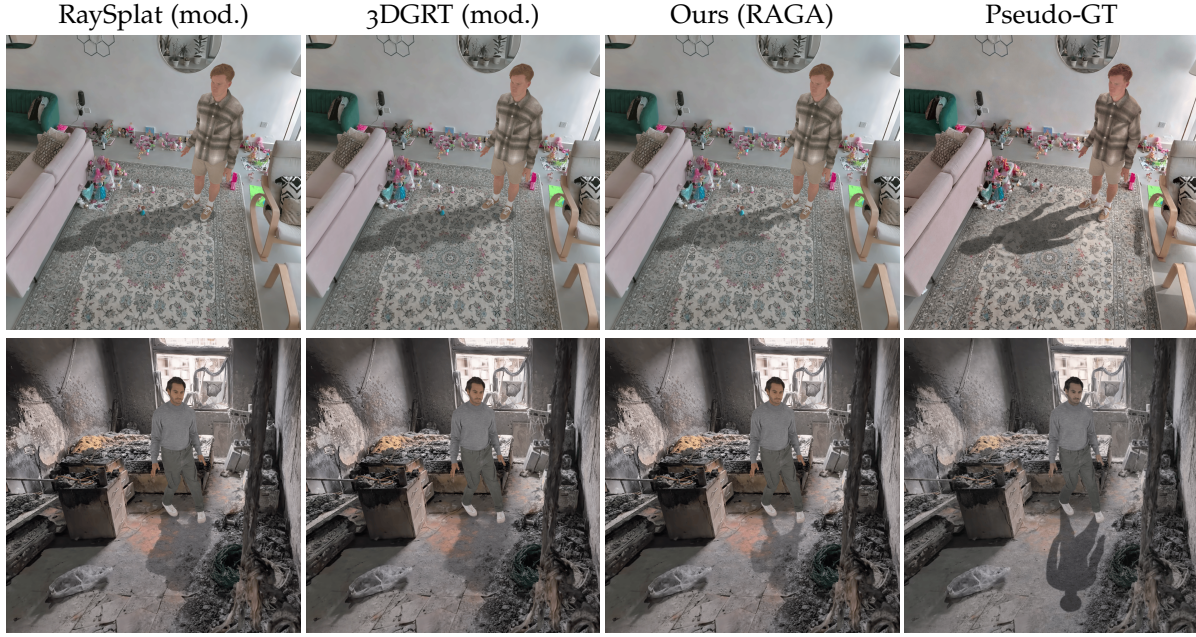


Figure 9.4: **Qualitative comparison on SuperSplat scenes.** Pseudo-GT is obtained using a proxy floor mesh and posed SMPL mesh via classical shadow mapping. Our method produces soft, physically plausible shadows that closely match the pseudo-GT. Both 3DGRT [Moenne-Loccoz et al., 2024] and RaySplat [Byrski et al., 2025] lose shadow detail due to their implausible attenuation models.

- **No support for Gaussian objects.** Mesh proxies do not generalize to arbitrary 3DGS objects inserted into scenes. Our method operates entirely in Gaussian space and naturally supports avatar–object interactions, as shown in the qualitative results (Sec. 9.4.6).

Avatars. We use 3DGS avatars from ActorsHQ [Işık et al., 2023] (48 cameras) and AvatarReX [Zheng et al., 2023], spanning a range of reconstruction quality. For multi-avatar and avatar-object scenarios, we additionally use NeuralDome [Zhang et al., 2023].

Baselines. Since no prior method addresses the same problem (shadow casting for animated 3DGS avatars in 3DGS scenes), we construct the strongest possible Gaussian-based baselines by modifying two recent ray-tracing methods for shadow casting:

- **3DGRT** [Moenne-Loccoz et al., 2024] (modified): Following 3DGRT, we first detect ray–Gaussian intersections using an icosahedron approximation around each Gaussian, then for detected intersections we accumulate opacity values to compute ray attenuation. This formulation does not use our line integral or avatar proxy.
- **RaySplat** [Byrski et al., 2025] (modified): We detect ray–Gaussian intersections using the exact analytic test from RaySplat, then accumulate opacity for each detected Gaussian. As with 3DGRT, this formulation uses neither our line integral nor avatar proxy.

Table 9.4: **Ablation studies.** Evaluated against pseudo-GT on ScanNet++. Best in each group is **bold**.

Metric	(A) Attenuation model			(B) Normalization	
	Exp. (Ours)	Linear	Beer-L.	$I_{\max}$ (Ours)	w/o norm.
SAE ↓	0.031	0.035	0.039	0.031	0.047
SM-IoU ↑	0.847	0.831	0.819	0.847	0.793
BF ↑	0.812	0.796	0.781	0.812	0.754
TSC ↓	0.0018	0.0021	0.0024	0.0018	0.0022

- **Mesh Shadow (oracle):** We render a shadow-only pass by casting shadows from the posed SMPL mesh onto a transparent mesh receiver via classical shadow mapping, then alpha-blend the result onto the 3DGS scene rendering.

9.4.2 Shadow Quality Evaluation

We evaluate shadow maps against the mesh-based pseudo-GT using three pixel-space metrics computed in an avatar-centric region of interest:

- **SAE** (Shadow Attenuation Error): Mean absolute error between predicted and pseudo-GT shadow within the shadow region Ω_s (lower is better).
- **SM-IoU** (Shadow Matte IoU): Intersection-over-union of binarized shadow masks (threshold 0.1) (higher is better).
- **BF** (Boundary F-measure): F-score between shadow boundaries with 2 px tolerance (higher is better).

Tab. 9.2 reports shadow quality averaged over 5 ScanNet++ scenes. Our method achieves the best results among all Gaussian-based approaches, approaching the mesh oracle. 3DGRT’s binary hit test based on its icosahedron approximation produces hard shadow boundaries without penumbrae (Fig. 9.2 a), leading to poor SAE and BF scores. RaySplat’s collision-based approach reduces to a max-response evaluation (Fig. 9.2 c–e) that lacks proper attenuation modeling. Fig. 9.4 shows a visual comparison.

9.4.3 Temporal Stability

A key contribution of RAGA is the avatar proxy (Sec. 9.2.8) that ensures temporally stable shadows under animation. We measure temporal shadow consistency (TSC) as

Table 9.5: **Avatar proxy ablation.** Temporal stability (TSC↓). Lower = more stable.

Variant	TSC↓
Full proxy (Ours)	0.0018
w/o topology lock	0.0037
w/o isotropic reg.	0.0032
No proxy	0.0061

Figure 9.5: **Visual ablations.** Without $I_{\max}$ normalization (left), large anisotropic Gaussians cast disproportionately dark shadows. The linear attenuation model (middle) produces harder shadow falloff than our full exponentiated model (right).

the mean per-pixel absolute difference between consecutive shadow frames:

$$\text{TSC} = \frac{1}{F-1} \sum_{t=1}^{F-1} \frac{1}{|\Omega|} \sum_{p \in \Omega} |S_t(p) - S_{t-1}(p)| ,$$

where S_t is the shadow map at frame t , F is the total number of frames, and Ω is the avatar-centric ROI. Lower TSC indicates more stable shadows. Tab. 9.3 shows that the avatar proxy substantially reduces temporal flickering. Fig. 9.7 shows temporal difference maps ($|S_t - S_{t-1}|$) confirming that the avatar proxy reduces shadow flickering.

9.4.4 Perceptual Study

We conduct a two-alternative forced-choice (2AFC) study with 12 naive raters across 5 ScanNet++ scenes, comparing **Ours (RAGA)** against a **No Shadow** baseline. Raters are shown 5 video clips (10 s at 30 fps) per scene with randomized method order and placement, and asked which rendering looks more realistic. On average, raters prefer RAGA 73% of the time.

9.4.5 Ablation Studies

We ablate each component of RAGA in Tab. 9.4. For the attenuation model, we compare linear, exponentiated, and Beer–Lambert transmittance updates (Eqs. (9.14)–(9.16));

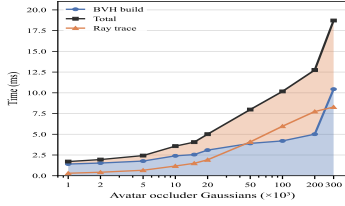


Figure 9.6: **Runtime vs. avatar Gaussians.**



No proxy

Ours (proxy)

Figure 9.7: **Proxy ablation.** $|S_t - S_{t-1}|$ maps: bright = large change. The proxy stabilizes shadows.

the exponentiated model provides the best trade-off. For line integral normalization, we compare the raw line integral I_{line} against our normalized thickness factor $\eta = I_{\text{line}}/I_{\text{max}}$ (Eq. (9.13)); without normalization, large anisotropic Gaussians cast disproportionately dark shadows (Fig. 9.5). For the avatar proxy, we ablate canonical topology locking and isotropic regularization (Tab. 9.5).

9.4.6 Qualitative Results

Fig. 9.8 shows results across diverse scenarios: single-avatar animation, multi-avatar motion, and avatar-object interaction in various ScanNet++ and SuperSplat [SuperSplat] scenes. Our method produces soft, physically plausible shadows that follow the avatar’s motion and respect scene geometry. We demonstrate results for multiple avatars interacting with scenes or avatars interacting with 3DGS objects placed in 3DGS scenes.

9.5 CONCLUSION

We presented RAGA, a principled and efficient framework for physically plausible shadow casting in animated 3DGS avatar-scene interaction scenarios. By operating entirely in Gaussian space, our approach overcomes a fundamental limitation of prior work, which relies on mesh-based shadow computation or approximate ray-Gaussian interactions that break down in dynamic settings.

At the core of RAGA is a ray-traced Gaussian shadow casting formulation that models light attenuation as a set of independent absorption events along rays cast toward light sources. Leveraging the closed-form solution of the ray-Gaussian line integral and analytically computable entry and exit points, our method enables stable and physically grounded aggregation of shadow contributions from multiple volumetric occluders. This principled formulation avoids the approximations commonly used in existing ray tracing approaches for 3DGS and results in significantly improved temporal stability and visual coherence.

To further address shadow instability in animated avatars, we introduced an avatar proxy that reduces variance caused by non-rigid deformations, leading to more consistent shadows during motion without sacrificing visual quality. Our GPU



Figure 9.8: **Qualitative results.** Our method casts plausible soft shadows for various 3DGS avatars animated alone and with objects across diverse scenes.

implementation integrates custom CUDA kernels with NVIDIA OptiX, exploiting hardware ray tracing to achieve near real-time performance. Extensive experiments across single-avatar, multi-avatar, and avatar–object interaction scenarios demonstrate that RAGA produces realistic, stable shadows.

While our method advances shadow casting for 3DGS avatars, several limitations remain. We model point and directional lights but do not support area lights, which would produce more realistic soft shadow penumbrae. Lighting on the avatar and scene remains baked into the 3DGS representation—we modulate scene Gaussian colors to create shadows but do not relight the avatar or scene under novel illumination. Similarly, we do not estimate light source positions from the scene; the user must specify them manually. Future work includes automatic light source estimation from 3DGS scenes, fully relightable avatars that respond to dynamic illumination, and area light support.

9.6 APPENDIX: PROOFS

This section provides the formal proofs for all propositions stated in Sec. 9.2: Lemma 9.1 and Propositions 9.1–9.7, covering the quadratic ray–Gaussian expansion (Sec. 9.2.2), the χ^2 intersection interval (Sec. 9.2.2), the closed-form line integral via the error function (Sec. 9.2.3), the global maximum $I_{\max}$ (Sec. 9.2.4), bounds on the normalized factor η (Sec. 9.2.4), the exponentiated opacity equivalence (Sec. 9.2.5), and the optical-depth mapping (Sec. 9.2.5).

9.6.1 Preliminaries and Notation

We restate the setup from Sec. 9.2.1 for self-containedness. We consider the unnormalized anisotropic Gaussian kernel

$$g(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right), \quad (9.20)$$

and a ray (or segment)

$$\mathbf{x}(\tau) = \mathbf{o} + \tau \mathbf{v}, \quad \tau \in [0, 1], \quad (9.21)$$

where $\mathbf{o}$ is the receiver point and $\mathbf{v} = \boldsymbol{\ell} - \mathbf{o}$ is the direction toward the light. We write $\mathbf{P} = \boldsymbol{\Sigma}^{-1} \succ 0$ for brevity and define the quadratic form along the ray:

$$q(\tau) \triangleq (\mathbf{x}(\tau) - \boldsymbol{\mu})^\top \mathbf{P}(\mathbf{x}(\tau) - \boldsymbol{\mu}). \quad (9.22)$$

Lemma 9.1 (Maximum of unnormalized Gaussian). $g(\mathbf{x}) \leq 1$ for all $\mathbf{x}$ and $g(\boldsymbol{\mu}) = 1$.

Proof. Since $\mathbf{P} \succ 0$, the quadratic form $(\mathbf{x} - \boldsymbol{\mu})^\top \mathbf{P}(\mathbf{x} - \boldsymbol{\mu}) \geq 0$ with equality iff $\mathbf{x} = \boldsymbol{\mu}$. Therefore the exponent is ≤ 0 and $g(\mathbf{x}) = \exp(-\frac{1}{2} \cdot \text{nonnegative}) \leq 1$, with $g(\boldsymbol{\mu}) = \exp(0) = 1$. $\square$

9.6.2 Proof that $q(\tau)$ is a quadratic and expressions for (A, B, C)

Proposition 9.1 (Quadratic expansion). Let $\boldsymbol{\delta} = \mathbf{o} - \boldsymbol{\mu}$. Then $q(\tau)$ is a quadratic polynomial

$$q(\tau) = A\tau^2 + 2B\tau + C, \quad (9.23)$$

with

$$A = \mathbf{v}^\top \mathbf{P} \mathbf{v}, \quad B = \mathbf{v}^\top \mathbf{P} \boldsymbol{\delta}, \quad C = \boldsymbol{\delta}^\top \mathbf{P} \boldsymbol{\delta}. \quad (9.24)$$

Proof. We have $\mathbf{x}(\tau) - \boldsymbol{\mu} = \mathbf{o} + \tau \mathbf{v} - \boldsymbol{\mu} = \boldsymbol{\delta} + \tau \mathbf{v}$. Then

$$\begin{aligned} q(\tau) &= (\boldsymbol{\delta} + \tau \mathbf{v})^\top \mathbf{P}(\boldsymbol{\delta} + \tau \mathbf{v}) \\ &= \boldsymbol{\delta}^\top \mathbf{P} \boldsymbol{\delta} + \tau \boldsymbol{\delta}^\top \mathbf{P} \mathbf{v} + \tau \mathbf{v}^\top \mathbf{P} \boldsymbol{\delta} + \tau^2 \mathbf{v}^\top \mathbf{P} \mathbf{v}. \end{aligned} \quad (9.25)$$

Since $\mathbf{P}$ is symmetric, $\boldsymbol{\delta}^\top \mathbf{P} \mathbf{v} = \mathbf{v}^\top \mathbf{P} \boldsymbol{\delta}$. Grouping terms yields

$$q(\tau) = (\mathbf{v}^\top \mathbf{P} \mathbf{v})\tau^2 + 2(\mathbf{v}^\top \mathbf{P} \boldsymbol{\delta})\tau + (\boldsymbol{\delta}^\top \mathbf{P} \boldsymbol{\delta}), \quad (9.26)$$

which gives Eq. (9.24). $\square$

Correspondence with Sec. 9.2.1. The whitening transform $\mathbf{L} = \text{diag}(s_x^{-1}, s_y^{-1}, s_z^{-1})\mathbf{R}^\top$ maps to Gaussian coordinates $\mathbf{y} = \mathbf{L}(\mathbf{x} - \boldsymbol{\mu})$. The quadratic coefficients $A = \|\mathbf{v}'\|_2^2$, $B = \mathbf{o}'^\top \mathbf{v}'$, $C = \|\mathbf{o}'\|_2^2$ (with $\mathbf{o}' = \mathbf{L}\boldsymbol{\delta}$ and $\mathbf{v}' = \mathbf{L}\mathbf{v}$) are identical to Eq. (9.24) since $\mathbf{P} = \mathbf{L}^\top \mathbf{L}$.

9.6.3 Proof of the χ^2 intersection interval and entry/exit times

We restrict contributions to the χ^2 level set:

$$q(\tau) \leq \chi^2. \quad (9.27)$$

Substituting $q(\tau) = A\tau^2 + 2B\tau + C$ gives

$$A\tau^2 + 2B\tau + (C - \chi^2) \leq 0. \quad (9.28)$$

Proposition 9.2 (Intersection on the line). *Assume $A > 0$ (true because $\mathbf{P} \succ 0$ and $\mathbf{v} \neq 0$). Define the discriminant*

$$\Delta = 4B^2 - 4A(C - \chi^2). \quad (9.29)$$

If $\Delta < 0$ then Eq. (9.28) has no real solutions. If $\Delta \geq 0$ then the solution set on the infinite line is the closed interval $[\tau^-, \tau^+]$ where

$$\tau^\pm = \frac{-2B \pm \sqrt{\Delta}}{2A} = -\frac{B}{A} \pm \frac{1}{2A}\sqrt{\Delta}, \quad \text{with } \tau^- \leq \tau^+. \quad (9.30)$$

Proof. The quadratic polynomial $p(\tau) = A\tau^2 + 2B\tau + (C - \chi^2)$ opens upward since $A > 0$. If $\Delta < 0$, $p(\tau)$ has no real roots and is strictly positive for all τ , so $p(\tau) \leq 0$ has no solutions. If $\Delta \geq 0$, $p(\tau)$ has two real roots given by the quadratic formula Eq. (9.30). For an upward-opening parabola, $p(\tau) \leq 0$ holds exactly between the roots (including endpoints), hence the solution set is $[\tau^-, \tau^+]$. $\square$

Correspondence with Sec. 9.2.2. The center point $\tau_c = -B/A$ and perpendicular distance $d_\perp^2 = C - B^2/A$ (Eq. (9.9)) relate to the discriminant as $\Delta = 4A(\chi^2 - d_\perp^2) = 4A\Delta_{\text{main}}$, where $\Delta_{\text{main}} = \chi^2 - d_\perp^2$. The entry/exit times $\tau_{\text{in}} = \tau_c - \sqrt{\Delta_{\text{main}}/A}$ and $\tau_{\text{out}} = \tau_c + \sqrt{\Delta_{\text{main}}/A}$ follow directly from Eq. (9.30).

Corollary 9.1 (Intersection on the segment). *For a segment domain $\tau \in [0, 1]$, the contributing interval is*

$$\tilde{\tau}_{\text{in}} = \max(0, \tau^-), \quad \tilde{\tau}_{\text{out}} = \min(1, \tau^+), \quad (9.31)$$

and the Gaussian contributes iff $\tilde{\tau}_{\text{in}} < \tilde{\tau}_{\text{out}}$.

Proof. Intersect the line-solution interval $[\tau^-, \tau^+]$ with the segment domain $[0, 1]$. The intersection is empty iff $\max(0, \tau^-) \geq \min(1, \tau^+)$; otherwise it is precisely the interval Eq. (9.31). $\square$

9.6.4 Proof of the closed-form truncated line integral

We define the truncated line integral

$$I \triangleq \int_{\tilde{\tau}_{\text{in}}}^{\tilde{\tau}_{\text{out}}} \exp\left(-\frac{1}{2}(A\tau^2 + 2B\tau + C)\right) d\tau, \quad (9.32)$$

where $A > 0$.

Proposition 9.3 (Completing the square). *We can rewrite the quadratic as*

$$A\tau^2 + 2B\tau + C = A(\tau - \tau_c)^2 + d_{\perp}^2, \quad \tau_c = -\frac{B}{A}, \quad d_{\perp}^2 = C - \frac{B^2}{A}. \quad (9.33)$$

Proof. Expand $A(\tau - \tau_c)^2 = A(\tau^2 - 2\tau_c\tau + \tau_c^2) = A\tau^2 - 2A\tau_c\tau + A\tau_c^2$. Match the linear coefficient: $-2A\tau_c = 2B \Rightarrow \tau_c = -B/A$. Then the constant term must satisfy $A\tau_c^2 + d_{\perp}^2 = C$, i.e. $d_{\perp}^2 = C - A\tau_c^2 = C - B^2/A$. $\square$

Proposition 9.4 (Closed form with error function). *The integral Eq. (9.32) equals*

$$I = \exp\left(-\frac{1}{2}d_{\perp}^2\right) \sqrt{\frac{\pi}{2A}} \left[\text{erf}\left(\sqrt{\frac{A}{2}}(\tau - \tau_c)\right) \right]_{\tau=\tilde{\tau}_{\text{in}}}^{\tau=\tilde{\tau}_{\text{out}}}. \quad (9.34)$$

Proof. Using Eq. (9.33),

$$\begin{aligned} I &= \int_{\tilde{\tau}_{\text{in}}}^{\tilde{\tau}_{\text{out}}} \exp\left(-\frac{1}{2}(A(\tau - \tau_c)^2 + d_{\perp}^2)\right) d\tau \\ &= \exp\left(-\frac{1}{2}d_{\perp}^2\right) \int_{\tilde{\tau}_{\text{in}}}^{\tilde{\tau}_{\text{out}}} \exp\left(-\frac{A}{2}(\tau - \tau_c)^2\right) d\tau. \end{aligned} \quad (9.35)$$

Let $u = \sqrt{\frac{A}{2}}(\tau - \tau_c)$ so that $d\tau = \sqrt{\frac{2}{A}} du$. Then

$$\int \exp(-u^2) du = \frac{\sqrt{\pi}}{2} \text{erf}(u) + \text{const}, \quad (9.36)$$

and therefore

$$\begin{aligned} \int_{\tilde{\tau}_{\text{in}}}^{\tilde{\tau}_{\text{out}}} \exp\left(-\frac{A}{2}(\tau - \tau_c)^2\right) d\tau &= \sqrt{\frac{2}{A}} \int_{u_{\text{in}}}^{u_{\text{out}}} \exp(-u^2) du = \sqrt{\frac{2}{A}} \cdot \frac{\sqrt{\pi}}{2} [\text{erf}(u)]_{u_{\text{in}}}^{u_{\text{out}}} \\ &= \sqrt{\frac{\pi}{2A}} \left[\text{erf}\left(\sqrt{\frac{A}{2}}(\tau - \tau_c)\right) \right]_{\tilde{\tau}_{\text{in}}}^{\tilde{\tau}_{\text{out}}}. \end{aligned} \quad (9.37)$$

Multiplying by $\exp(-d_{\perp}^2/2)$ gives Eq. (9.34). $\square$

9.6.5 Proof of the global maximum line integral I_{max}

We claim the maximum possible line integral *within the same χ^2 truncation* occurs for a ray passing through the mean and aligned with the largest standard deviation axis.

Assume the covariance is parameterized as $\Sigma = \mathbf{R} \text{diag}(s_x^2, s_y^2, s_z^2) \mathbf{R}^{\top}$. Define $s_{\text{max}} = \max(s_x, s_y, s_z)$.

Step 1: Reduce to Gaussian coordinates. Let $\mathbf{L} = \text{diag}(s_x^{-1}, s_y^{-1}, s_z^{-1}) \mathbf{R}^\top$ and define $\mathbf{y} = \mathbf{L}(\mathbf{x} - \boldsymbol{\mu})$. Then $g(\mathbf{x}) = \exp(-\frac{1}{2}\|\mathbf{y}\|^2)$ and the χ^2 ellipsoid becomes $\|\mathbf{y}\|^2 \leq \chi^2$.

Step 2: Center pass maximizes the exponential prefactor. Along any line in $\mathbf{y}$ -space, the integrand is $\exp(-\frac{1}{2}\|\mathbf{y}(\tau)\|^2) \leq 1$ with equality only at $\mathbf{y} = 0$. For fixed direction, the maximum integral over the truncated interval is achieved when the line passes through $\mathbf{y} = 0$ (i.e. through $\boldsymbol{\mu}$ in world space), because this maximizes the integrand pointwise near the center and yields the largest possible mass under a symmetric unimodal function.

More formally, in the quadratic form $q(\tau) = A\tau^2 + 2B\tau + C$, the center pass corresponds to $B = 0$ and $C = 0$, hence $d_\perp^2 = C - B^2/A = 0$, which maximizes the multiplicative factor $\exp(-d_\perp^2/2)$ in Eq. (9.34).

Step 3: Alignment with largest axis maximizes the traversal length in world space. For a center pass, the truncation interval in *Gaussian coordinates* is the segment of length $2\sqrt{\chi^2}$ along the normalized direction in $\mathbf{y}$ -space. Mapping back to world space scales lengths by the corresponding standard deviation along that direction. The maximum world-space length for a unit-length direction is achieved when the ray aligns with the principal axis with the largest standard deviation $s_{\max}$.

Closed form. Under a unit-speed ray parameterization where τ measures world distance, the center-pass 1D integrand reduces to

$$\exp\left(-\frac{s^2}{2s_{\max}^2}\right), \quad s \in [-s_{\max}\sqrt{\chi^2}, s_{\max}\sqrt{\chi^2}], \quad (9.38)$$

where s is signed distance along the ray. Thus

$$I_{\max} = \int_{-s_{\max}\sqrt{\chi^2}}^{s_{\max}\sqrt{\chi^2}} \exp\left(-\frac{s^2}{2s_{\max}^2}\right) ds. \quad (9.39)$$

Substitute $u = s/(\sqrt{2}s_{\max})$, $ds = \sqrt{2}s_{\max}du$, to get

$$\begin{aligned} I_{\max} &= \sqrt{2}s_{\max} \int_{-\sqrt{\chi^2/2}}^{\sqrt{\chi^2/2}} e^{-u^2} du = \sqrt{2}s_{\max} \cdot \frac{\sqrt{\pi}}{2} [\text{erf}(u)]_{-\sqrt{\chi^2/2}}^{\sqrt{\chi^2/2}} \\ &= \sqrt{2}s_{\max} \cdot \frac{\sqrt{\pi}}{2} \cdot (\text{erf}(\sqrt{\chi^2/2}) - \text{erf}(-\sqrt{\chi^2/2})) \\ &= \sqrt{2}s_{\max} \cdot \frac{\sqrt{\pi}}{2} \cdot 2 \text{erf}(\sqrt{\chi^2/2}) \\ &= \sqrt{2\pi} s_{\max} \text{erf}\left(\sqrt{\frac{\chi^2}{2}}\right), \end{aligned} \quad (9.40)$$

using the odd symmetry $\text{erf}(-x) = -\text{erf}(x)$. This proves the stated maximum Eq. (9.40). $\square$

Non-unit parameterization. The method in Sec. 9.2.1 uses $\mathbf{x}(\tau) = \mathbf{o} + \tau \mathbf{v}$ with $\tau \in [0, 1]$ and $\mathbf{v}$ not unit-length. Then $d\tau$ corresponds to world distance $ds = \|\mathbf{v}\| d\tau$, so the maximum in τ -units becomes

$$I_{\max} = \frac{1}{\|\mathbf{v}\|_2} \sqrt{2\pi} s_{\max} \operatorname{erf}\left(\sqrt{\frac{\chi^2}{2}}\right), \quad (9.41)$$

matching Eq. (9.13).

9.6.6 Justification of the normalized factor semantics

Proposition 9.5 (Bounds for η). *With $\eta = I/I_{\max}$ and $I_{\max} > 0$, we have $\eta \geq 0$. If additionally $I \leq I_{\max}$ (i.e., if $I_{\max}$ is a valid global upper bound for the chosen parameterization), then $\eta \in [0, 1]$.*

Proof. The integrand in Eq. (9.32) is nonnegative, so $I \geq 0$ and thus $\eta \geq 0$. If $I \leq I_{\max}$, then $\eta = I/I_{\max} \leq 1$. In practice we also clamp η to $[0, 1]$. $\square$

9.6.7 Equivalence between exponentiated attenuation and effective opacity

Proposition 9.6 (Equivalence of the two exponentiated forms). *Let $\alpha \in [0, 1)$ and $\eta \geq 0$. Define $\alpha_{\text{eff}} = 1 - (1 - \alpha)^\eta$. Then*

$$(1 - \alpha)^\eta = 1 - \alpha_{\text{eff}}. \quad (9.42)$$

Therefore $T \leftarrow T(1 - \alpha)^\eta$ is identical to $T \leftarrow T(1 - \alpha_{\text{eff}})$.

Proof. By definition, $\alpha_{\text{eff}} = 1 - (1 - \alpha)^\eta$, rearranging gives $(1 - \alpha)^\eta = 1 - \alpha_{\text{eff}}$. $\square$

Corollary 9.2 (Thickness additivity). *For two traversals with factors η_1, η_2 , the combined transmittance multiplier is*

$$(1 - \alpha)^{\eta_1} (1 - \alpha)^{\eta_2} = (1 - \alpha)^{\eta_1 + \eta_2}. \quad (9.43)$$

Proof. Immediate from exponent rules. $\square$

9.6.8 Optical-depth mapping $\kappa(\alpha) = -\ln(1 - \alpha)$

Proposition 9.7 (From alpha to optical depth). *For $\alpha \in [0, 1)$, define $\kappa = -\ln(1 - \alpha) \geq 0$. Then*

$$1 - \alpha = e^{-\kappa}. \quad (9.44)$$

Proof. Take exponentials: $e^{-\kappa} = e^{\ln(1-\alpha)} = 1 - \alpha$. Since $1 - \alpha \in (0, 1]$, $-\ln(1 - \alpha) \geq 0$. $\square$

Corollary 9.3 (Small- α approximation). *As $\alpha \rightarrow 0$, $\kappa = -\ln(1 - \alpha) = \alpha + O(\alpha^2)$.*

Proof. Use the Taylor expansion $\ln(1 - x) = -(x + \frac{x^2}{2} + \dots)$ for $|x| < 1$. $\square$

AHOY! ANIMATABLE HUMANS UNDER OCCLUSION FROM YOUTUBE VIDEOS WITH GAUSSIAN SPLATTING AND VIDEO DIFFUSION PRIORS



Figure 10.1: **AHOY** reconstructs animatable 3D avatars from occluded YouTube video. Given a monocular YouTube video with occlusion, AHOY reconstructs a complete, animatable 3D human avatar using video diffusion priors and 3D Gaussian Splatting, enabling photorealistic human animation within 3D scenes.

The preceding chapters of Part II assume access to multi-view capture for avatar reconstruction. This chapter removes that requirement by exploiting a key advantage of 3DGS over classical mesh representations: 3DGS can be reconstructed from images alone—even sparse or monocular video—whereas mesh-based avatar reconstruction typically requires dense multi-view capture or depth sensors. We present AHOY, a method for reconstructing complete, animatable 3D Gaussian avatars from in-the-wild monocular video despite heavy occlusion. Existing methods assume unoccluded

input—a fully visible subject, often in a canonical pose—excluding the vast majority of real-world footage where people are routinely occluded by furniture, objects, or other people. Reconstructing from such footage poses fundamental challenges: large body regions may never be observed, and multi-view supervision per pose is unavailable. We address these challenges with four contributions: (i) a hallucination-as-supervision pipeline that uses identity-finetuned diffusion models to generate dense supervision for previously unobserved body regions; (ii) a two-stage canonical-to-pose-dependent architecture that bootstraps from sparse observations to full pose-dependent Gaussian maps; (iii) a map-pose/LBS-pose decoupling that absorbs multi-view inconsistencies from the generated data; (iv) a head/body split supervision strategy that preserves facial identity. We evaluate on YouTube videos and on multi-view capture data with significant occlusion and demonstrate state-of-the-art reconstruction quality. We also demonstrate that the resulting avatars are robust enough to be animated with novel poses and composited into 3DGS scenes captured using cell-phone video. Video results are available at <https://miraymen.github.io/ahoy>.

10.1 INTRODUCTION

Reconstructing photorealistic, animatable 3D humans from monocular input is a long-standing goal in computer vision and graphics, with applications spanning gaming, virtual reality, and film production. Recent methods based on NeRFs [Peng et al., 2021, Weng et al., 2022, Habermann et al., 2023] and 3DGS [Kocabas et al., 2024, Qian et al., 2024, Moreau et al., 2024, Zielonka et al., 2025, Qiu et al., 2025a, Zhuang et al., 2025, Huang et al., 2025, Dong et al., 2025] have made remarkable progress, producing high-fidelity avatars that can be re-posed in real time. Yet these methods assume unoccluded monocular input, or a fully visible subject, often in a canonical pose.

These assumptions exclude the vast majority of real-world footage. YouTube alone hosts billions of videos of people in everyday activities—cooking, exercising, conversing—where the body is occluded by furniture, other people, or the scene itself. The viewpoints are limited, and large portions of the body may never be observed or cut off in the camera shot. If we could reconstruct complete, animatable avatars from such videos, we would unlock an essentially unlimited source of diverse human appearance—every body type, clothing style, appearance—without controlled capture, multi-view rigs, or cooperation from the subject.

In this work, we present **AHOY** (Animatable Humans under Occlusion from YouTube Videos), a method that reconstructs complete, animatable 3D Gaussian avatars from in-the-wild monocular video despite heavy occlusion, and animates them within photorealistic 3D scenes (Fig. 10.1). Achieving this requires overcoming several challenges that do not arise when the subject is fully visible.

Bootstrapping a complete avatar from partial observations. With large body regions never observed, there is no direct supervision for a significant portion of the avatar. We address this in three stages. First, we build a coarse 3DGS avatar from sparse multi-view images generated from the partially observed subject (Sec. 10.2.1). Second, we render this coarse avatar in structured motion sequences and refine the imperfect

renderings via an identity-finetuned video diffusion model and diffusion inversion producing hallucinated supervision videos that cover previously unobserved body surfaces from multiple viewpoints and poses (Sec. 10.2.2). Third, we train a full avatar with pose-dependent deformation using these hallucinated videos as supervision (Sec. 10.2.3).

Learning pose-dependent appearance without multi-view capture. High-fidelity avatars require pose-dependent deformation to capture how clothing changes with body configuration. In the first stage, no multi-view supervision for pose-dependent deformation exists. We therefore learn an avatar that does not model pose-dependent deformations. We adapt the 2D Gaussian map design from Animatable Gaussians [Li et al., 2024d] to learn a canonical pose-independent Gaussian map that captures the texture and structure of the human. We also adopt the use of per-identity networks that use 2D maps to represent both input pose and output 3DGS properties. To obtain the pose-dependent multi-view supervision missing from the original video, we design the structured motion sequences of the second stage—full-body turns paired with target poses such as sitting and reaching—to provide both pose-dependent deformations and in-effect 360° multi-view coverage per pose. In the third stage, these hallucinated videos supply the dense multi-view supervision needed to upgrade from canonical to pose-dependent Gaussian maps that capture high-fidelity appearance changes across body configurations. This canonical-to-pose-dependent progression is the architectural insight that makes pose-dependent learning possible (Secs. 10.2.2.2 and 10.2.3.1).

Absorbing multi-view inconsistencies from diffusion-generated data. Each hallucinated video is independently sampled from the diffusion model and is therefore not geometrically consistent across viewpoints and frames. Naively training on these videos produces artifacts. To address this, we decouple per-frame variations by introducing two distinct poses: a ‘map pose’ and an ‘LBS pose.’ Specifically, the map pose, which is taken as input for our per-identity networks, is shared across frames with similar poses to ensure consistent Gaussian attributes prediction for a given pose. In contrast, the LBS pose, which deforms the output Gaussians from our per-identity networks, is optimized per-frame to absorb the geometric inconsistencies inherent in the video diffusion model (Sec. 10.2.3.2).

Preserving facial identity under unreliable diffusion outputs. Video diffusion models generate identity-inconsistent faces. We partition the avatar into head and body regions via SMPL-to-FLAME correspondence, mask head pixels from the diffusion-based body supervision, and instead supervise the head through a dedicated multi-view face diffusion model that preserves identity (Sec. 10.2.3.3).

We evaluate AHOY on YouTube videos featuring humans under significant occlusion and on multi-view capture data with comparable occlusion, and demonstrate state-of-the-art reconstruction quality. The resulting avatars are robust enough to be animated with novel poses and composited into 3DGS scenes reconstructed from casual cell-phone video—yielding an end-to-end pipeline from commodity RGB input to photorealistic human-scene rendering.

In summary, our contributions are:

1. A method for reconstructing complete, animatable 3DGS avatars from in-the-wild monocular video under heavy occlusion—unlocking YouTube-scale footage as a

source of 3D human assets.

2. A hallucination-as-supervision approach using identity-finetuned video diffusion and diffusion inversion.
3. A two-stage canonical-to-pose-dependent architecture that allows for learning high-fidelity pose-dependent deformations in 3DGS avatars.
4. A map-pose/LBS-pose decoupling that enables pose-dependent Gaussian maps for high-fidelity avatar learning from multi-view-inconsistent generated data.
5. A head/body split supervision for facial identity preservation.

10.2 METHOD

Given a monocular video in which a human subject is partially occluded, our goal is to reconstruct a complete, animatable 3D Gaussian avatar that can be driven by arbitrary poses. Our pipeline (Fig. 10.2) proceeds in four stages: (1) coarse avatar construction from the partially observed subject (Sec. 10.2.1); (2) hallucinated supervision generation via identity-finetuned video diffusion (Sec. 10.2.2); (3) full avatar learning with pose-dependent Gaussian maps trained on this supervision (Sec. 10.2.3); and (4) animation and scene composition (Sec. 10.2.4).

10.2.1 Coarse Avatar Construction

We first construct a coarse 3DGS avatar from the partially observed subject (Block 1 in Fig. 10.2).

10.2.1.1 Texture Mapping and Inpainting

Given a monocular YouTube video with frames $\{\mathbf{I}_t\}_{t=1}^T$, we first estimate per-frame SMPL parameters $\{(\theta_t, \beta)\}$ using NLF [Sáráandi and Pons-Moll, 2024], and obtain visibility masks $\{\mathbf{M}_t\}$ via SAM 3 [Carion et al., 2025].

For each frame t , DensePose [Güler et al., 2018] provides a mapping $D_t: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ from image pixels to UV coordinates on the SMPL body surface. We define a canonical texture map $\mathbf{T} \in \mathbb{R}^{H_{uv} \times W_{uv} \times 3}$ and an occupancy mask $\mathbf{O} \in \{0, 1\}^{H_{uv} \times W_{uv}}$ that tracks which texels have been filled. For each visible pixel $\mathbf{p}$ (where $\mathbf{M}_t(\mathbf{p}) = 1$), we compute its UV coordinate $\mathbf{u} = D_t(\mathbf{p})$ and apply a first-write-wins rule:

$$\mathbf{T}(\mathbf{u}) \leftarrow \begin{cases} \mathbf{I}_t(\mathbf{p}) & \text{if } \mathbf{O}(\mathbf{u}) = 0, \\ \mathbf{T}(\mathbf{u}) & \text{otherwise,} \end{cases} \quad (10.1)$$

setting $\mathbf{O}(\mathbf{u}) \leftarrow 1$ after each write. By iterating over all T frames, we accumulate a partial canonical texture that covers as much of the body surface as the video observations allow, while avoiding temporal blending artifacts.

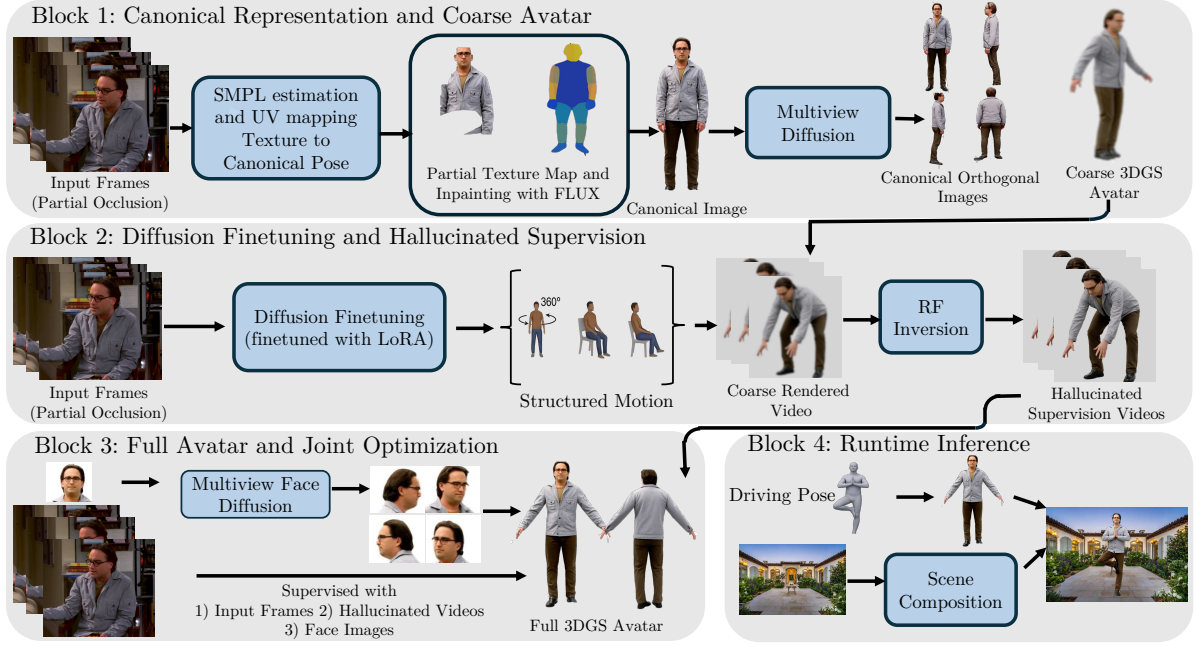


Figure 10.2: **Method overview.** **Block 1** (Sec. 10.2.1): We map observed textures from partially occluded video onto a canonical pose via DensePose UV correspondences, inpaint missing regions with FLUX, and generate multi-view canonical-pose images with multi-view diffusion to supervise a coarse 3DGS avatar using canonical Gaussian maps. **Block 2** (Sec. 10.2.2): We finetune a video diffusion model (Wan 2.2) with LoRA to capture the subject’s identity, render the coarse avatar under structured motion sequences, and refine these renderings via RF-Inversion through the identity-finetuned latent space to produce hallucinated supervision videos. **Block 3** (Sec. 10.2.3): The hallucinated videos supervise a full avatar with pose-dependent Gaussian maps, where per-frame poses and cameras are jointly optimized to absorb multi-view inconsistencies; a separate FLAME-based head path preserves facial identity. **Block 4** (Sec. 10.2.4): At inference, the avatar is driven by novel poses and composited into 3DGS scenes.

We define a fixed front-facing canonical pose θ_* and compute its DensePose map D_* , which assigns a UV coordinate to every pixel of a person standing in θ_* . The canonical-pose image is obtained by an inverse warp [Neverova et al., 2018]: for each canonical pixel $\mathbf{p}_*$ with UV coordinate $\mathbf{u}_* = D_*(\mathbf{p}_*)$, we look up the color from the accumulated texture atlas, $\mathbf{I}_*^{\text{part}}(\mathbf{p}_*) = \mathbf{T}(\mathbf{u}_*)$. Pixels whose UV texels were never filled ($\mathbf{O}(\mathbf{u}_*) = 0$) are marked as missing in a binary mask $\mathbf{M}_*^{\text{miss}}$. A pretrained FLUX [Esser et al., 2024] inpainting model fills in the missing regions:

$$\mathbf{I}_* = \text{Inpaint}(\mathbf{I}_*^{\text{part}}, \mathbf{M}_*^{\text{miss}}), \quad (10.2)$$

yielding a complete canonical-pose image $\mathbf{I}_*$ that serves as the basis for multi-view generation.

10.2.1.2 Multi-View Generation and Coarse 3DGS Avatar

The inpainted canonical image $\mathbf{I}_*$ provides only a single viewpoint. To obtain broader coverage, we generate $V-1=3$ additional canonical-pose views (back, left, right) using a pretrained multi-view diffusion model [Chen et al., 2025b], discarding its 3D reconstruction branch. Together with the frontal view $\mathbf{I}_*^1 = \mathbf{I}_*$, this yields $V=4$ canonical-pose images $\{\mathbf{I}_*^v\}_{v=1}^V$ at known orthogonal cameras $\{\gamma_v\}_{v=1}^V$.

Canonical Gaussian maps. Following [Mir et al., 2025a, Li et al., 2024d], we densely sample N_H points on the SMPL mesh in canonical pose θ_* to define a canonical Gaussian template $\mathcal{G}_*^H$. These are projected from front and back views to create canonical position maps with one-to-one pixel-to-Gaussian correspondence. Unlike the pose-dependent human maps in [Mir et al., 2025a], we use time-invariant canonical maps, as our monocular occluded setting lacks the multi-view coverage needed for pose-conditioned offsets.

StyleUNet prediction. A StyleUNet [Wang et al., 2023] S processes the canonical maps and predicts per-pixel offsets for each Gaussian property from the template, yielding the canonical Gaussian set $\mathcal{G}^H = S(\theta_*)$. Additionally, S predicts per-Gaussian skinning weights $\mathbf{w}_{k,j}^H$ for $j=1, \dots, J$ joints.

LBS deformation. For each frame t with estimated pose θ_t , we deform the canonical Gaussians into posed space via Linear Blend Skinning (LBS) using the predicted skinning weights $\mathbf{w}_{k,j}^H$. We denote the complete deformation as $\mathcal{T}(\mathcal{G}^H; \theta_t)$. Colors and opacities remain as predicted by the StyleUNet.

Training objective. The coarse avatar is supervised by two sources: (i) the V canonical-pose images at known cameras, and (ii) the T observed video frames at estimated poses and observed camera γ^{obs} (the camera associated with NLF reference frame). Let $\hat{\mathbf{I}}_t = \mathcal{R}(\mathcal{T}(\mathcal{G}^H; \theta_t); \gamma^{\text{obs}})$ denote the rendering at frame t . The training loss is:

$$\mathcal{L}_{\text{coarse}} = \sum_{v=1}^V \|\mathbf{I}_*^v - \mathcal{R}(\mathcal{G}^H; \gamma_v)\|_1 + \sum_{t=1}^T \|(\mathbf{I}_t - \hat{\mathbf{I}}_t) \odot \mathbf{M}_t\|_1, \quad (10.3)$$

where $\odot$ denotes element-wise multiplication and $\mathbf{M}_t$ masks out occluded regions. We additionally employ a perceptual loss [Zhang et al., 2018b] with the same masking. This yields a coarse avatar faithful to the visible portions of the subject but with artifacts in unobserved regions.

10.2.2 Hallucinated Supervision Generation

The coarse avatar (Sec. 10.2.1) is supervised by only $V=4$ canonical-pose images and the occluded input frames. This limited coverage means that unobserved regions—the back of the body, inner arms, or any area persistently hidden by the occluder—remain poorly reconstructed, and the avatar cannot generalize to novel poses and camera angles beyond those seen during training. We now proceed to generate hallucinated videos for further supervision (Block 2 in Fig. 10.2).

10.2.2.1 Identity-Specific Video Diffusion Finetuning

A generic video diffusion model lacks knowledge of the subject’s specific appearance, so we finetune one on the input video to build an identity-specific prior. We adapt the identity encoding stage of Dynamic Concepts [Abdal et al., 2025b] to Wan 2.2 [Wan Team, 2025]. Let $\mathbf{v}_\phi$ denote the velocity field of the pretrained rectified flow model. We learn a LoRA update $\mathbf{AB}$ together with a learnable text token $[\mathbf{v}]$ that binds to the target identity in the text encoder’s embedding space. Following [Abdal et al., 2025b], we treat the input video as an *unordered* set of frames—removing temporal information—so that the LoRA captures appearance and identity without entangling motion dynamics. The training objective is the flow matching loss:

$$\mathcal{L}_{\text{id}} = \mathbb{E}_{\tau, \mathbf{z}_0} \left[\left\| (\mathbf{z}_1 - \mathbf{z}_0) - \mathbf{v}_\phi(\mathbf{z}_\tau, \tau; \text{“a } [\mathbf{v}] \text{ person”}) \right\|^2 \right], \quad (10.4)$$

where $\mathbf{z}_0 = \text{Enc}(\mathbf{I}_t)$ is the VAE-encoded latent, $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$ is noise ($\mathbb{I}$ denotes the identity matrix), $\mathbf{z}_\tau = (1 - \tau)\mathbf{z}_0 + \tau\mathbf{z}_1$ is the interpolated latent at flow time $\tau \in [0, 1]$, and the velocity field is parameterized by the LoRA update $\mathbf{W}' = \mathbf{W} + \mathbf{AB}$ applied to the pretrained weights $\mathbf{W}$. After training, the finetuned model $\mathbf{v}_\phi$ generates videos that faithfully preserve the subject’s identity, appearance, and clothing—which we exploit in the next stages to hallucinate occluded regions.

10.2.2.2 Supervision Video Generation

We now use the identity-finetuned model to generate supervision videos that cover the missing body regions. Rather than generating arbitrary motions, we curate a set of N motion sequences designed to systematically expose all body surfaces to the camera while providing the multi-view coverage needed to learn pose-dependent Gaussian maps (Sec. 10.2.3).

Structured Motion (turn). Each turn-based sequence pairs a target body action—e.g., , arms lifting or both arms raised—with a full 360° body rotation. The key observation is that all frames within such a sequence can be treated as views of the *same* pose from different angles: the body action remains approximately constant while the rotation sweeps the viewpoint around the person. A turn with T' frames thus provides in-effect T' -view supervision of a single pose, enabling the model to learn how the Gaussian representation should look from every direction under that specific body configuration.

Structured Motion (static). Certain poses, such as sitting, are incompatible with a full-body turn. For these, we generate multiple videos of the same sitting pose from different camera angles. We then synchronize frames across these videos where the pose matches and supervise each pose with the images from all corresponding videos, providing cross-video multi-view supervision analogous to the within-video coverage of the turn sequences.

Coarse rendering. For each motion sequence n , we animate the coarse avatar with the corresponding poses $\{\theta'_t\}_{t=1}^{T'}$ and render from the observed camera γ^{obs} :

$$\hat{\mathbf{V}}_n = \left\{ \mathcal{R}(\mathcal{T}(\mathcal{G}^H; \theta'_t); \gamma^{\text{obs}}) \right\}_{t=1}^{T'}. \quad (10.5)$$

These renderings capture the correct body pose and silhouette but contain artifacts and missing textures in regions that were occluded in the input video. We refine them through the diffusion latent space below.

10.2.2.3 Hallucination via RF-Inversion

The coarse renderings $\hat{\mathbf{V}}_n$ from above preserve the target body layout but suffer from texture artifacts in unobserved regions. We refine them by passing each video through the latent space of the identity-finetuned model $\mathbf{v}_\phi$, leveraging RF-Inversion [Rout et al., 2025] to replace artifacts with identity-consistent appearance.

Let Enc and Dec denote the VAE encoder and decoder, and let $\mathbf{z}_0 = \text{Enc}(\hat{\mathbf{V}}_n)$ be the encoded coarse video. The forward rectified flow ODE maps this to a noisy latent $\mathbf{z}_1$, and reversing the flow with the identity-finetuned model decodes a hallucinated video:

$$\mathbf{z}_1 = \mathbf{z}_0 + \int_0^1 \mathbf{v}_\phi(\mathbf{z}_\tau, \tau) d\tau, \quad \tilde{\mathbf{z}}_0 = \mathbf{z}_1 + \int_1^0 \mathbf{v}_\phi(\mathbf{z}_\tau, \tau) d\tau, \quad \tilde{\mathbf{V}}_n = \text{Dec}(\tilde{\mathbf{z}}_0). \quad (10.6)$$

The *forward pass encodes the spatial structure*—body pose and silhouette—of the coarse rendering into $\mathbf{z}_1$, while the *reverse pass preserves this layout but steers the appearance* toward the subject’s identity via the LoRA, filling in previously missing regions with plausible textures. Applying this process to all N motion sequences yields the hallucinated video set $\{\tilde{\mathbf{V}}_n\}_{n=1}^N$, which provides multi-view body coverage that serves as supervision for the full avatar (Sec. 10.2.3).

10.2.3 Learning the Full Avatar under Multi-View Inconsistencies

The hallucinated videos $\{\tilde{\mathbf{V}}_n\}_{n=1}^N$ from Sec. 10.2.2 provide dense body coverage but are not perfectly multi-view consistent, as each video is independently sampled from the diffusion model. We address this with (i) a switch from canonical to pose-dependent Gaussian maps, enabled by the in-effect multi-view supervision from the hallucinated sequences, (ii) joint optimization of Gaussian, pose, and camera parameters to absorb inconsistencies, and (iii) a separate FLAME-based head supervision path that preserves facial identity.

10.2.3.1 Pose-Dependent Gaussian Maps

The coarse stage (Sec. 10.2.1) used time-invariant canonical maps because multi-view supervision for multiple poses was unavailable. The hallucinated videos now provide this coverage: in a turn-based sequence, all T' frames observe approximately the same body pose from different viewpoints, yielding in-effect T' -view supervision for that pose. For static-pose sequences (e.g., sitting), we synchronize frames across videos rendered from different angles where the same pose occurs, providing cross-video multi-view supervision. This dense coverage enables us to upgrade to pose-dependent Gaussian maps, following [Mir et al., 2025a, Li et al., 2024d]: we project posed SMPL

vertices onto front and back UV maps for each frame, and a re-initialized StyleUNet S takes these pose-dependent maps as input and predicts per-pixel Gaussian property offsets from the canonical template, yielding $\mathcal{G}_n^H = S(\theta_n^{\text{map}})$.

A crucial distinction arises between two roles of the pose: the *map pose* θ_n^{map} , which is the input to the StyleUNet and remains *fixed* across all frames of sequence n , and the *LBS pose* $\theta_{n,t}^{\text{lbs}}$, which deforms the predicted Gaussians into posed space and varies per frame. For a turn-based sequence, the map pose encodes the target body configuration (e.g., arms raised) so that S produces a single set of Gaussians representing that pose; the per-frame LBS pose then handles the body rotation, mapping these Gaussians to the correct 3D orientation for each viewpoint.

10.2.3.2 Joint Optimization with Pose and Camera Correction

Since the hallucinated videos are occlusion-free, we estimate SMPL body poses using NLF [Sáráandi and Pons-Moll, 2024] and assume cameras in the NLF reference frame. These estimates initialize both the map pose θ_n^{map} (for pose-dependent input maps) and the per-frame LBS pose $\theta_{n,t}^{\text{lbs}}$. We optimize S jointly with per-frame LBS pose corrections $\{\Delta\theta_{n,t}\}$ and camera corrections $\{\Delta\gamma_{n,t}\}$. Let $\tilde{\mathbf{I}}_{n,t}$ denote frame t of hallucinated video n . We exclude head pixels via a mask $\mathbf{M}_{n,t}^{\text{head}}$ obtained from Sapiens [Khiredkar et al., 2024] and SAM 3 [Carion et al., 2025], as diffusion-generated faces are unreliable. The body loss is:

$$\mathcal{L}_{\text{body}} = \sum_{n,t} \|(\tilde{\mathbf{I}}_{n,t} - \hat{\mathbf{I}}_{n,t}) \odot (1 - \mathbf{M}_{n,t}^{\text{head}})\|_1, \quad (10.7)$$

where $\hat{\mathbf{I}}_{n,t} = \mathcal{R}(\mathcal{T}(\mathcal{G}_n^H; \theta_{n,t}^{\text{lbs}} + \Delta\theta_{n,t}); \gamma^{\text{obs}} + \Delta\gamma_{n,t})$ is the rendered image with corrected LBS pose and camera. The map pose θ_n^{map} is shared across all frames of sequence n , so every hallucinated view supervises the same pose-dependent Gaussian map. The LBS pose corrections $\Delta\theta_{n,t}$ absorb per-frame multi-view inconsistencies from the diffusion-generated data, introducing a deliberate divergence between the map pose and the LBS pose that allows the canonical Gaussians to remain consistent while the deformation adapts to each frame.

We retain observed-frame supervision from the input video, analogous to the second term of Eq. (10.3):

$$\mathcal{L}_{\text{obs}} = \sum_t \|(\mathbf{I}_t - \hat{\mathbf{I}}_t^{\text{obs}}) \odot \mathbf{M}_t\|_1,$$

where $\hat{\mathbf{I}}_t^{\text{obs}}$ is rendered with the original estimated poses and cameras.

10.2.3.3 Head Identity Preservation via FLAME

Head regions in the Wan-generated videos are unreliable for facial identity. Instead, we leverage the multi-view face diffusion model from CAP4D [Taubner et al., 2025] to generate identity-consistent head images $\{\mathbf{I}_p^{\text{face}}\}_{p=1}^P$ from P viewpoints with known cameras $\{\gamma_p^{\text{face}}\}$.

Using the SMPL-to-FLAME [Li et al., 2017] vertex correspondence, we partition the canonical Gaussians into head and body regions:

$$\mathcal{G}^{\text{head}} = \{(\mu_k, \dots) \in \mathcal{G}^H \mid k \in \mathcal{H}_{\text{head}}\}, \quad (10.8)$$

where $\mathcal{H}_{\text{head}} \subset \{1, \dots, N_H\}$ is the set of Gaussian indices corresponding to the SMPL head region. For head supervision, we deform only these Gaussians using FLAME expression parameters ξ_p :

$$\mathcal{L}_{\text{head}} = \sum_{p=1}^P \|\mathbf{I}_p^{\text{face}} - \mathcal{R}(\mathcal{T}_{\text{FLAME}}(\mathcal{G}^{\text{head}}; \xi_p); \gamma_p^{\text{face}})\|_1, \quad (10.9)$$

where $\mathcal{T}_{\text{FLAME}}$ applies FLAME-based deformation to the head Gaussians while leaving their canonical positions shared with the full body representation.

10.2.3.4 Total Objective

The full avatar is trained with the combined loss:

$$\mathcal{L} = \lambda_{\text{body}} \mathcal{L}_{\text{body}} + \lambda_{\text{obs}} \mathcal{L}_{\text{obs}} + \lambda_{\text{head}} \mathcal{L}_{\text{head}} + \lambda_{\text{per}} \mathcal{L}_{\text{per}} + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}}, \quad (10.10)$$

where $\mathcal{L}_{\text{per}}$ is a perceptual loss [Zhang et al., 2018b] and $\mathcal{L}_{\text{reg}}$ regularizes the skinning weights toward SMPL-derived values. The weighting coefficients λ_{body} , λ_{obs} , λ_{head} , λ_{per} , and λ_{reg} balance the respective terms. Because the hallucinated sequences span a wide range of body configurations—standing, sitting, arms raised, turning—the training pose distribution is broad enough that the learned pose-dependent Gaussian maps generalize well to novel driving poses at inference time.

10.2.4 Animation in 3D Scenes

The preceding stages produce a complete, animatable 3DGS avatar that can be driven by arbitrary SMPL poses via LBS with pose-dependent Gaussian maps (Sec. 10.2.3.1). Following AHA! [Mir et al., 2025b], we animate this avatar within reconstructed 3DGS scenes. To prevent out-of-distribution artifacts, we apply PCA-based pose projection at inference, following [Li et al., 2024d], constraining novel driving poses to the training distribution. Given a scene reconstructed as a static Gaussian set $\mathcal{G}^S$ from casual cell-phone video [Kerbl et al., 2023] and a driving motion of T'' frames $\{\theta_t\}_{t=1}^{T''}$, we deform the avatar at each timestep and render $\mathbf{I}_{\text{out}} = \mathcal{R}(\mathcal{T}(\mathcal{G}_t^H; \theta_t) \cup \mathcal{G}^S; \gamma)$, where $\mathcal{G}_t^H = S(\theta_t)$ uses the (PCA-projected) pose-dependent Gaussian maps, producing photorealistic novel-view videos of the reconstructed human animated within the target environment.

10.3 IMPLEMENTATION DETAILS

We first describe the network architecture, then detail the training procedure for each of the three pipeline stages: coarse avatar reconstruction, identity-specific LoRA finetuning, and full pose-dependent avatar optimization.

10.3.1 Network Architecture

StyleUNet. We employ the same StyleUNet architecture used in Ch. 7 (see Sec. 7.3.1 and Fig. 7.7 for the full architectural description). In AHOY, we instantiate a single StyleUNet S that takes position maps as input and predicts per-pixel offsets from the canonical Gaussian template via three output heads: color ($C_{\text{out}} = 3$), position ($C_{\text{out}} = 3$, scaled by 0.05), and attributes ($C_{\text{out}} = 8$: opacity, scales, and rotation quaternion). The StyleUNet additionally predicts per-Gaussian skinning weights $\mathbf{w}_{k,j}^H$ for LBS deformation. In the coarse stage (Sec. 10.2.1), S receives time-invariant canonical position maps; in the full avatar stage (Sec. 10.2.3), S is re-initialized to receive pose-dependent position maps.

LoRA. For identity-specific video diffusion finetuning (Sec. 10.2.2.1), we apply LoRA with rank $r = 128$ to the Wan 2.2 [Wan Team, 2025] velocity field. A learnable text token $[v]$ is jointly optimized to bind the subject’s identity in the text encoder’s embedding space.

10.3.2 Training Procedure

Stage 1: Coarse avatar. Since the coarse avatar uses a time-invariant canonical position map (Sec. 10.2.1), we first train the StyleUNet for approximately 2,000 iterations on this canonical map with a target of zero offsets, forcing the network to learn the identity mapping before any deformation. We use Adam [Kingma and Ba, 2015] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and an initial learning rate of 10^{-4} . We apply a cosine annealing schedule:

$$\text{lr}(i) = \text{lr}_{\text{init}} \cdot \left[\frac{1}{2} (\cos(\pi \cdot i / N_{\text{iter}}) + 1) (1 - \eta_{\text{min}}) + \eta_{\text{min}} \right], \quad (10.11)$$

where i is the current iteration, N_{iter} is the total number of iterations, and $\eta_{\text{min}} = 0.05$ ensures the learning rate never drops below 5% of the initial value. The coarse StyleUNet is trained for 400,000 iterations with a batch size of 1; at each iteration, we randomly sample one camera view and render a 512×512 patch. Loss weights: $\lambda_1 = 1.0$, $\lambda_{\text{per}} = 0.1$, $\lambda_{\text{reg}} = 0.005$.

Stage 2: Identity-specific LoRA finetuning. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and weight decay 0.01. The learning rate is 10^{-4} for the LoRA parameters and 10^{-5} for the learnable text token. We apply gradient clipping with a maximum norm of 0.05. The LoRA and learnable text token $[v]$ are trained for 600 iterations with a batch size of 8. Dropout regularization with rate 0.8 is applied to the LoRA $\mathbf{B}$ matrices, and text prompt dropout with probability 0.1.

Stage 3: Full avatar. The canonical Gaussians are initialized from the final coarse avatar. Since Stage 3 uses pose-dependent position maps (Sec. 10.2.3.1), we pretrain the re-initialized StyleUNet for $\sim 2,000$ iterations on randomly sampled pose-dependent maps, with the coarse avatar’s final Gaussians as the target, forcing the network to directly predict the coarse avatar for any input pose before refinement. Pose and camera corrections $\{\Delta\theta_{n,t}\}$, $\{\Delta\gamma_{n,t}\}$ are initialized to zero. We use Adam with the same cosine annealing schedule (Eq. (10.11)). The full avatar StyleUNet is trained

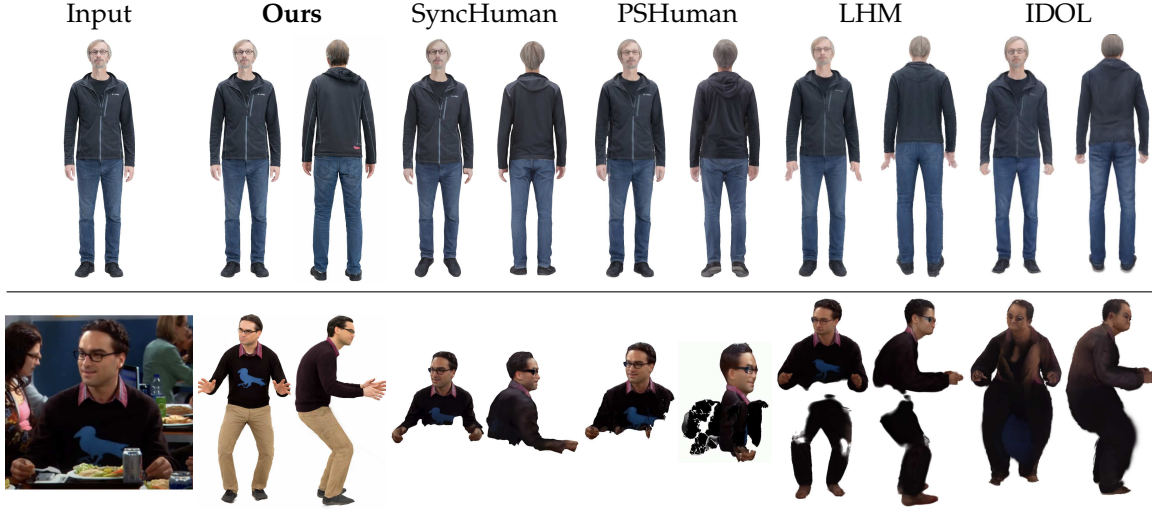


Figure 10.3: **Static reconstruction comparison (Zoom In)**. Top: canonical-pose input. Bottom: occluded input. Our animatable avatar is posed to match the target view.

for 500,000 iterations with a batch size of 1 and 512×512 patches. Loss weights: $\lambda_{\text{body}} = 1.0$, $\lambda_{\text{obs}} = 1.0$, $\lambda_{\text{head}} = 0.5$, $\lambda_{\text{per}} = 0.1$, $\lambda_{\text{reg}} = 0.005$.

10.4 EXPERIMENTS

We evaluate AHOY along four axes: *animation on YouTube* (Sec. 10.4.2); *static reconstruction on YouTube* (Sec. 10.4.3); *multi-view evaluation on BEHAVE* with ground-truth side views measuring novel-view and novel-pose quality under occlusion (Sec. 10.4.4); and *component analysis* (Sec. 10.4.5). We further demonstrate animation within photorealistic 3DGS scenes (Sec. 10.4.6).

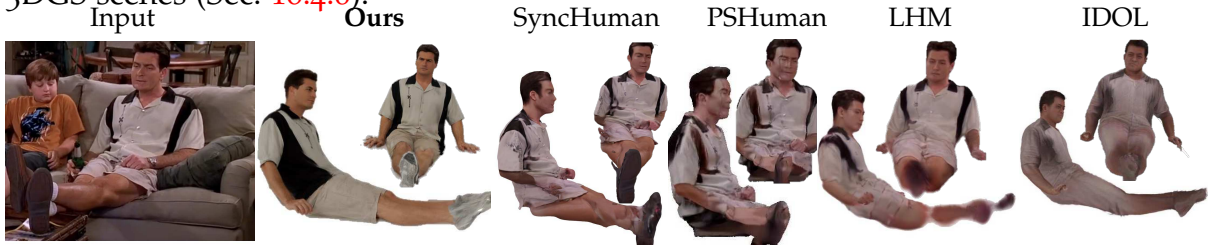


Figure 10.4: **Reconstruction under difficult fully visible poses**. Static methods struggle to reconstruct humans under difficult poses, while our method allows reposing of 3DGS avatars to match the target pose.

10.4.1 Experimental Setup

Datasets. We evaluate on two benchmarks that provide naturally occluded human subjects. **BEHAVE** [Bhatnagar et al., 2022] contains multi-view captures (4 Kinect cameras) of humans interacting with objects, producing significant body occlusion. We

Table 10.1: **Static reconstruction on YouTube videos.** We compare under occluded and canonical-pose input settings.

Method	Occluded			Canonical		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
PSHuman	19.47	0.811	0.197	21.88	0.858	0.145
SyncHuman	19.82	0.819	0.189	22.03	0.863	0.140
LHM	19.12	0.803	0.207	21.18	0.845	0.160
IDOL	17.31	0.771	0.241	19.78	0.819	0.189
Ours	22.01	0.881	0.109	23.05	0.892	0.098

evaluate on 8 sequences. The multi-view ground truth enables rigorous quantitative evaluation. **YouTube videos.** We collect 50 videos of humans in diverse everyday activities with significant occlusion. For these sequences, ground-truth multi-view images are unavailable, so we compare against held-out frames from the original video. **Metrics.** We report PSNR, SSIM [Wang et al., 2004], and LPIPS [Zhang et al., 2018b].

Baselines. Our method is video-based, while all baselines are single-image methods. We consider recent animatable avatar methods with public code: LHM [Qiu et al., 2025a] and IDOL [Zhuang et al., 2025]. We evaluate baselines under two input settings: (1) receiving the original occluded frame (first video frame) directly, and (2) receiving the unoccluded canonical-pose image generated by our pipeline, giving them the best possible input. We additionally compare with static reconstruction methods PSHuman [Li et al., 2025] and SyncHuman [Chen et al., 2025b], which do not produce animatable avatars. We note that AdaHuman [Huang et al., 2025], Dream-Lift-Animate [Buehler et al., 2026], MoGA [Dong et al., 2025], and MEAT [Wang et al., 2025g] are plausible baselines, but none have publicly available code.

10.4.2 Animation on YouTube

We compare with LHM [Qiu et al., 2025a] and IDOL [Zhuang et al., 2025]—the only baselines that produce animatable avatars—under two input settings (Tab. 10.2, Fig. 10.5). We animate reconstructed avatars to novel poses and compare against held-out video frames. **Canonical-pose input.** To provide a fairer comparison, we give baselines the unoccluded canonical-pose images generated by our pipeline (Sec. 10.2.1.1). Even with complete input, our full pipeline outperforms these methods (Fig. 10.5, top). **Occluded input.** When baselines receive the original occluded video frames, the realism of their avatars drops significantly (Fig. 10.5, bottom).



Figure 10.5: **Animation comparison (Zoom In).** Top: canonical-pose input. Bottom (below line): occluded input. Ours produces higher-fidelity avatars in both settings.

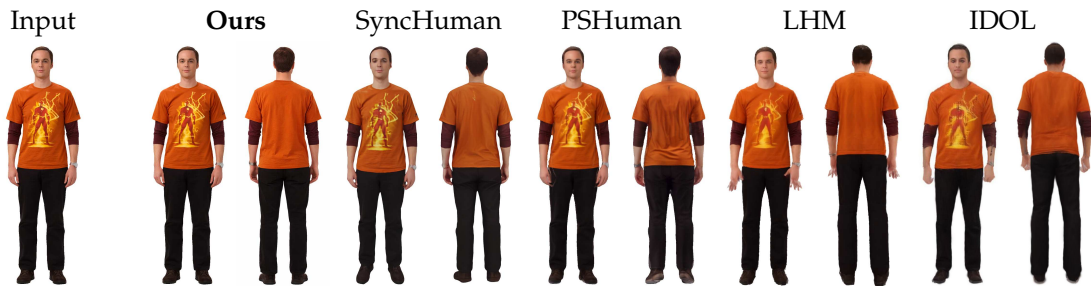


Figure 10.6: **Additional static reconstruction comparison.** All methods receive an unoccluded canonical-pose image. Our animatable avatar is posed to match the target view.

10.4.3 Static Reconstruction on YouTube

We compare with PSHuman [Li et al., 2025], SyncHuman [Chen et al., 2025b], LHM [Qiu et al., 2025a], and IDOL [Zhuang et al., 2025] on YouTube videos under two input settings (Tab. 10.1, Figs. 10.3–10.6). Each method reconstructs from its respective

Table 10.2: **Animation comparison on YouTube videos.** We animate reconstructed avatars to novel poses and compare against held-out video frames.

Method	Occluded input			Canonical-pose input		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
LHM	19.03	0.821	0.181	20.17	0.842	0.159
IDOL	16.52	0.769	0.231	18.52	0.808	0.191
AHOY (Ours)	22.81	0.887	0.107	22.83	0.888	0.106

input, and all reconstructions are rendered from the input viewpoint and compared against the original video frame. **Occluded input.** Under occlusion, all baselines fail to produce complete reconstructions, while our method generates a full avatar that can be accurately posed to match the target view. **Canonical-pose input.** To provide a fairer comparison, we give all baselines the unoccluded canonical-pose images generated by our pipeline. Even with complete input, our method outperforms all baselines. In Fig. 10.4, we demonstrate that static methods struggle to reconstruct humans under difficult poses, while our method allows reposing of avatars to match the target pose.

10.4.4 BEHAVE Evaluation

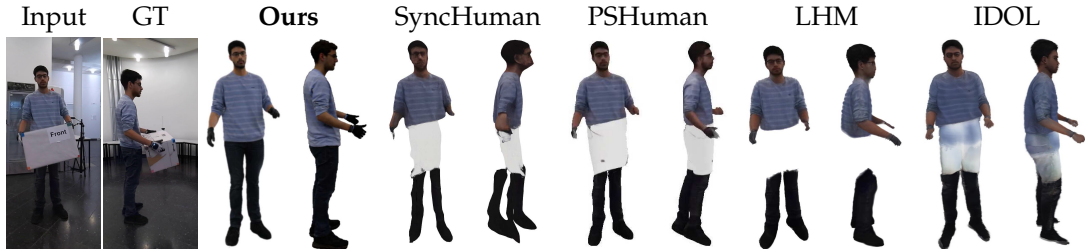


Figure 10.7: **Novel-view evaluation on BEHAVE.** We render each reconstruction from held-out BEHAVE side cameras and compare against the ground-truth side views.

We evaluate all methods on BEHAVE (Tab. 10.3, Fig. 10.7). All methods receive the occluded front-camera view as input. For *novel-view* evaluation, we render each reconstruction from held-out side cameras and compare against ground-truth side views. For *novel-pose* evaluation, we animate the reconstructed avatars to unseen ground-truth SMPL-X poses and render from held-out side cameras. Our method significantly outperforms all baselines in both settings.

10.4.5 Ablation Study

We ablate each key component of our pipeline on BEHAVE to isolate its contribution, evaluating both novel-view and novel-pose settings as in Sec. 10.4.4 (Tab. 10.4, Fig. 10.8).

Table 10.3: **Quantitative evaluation on BEHAVE.** Novel-view and novel-pose settings.

Method	Novel View			Novel Pose		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
PSHuman	18.47	0.812	0.198	<i>n/a</i>		
SyncHuman	19.34	0.831	0.172	<i>n/a</i>		
LHM	18.21	0.807	0.203	16.93	0.774	0.238
IDOL	17.38	0.783	0.231	15.87	0.748	0.267
Ours	24.12	0.905	0.090	22.81	0.889	0.110

Table 10.4: **Ablation study on BEHAVE.** Each row removes one component from the full pipeline. *Novel View*: render from held-out side cameras. *Novel Pose*: animate to unseen ground-truth SMPL-X poses and render from held-out side cameras.

Configuration	Novel View			Novel Pose		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
AHOY (full)	24.12	0.905	0.090	22.81	0.889	0.110
(A) Coarse avatar only	16.10	0.753	0.258	14.30	0.718	0.301
(B) w/o RF-Inversion	21.10	0.862	0.143	19.40	0.838	0.168
(C) w/o map/LBS decoupling	22.60	0.886	0.118	20.60	0.856	0.152
(D) w/o head/body split	23.60	0.898	0.098	22.20	0.882	0.120

(A) Coarse avatar only. We skip the hallucination and full avatar stages, using only the coarse canonical-map avatar (Sec. 10.2.1). This removes pose-dependent appearance and the diffusion-based supervision for occluded regions. **(B) Without RF-Inversion.** We skip RF-Inversion and supervise the avatar without any new hallucinated videos of the person in new poses; this results in significant artifacts in unobserved regions (Sec. 10.2.2.3). **(C) Without map-pose/LBS-pose decoupling.** We do not use pose-dependent maps for StyleUNet training and do not optimize LBS deformation, removing the decoupling that absorbs multi-view inconsistencies (Sec. 10.2.3.2). **(D) Without head/body split.** We supervise the entire avatar, including the head, using the hallucinated Wan videos, without the separate head path (Sec. 10.2.3.3).

10.4.6 Animation in 3D Scenes

We demonstrate that the avatars reconstructed by AHOY are robust enough to be animated with novel poses and composited into 3DGS scenes reconstructed from casual cell-phone video (Fig. 10.9).

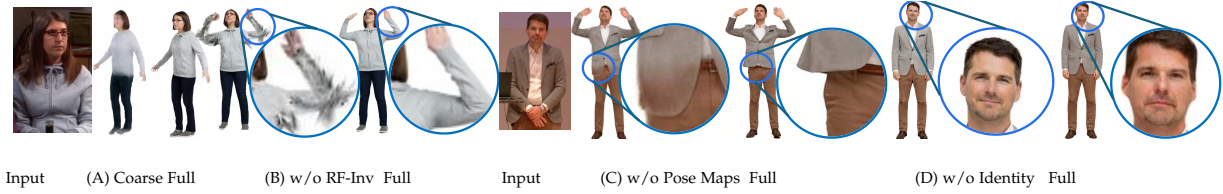


Figure 10.8: **Visual ablations.** Each pair shows the ablated result (w/o) and the full pipeline. (A) Without hallucination, the coarse-only avatar lacks detail in unobserved regions. (B) Without RF-Inversion, hallucinated supervision retains rendering artifacts. (C) Without map/LBS decoupling, multi-view inconsistencies cause blurring. (D) Without the head/body split, facial identity is lost.



Figure 10.9: **Animation in 3DGS scenes.** Avatars from occluded YouTube video, animated and composited into 3DGS scenes from casual cell-phone video.

10.5 CONCLUSION

We have presented AHOY, a method that reconstructs complete, animatable 3D Gaussian avatars from in-the-wild monocular video despite heavy occlusion—unlocking YouTube-scale footage as a source of 3D human assets. Our key insight is to use an identity-finetuned video diffusion model as a bridge between imperfect coarse renderings and dense, realistic supervision: by embedding coarse avatar renderings into the model’s latent space via RF-Inversion and decoding them back, the identity prior fills in previously unobserved regions while preserving body layout. The resulting avatars are robust enough to be animated with novel poses and composited into 3DGS scenes reconstructed from casual cell-phone video. Our method inherits limitations from its diffusion backbone: the identity-finetuned model may hallucinate plausible but incorrect details for body regions that are never observed, and the quality of the final avatar is bounded by the fidelity of the video diffusion model. Additionally, our pipeline requires several stages of optimization and diffusion inference, making it substantially slower than feed-forward methods. Addressing these limitations suggests several promising directions: scaling to stronger video diffusion models as they improve and incorporating geometric consistency constraints into the hallucination stage. We hope that AHOY demonstrates that occlusion—long treated as a nuisance to be avoided—can be overcome, opening the door to learning from the vast diversity of human appearance captured in everyday video.

CONCLUSION AND FUTURE WORK

This thesis has explored a fundamental question in computer vision and graphics: how should we represent 3D humans and their environments to enable realistic interaction? The first part leverages classical mesh and point cloud representations for human appearance, capture, and motion synthesis. We then asked whether the geometric capabilities that make classical representations so effective—navigation, collision detection, path planning, shadow casting—can be recovered entirely from 3D Gaussian fields. The second part demonstrates that they can.

In Part I, we addressed foundational problems of human appearance, capture, and motion synthesis using classical mesh and point cloud representations. These representations provide explicit geometry—surfaces, signed distance fields, walkable regions—that enables well-understood algorithms for physics, collision detection, and path planning. In Ch. 4, we proposed Pix2Surf, the first method to map texture from clothing images to 3D garments worn atop SMPL, using a SMPL+G formulation that generalizes to applications such as modeling pose-dependent deformations [Patel et al., 2020] and modeling size-dependent deformations. In Ch. 5, we presented HPS, the first method to recover the full 3D pose of humans registered with large 3D scenes (300–2500 m^2) using only body-mounted sensors, capturing a dataset consisting of 7 subjects and more than 3 hours of motion. In Ch. 6, we presented CoHMo, a method to synthesize long-term human motion across diverse 3D scenes by repeatedly synthesizing motion in a canonical coordinate frame.

In Part II, we showed that 3DGS can replace classical representations across the full animation pipeline. In Ch. 7, we presented InteractSplat, an end-to-end pipeline that reconstructs separate controllable human and object models from multi-view video and animates them together in diverse 3DGS environments, enabling long-horizon sequences involving locomotion, object manipulation, and collision avoidance. In Ch. 8, we presented SALA, the first method to synthesize human interactions in diverse 3D environments using 3DGS as the sole underlying 3D representation, enabling photorealistic free-viewpoint rendering of human–scene interactions. In Ch. 9, we presented RAGA, a ray-traced Gaussian shadow casting formulation that computes physically plausible shadows entirely in Gaussian space at interactive rates. In Ch. 10, we presented AHOY, a method that reconstructs complete, animatable 3DGS avatars from in-the-wild YouTube video despite heavy occlusion, unlocking YouTube-scale footage as a source of 3D human assets.

Taken together, these contributions demonstrate that the choice of 3D representation fundamentally shapes what is achievable in human–scene interaction: classical representations provide the geometric machinery for robust capture and motion synthesis, while 3DGS enables photorealistic rendering at a quality unattainable with classical pipelines. By developing methods across both representation families, we

show that 3DGS can serve as a complete replacement for classical representations in the human–scene animation pipeline—from navigation and locomotion to contact modeling and shadow casting—while delivering superior visual fidelity. We hope that this work takes us one step closer to *Real Virtual Humans*.

11.1 KEY INSIGHTS

In this section, we present the key insights that allow us to make progress on the problems addressed in this thesis.

11.1.1 Pix2Surf

In Pix2Surf, our key idea is to use existing techniques from avatar construction from images—specifically matching silhouettes of meshes with images—to extract texture from clothing images. Another key insight is to leverage neural networks to map texture from images to UV maps. Our key idea here is to formulate this problem in terms of coordinates instead of images. As the neural network has to reason about coordinates instead of images, it simplifies the problem significantly and allows for easier learning.

11.1.2 HPS

In HPS, our key insight is to solve the problem of pose estimation and localization entirely using body-mounted sensors without the use of external cameras. This allows our method to capture humans without the restriction of space imposed by the field of view of external cameras or by occlusions that are common when using external cameras.

A problem encountered in this setup is that the camera localization fails badly and quite often. As a consequence, the mesh obtained from one time frame to another displays significant discontinuities. Our key insight to resolve this problem is to leverage localization obtained using IMUs to plug in the time frames where camera localization fails. This allows us to obtain smooth motions.

Another problem with this setup is that the pose obtained using IMUs has an orientation significantly skewed from the natural orientation of motion. We resolve the problem by using ideas from visual-inertial odometry to align the IMU orientation with the camera localization trajectory.

11.1.3 CoHMo

In Ch. 6, our key insight is to use scene-agnostic human motion data to synthesize human motion in 3D scenes. Unlike human motion data registered to 3D scenes, motion data without scenes is abundant and relatively easily captured. In this chapter,

we develop a method that allows us to abstract away most detail about the geometry in 3D scenes and use this representation to synthesize human motion registered to 3D scenes.

Another insight presented in this chapter is to represent actions in 3D scenes with keypoints. This provides animators with significant control over the motion synthesis process. The animator can specify the exact point in the 3D scene where the synthesized person should touch the scene or the exact place in the scene where the person should sit.

11.1.4 InteractSplat

In InteractSplat, our key insight is to decouple human and object reconstruction, enabling independent pose control and, for the first time, synthesis of novel human–object interactions in varied 3DGS environments. A central technical idea is to represent dynamic objects using canonical Gaussian maps—2D surface manifolds that yield sharper textures than volumetric 3D Gaussians, especially under heavy mutual occlusion. Another key insight is the use of dynamic 2D maps that update after every interaction, enabling collision-free path planning in changing environments.

11.1.5 SALA

In Ch. 8, our key insight is that 3D Gaussian Splatting can serve as the sole 3D representation for human–scene animation, replacing meshes and point clouds entirely. This is non-trivial because 3DGS provides none of the geometric quantities that existing animation pipelines rely on—no navigable surfaces, no signed distance fields, no walkable regions. We address this by extracting all necessary structures directly from raw Gaussian fields: opacity culling and PCA-aligned projection for navigation, an egocentric walkability grid for locomotion, and soft nearest-neighbour distances for contact refinement.

11.1.6 RAGA

In Ch. 9, our key insight is that shadow casting for 3DGS avatars can be formulated as a set of independent absorption events along rays cast toward light sources, where each occluding Gaussian contributes an attenuation weight derived from the exact line integral of its opacity profile. This weight captures how the ray traverses the occluder—a ray that merely grazes a Gaussian accumulates little opacity, while one passing through its major axis accumulates much more. This principled formulation avoids the binary hit tests used by prior ray-based methods and produces temporally stable shadows.

11.1.7 AHOY

In Ch. 10, our key insight is to use an identity-finetuned video diffusion model as a bridge between imperfect coarse renderings and dense, realistic supervision. By embedding coarse avatar renderings into the diffusion model’s latent space and decoding them back, the identity prior fills in previously unobserved body regions while preserving body layout. A second key insight is the canonical-to-pose-dependent progression: we first learn a pose-independent avatar from sparse observations, then use hallucinated multi-view videos to upgrade it to a full pose-dependent model that captures how clothing changes with body configuration.

11.2 FUTURE WORK

The methods presented in this thesis open up several research directions, ranging from concrete extensions of the work presented here to more speculative long-term questions.

11.2.1 Scaling Motion Synthesis with Video Diffusion Models

Current motion synthesis approaches, including CoHMo (Ch. 6), struggle to generate diverse, long-horizon behaviors in complex 3D scenes. For instance, no existing method can generate diverse locomotion styles (fast walks, running, hopping) along a controlled path, nor varied manipulation motions such as grabbing at different heights or walking upstairs. We contend that this is fundamentally a data problem.

There is a strong analogy between modeling human language and human motion. The underlying vocabulary (words vs. motion segments) is typically represented using a latent space, and transformers generate next tokens autoregressively. Post-training methods are also shared: once an autoregressive model is obtained, it can be guided using reinforcement learning (RL) to satisfy specific constraints—a path in motion synthesis, or a question in language modeling. Yet all current motion synthesis research relies on the AMASS dataset or its derivatives, totalling roughly 40 hours of motion data. State-of-the-art language models, by contrast, are trained on trillions of tokens.

If the underlying structure of these two problems is similar, motion synthesis models should also benefit from dataset scaling. Current video generation models such as Wan 2.2 can produce photorealistic videos of humans performing a wide range of actions, and pose estimation methods under controlled settings (no camera motion, no occlusion) produce near-perfect results, albeit with some jitter. Integrating this pipeline with physics-based stabilisers could generate motion data at scale—perhaps a million or more hours. Such a dataset would enable the study of scaling laws for motion synthesis, particularly in combination with RL for controlled synthesis in 3D scenes.

11.2.2 Bridging Neural and Classical Rendering for Human–Scene Interaction

In Ch. 8, we demonstrated that neural rendering alone, without recourse to classical rendering, can generate photorealistic human–scene interaction. While the resulting photorealism far exceeds that of methods relying on classical rendering, neural rendering currently lacks many properties of classical pipelines—such as explicit material decomposition, physically-based light transport, and composability. An important future direction is to bridge this gap, for instance by incorporating learnings from inverse rendering techniques for general objects and scenes into the 3DGS avatar reconstruction pipeline.

11.2.3 Neural Rendering for Human–Scene Datasets

High-quality 3D human datasets are critical for advancing pose estimation and motion understanding. Synthetic datasets such as BEDLAM 2.0 have demonstrated that they can yield better performance than real datasets, primarily because they can be scaled to a much larger degree than real-world captures. However, these datasets suffer from a fundamental limitation: a significant domain gap in the rendered images.

This gap could be bridged by replacing classical rendering pipelines entirely with neural rendering. This would entail scaling the techniques developed in Ch. 8. In that work, we use only five human identities for 3DGS avatars. Creating 3DGS avatars for a larger number of identities—either through multi-view capture or by improving the photorealism of our YouTube reconstruction method (Ch. 10)—and using scenes from open-source community platforms would enable the creation of more photorealistic synthetic datasets while retaining ground-truth annotations from controlling the underlying parameters.

11.2.4 Simulators for Robotics and Autonomous Driving

Another area that could benefit from neural rendering is simulator construction for robotics and autonomous driving. In Ch. 7, we showed that human–object interactions in 3D scenes achieve much higher photorealism when rendered using neural rather than classical techniques. By scaling to multiple scenes, simulators requiring human–scene or human–object interaction could be built entirely using neural rendering. Agents trained in such environments could in principle be deployed more easily in the real world, because the domain gap between these simulated worlds and reality would be significantly reduced.

11.2.5 Rethinking the Role of 3D Structure in Visualization

All 3D representations used in this thesis—meshes, point clouds, 3DGS, NeRFs—implicitly assume that the world has a 3D structure. We query this structure from a

novel viewpoint to generate renderings, and because there is underlying 3D geometry, the renderings remain multi-view consistent. However, recent video diffusion models, trained entirely on 2D videos, demonstrate that they learn some notion of 3D structure purely from video data. Models such as Google’s Genie can be queried to generate visualizations of a world by moving a virtual camera, without maintaining any explicit 3D representation.

Currently, the latent spaces of such models cannot be queried simultaneously from multiple viewpoints or modified with explicit constraints—for instance, there is no way to generate a human walking along a constrained path specified in Cartesian coordinates. If these limitations were overcome, enabling for example multiplayer renderings from simultaneous viewpoints or editing motion and lighting via explicit constraints, there would be no need for explicit 3D structure. All properties that classical computer graphics encodes geometrically would instead be learned from data. Whether this is achievable remains an open question, but it represents a fundamental shift worth investigating.

BIBLIOGRAPHY

- R. Abdal, Y. Qin, and P. Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- R. Abdal, Y. Qin, and P. Wonka. Image2stylegan++: How to edit the embedded images? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- R. Abdal, W. Yifan, Z. Shi, Y. Xu, R. Po, Z. Kuang, Q. Chen, D.-Y. Yeung, and G. Wetzstein. Gaussian shell maps for efficient 3d human generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- R. Abdal, O. Patashnik, E. Deyneka, H. Chen, A. Siarohin, S. Tulyakov, D. Cohen-Or, and K. Aberman. Zero-shot dynamic concept personalization with grid-based LoRA. In *ACM SIGGRAPH Asia Conference Proceedings*, 2025a.
- R. Abdal, O. Patashnik, I. Skorokhodov, W. Menapace, A. Siarohin, S. Tulyakov, D. Cohen-Or, and K. Aberman. Dynamic concepts personalization from single videos. In *ACM SIGGRAPH Conference Proceedings*, 2025b.
- E. Aksan, M. Kaufmann, and O. Hilliges. Structured prediction helps 3d human motion modelling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. First two authors contributed equally.
- E. Aksan, P. Cao, M. Kaufmann, and O. Hilliges. A spatio-temporal transformer for 3d human motion prediction. *arXiv preprint arXiv:2004.08692*, 2020.
- A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- T. Alldieck, M. Magnor, W. Xu, C. Theobalt, and G. Pons-Moll. Detailed human avatars from monocular video. In *International Conference on 3D Vision (3DV)*, 2018a.
- T. Alldieck, M. Magnor, W. Xu, C. Theobalt, and G. Pons-Moll. Video based reconstruction of 3D people models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b.
- T. Alldieck, M. Magnor, B. L. Bhatnagar, C. Theobalt, and G. Pons-Moll. Learning to reconstruct people in clothing from a single RGB camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019a.
- T. Alldieck, G. Pons-Moll, C. Theobalt, and M. Magnor. Tex2shape: Detailed full human body geometry from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019b.

- T. Alldieck, H. Xu, and C. Sminchisescu. imGHUM: Implicit generative models of 3d human shape and articulated pose. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- B. Allen, B. Curless, Z. Popović, and A. Hertzmann. Learning a correlated model of identity and pose-dependent body shape variation for real-time synthesis. In *ACM SIGGRAPH Conference Proceedings*. Eurographics Association, 2006.
- D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: Shape completion and animation of people. *ACM Transactions on Graphics*, 2005.
- R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- Aria. Project aria. <https://about.fb.com/realitylabs/projectaria/>, 2020.
- Azure-Kinect. Microsoft azure kinect. https://en.wikipedia.org/wiki/Azure_Kinect, 2020.
- G. Balakrishnan, A. Zhao, A. V. Dalca, F. Durand, and J. Guttag. Synthesizing images of humans in unseen poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- A. O. Bălan and M. J. Black. The naked truth: Estimating body shape under clothing. In *European Conf. on Computer Vision*, 2008.
- D. Baraff and A. Witkin. Large steps in cloth simulation. In *ACM SIGGRAPH Conference Proceedings*, 1998.
- G. Barquero, S. Escalera, and C. Palmero. Seamless human motion composition with blended positional encodings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- J. Bednarik, P. Fua, and M. Salzmann. Learning to reconstruct texture-less deformable surfaces from a single view. In *International Conference on 3D Vision (3DV)*, 2018.
- S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.
- P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, 1992.

- B. L. Bhatnagar, S. Singh, C. Arora, and C. Jawahar. Unsupervised learning of deep feature representation for clustering egocentric actions. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017.
- B. L. Bhatnagar, G. Tiwari, C. Theobalt, and G. Pons-Moll. Multi-garment net: Learning to dress 3d people from images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- B. L. Bhatnagar, X. Xie, I. A. Petrov, C. Sminchisescu, C. Theobalt, and G. Pons-Moll. Behave: Dataset and method for tracking human object interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- Z. Bi, Y. Zeng, C. Zeng, F. Pei, X. Feng, K. Zhou, and H. Wu. Gs³: Efficient relighting with triple gaussian splatting. In *ACM SIGGRAPH Asia Conference Papers*, 2024.
- A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorber, Y. Levi, Z. English, V. Voleti, A. Letts, V. Jampani, and R. Rombach. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.
- M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison. Codeslam—learning a compact, optimisable representation for dense visual slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- F. Bogo, M. J. Black, M. Loper, and J. Romero. Detailed full-body reconstructions of moving people from monocular RGB-D sequences. In *IEEE International Conf. on Computer Vision*, 2015.
- F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *European Conference on Computer Vision (ECCV)*, 2016.
- L. Bolanos, S.-Y. Su, and H. Rhodin. Gaussian shadow casting for neural characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- G. Bradski. The OpenCV library. <https://opencv.org>, 2000.
- M. Brand and A. Hertzmann. Style machines. In *ACM SIGGRAPH Conference Proceedings, SIGGRAPH '00*. ACM Press/Addison-Wesley Publishing Co., 2000.
- T. Brooks, B. Peebles, C. Holmes, W. DePue, Y. Guo, L. Jing, D. Schnurr, J. Taylor, T. Luhman, E. Luhman, C. Ng, R. Wang, and A. Ramesh. Video generation models as world simulators. <https://openai.com/research/video-generation-models-as-world-simulators>, 2024.
- T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan,

- R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- M. C. Buehler, Y. Yuan, X. Li, Y. Huang, K. Nagano, and U. Iqbal. Dream, lift, animate: From single images to animatable gaussian avatars. In *International Conference on 3D Vision (3DV)*, 2026.
- K. Byrski, M. Mazur, J. Tabor, T. Dziarmaga, M. Kądziołka, D. Baran, and P. Spurek. Raysplats: Ray tracing based gaussian splatting. *arXiv preprint arXiv:2501.19196*, 2025.
- ByteDance Seed Team. Seedance 1.0: Exploring the boundaries of video generation models. *arXiv preprint arXiv:2506.09113*, 2025a.
- ByteDance Seed Team. Seedance 1.5 pro: A native audio-visual joint generation foundation model. *arXiv preprint arXiv:2512.13507*, 2025b.
- ByteDance Seed Team. Seedance 2.0. https://seed.bytedance.com/en/seedance2_o, 2026.
- C. Cao, Y. Zhang, Y. Wu, H. Lu, and J. Cheng. Egocentric gesture recognition using recurrent 3d convolutional neural networks with spatiotemporal transformer modules. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.
- Y. Cao, Y.-P. Cao, K. Han, Y. Shan, and K.-Y. K. Wong. DreamAvatar: Text-and-shape guided 3D human avatar generation via diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- Z. Cao, H. Gao, K. Mangalam, Q.-Z. Cai, M. Vo, and J. Malik. Long-term human motion prediction with scene context. In *European Conference on Computer Vision (ECCV)*, 2020.
- N. Carion et al. Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.
- E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. D. Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis, T. Karras, and G. Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *International Conference on 3D Vision (3DV)*, 2017.
- D. Chang, Y. Shi, Q. Gao, H. Xu, J. Fu, G. Song, Q. Yan, Y. Zhu, X. Yang, and M. Soleymani. MagicPose: Realistic human poses and facial expressions retargeting

- with identity-aware diffusion. In *International Conference on Machine Learning (ICML)*, 2024.
- H. Chen, Z. Lin, and J. Zhang. GI-GS: Global illumination decomposition on gaussian splatting for inverse rendering. In *International Conference on Learning Representations (ICLR)*, 2025a.
- W. Chen, H. Wang, Y. Li, H. Su, Z. Wang, C. Tu, D. Lischinski, D. Cohen-Or, and B. Chen. Synthesizing training images for boosting human 3d pose estimation. In *International Conference on 3D Vision (3DV)*, 2016.
- W. Chen, P. Li, W. Zheng, C. Zhao, M. Li, Y. Zhu, Z. Dou, R. Wang, and Y. Liu. Synchronizing 2D and 3D generative models for single-view human reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.
- X. Chen, B. Zhou, F. Lu, L. Wang, L. Bi, and P. Tan. Garment modeling with a depth camera. *ACM Transactions on Graphics*, 2015.
- X. Chen, Y. Zheng, M. J. Black, O. Hilliges, and A. Geiger. Snarf: Differentiable forward skinning for animating non-rigid neural implicit shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- Y. Chen, Z. Chen, C. Zhang, F. Wang, X. Yang, Y. Wang, Z. Cai, L. Yang, H. Liu, and G. Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024a.
- Z. Chen and Z. Liu. Relighting4d: Neural relightable human from videos. In *European Conference on Computer Vision (ECCV)*, 2022.
- Z. Chen et al. Urhand: Universal relightable hands. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.
- W. Cheng, R. Chen, W. Yin, S. Fan, K. Chen, H. He, H. Luo, Z. Cai, J. Wang, Y. Gao, Z. Yu, Z. Lin, D. Ren, L. Yang, Z. Liu, C. C. Loy, C. Qian, W. Wu, D. Lin, B. Dai, and K.-Y. Lin. Dna-rendering: A diverse neural actor repository for high-fidelity human-centric rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- J. Chibane, T. Alldieck, and G. Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- S. Clavet. Motion matching and the road to next-gen animation. In *Game Development Conference*, 2016.
- G. Comanici et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

- Y. Cui, W. Chang, T. Nöll, and D. Stricker. Kinectavatar: fully automatic body capture using a single kinect. In *Asian Conf. on Computer Vision*, 2012.
- A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- R. Daněřek, E. Dibra, C. Öztireli, R. Ziegler, and M. Gross. Deepgarment: 3d garment shape estimation from a single image. *Computer Graphics Forum*, 2017.
- L. Dang, Y. Nie, C. Long, Q. Zhang, and G. Li. Msr-gcn: Multi-scale residual graph convolution networks for human motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- V. Delaitre, D. F. Fouhey, I. Laptev, J. Sivic, A. Gupta, and A. A. Efros. Scene semantics from long-term observation of people. In *European Conference on Computer Vision (ECCV)*, 2012.
- B. Deng, J. P. Lewis, T. Jeruzalski, G. Pons-Moll, G. Hinton, M. Norouzi, and A. Tagliasacchi. Nasa neural articulated shape approximation. In *European Conference on Computer Vision (ECCV)*, 2020.
- C. Desai and D. Ramanan. Detecting actions, poses, and objects with relational phraselets. In *European Conference on Computer Vision (ECCV)*, 2012.
- D. DeTone, T. Malisiewicz, and A. Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018.
- J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *NAACL-HLT (1)*. Association for Computational Linguistics, 2019.
- H. Dhamo, Y. Nie, A. Moreau, J. Song, R. Shaw, Y. Zhou, and E. Pérez-Pellitero. Headgas: Real-time animatable head avatars via 3d gaussian splatting. In *European Conference on Computer Vision (ECCV)*, 2024.
- J.-N. Dihlmann, A. Majumdar, A. Engelhardt, R. Braun, and H. P. Lensch. Subsurface scattering for 3D gaussian splatting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- C. Diller and A. Dai. Cg-hoi: Contact-guided 3d human-object interaction generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- H. Dong, X. Liang, K. Gong, H. Lai, J. Zhu, and J. Yin. Soft-gated warping-gan for pose-guided person image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

- H. Dong, X. Liang, X. Shen, B. Wang, H. Lai, J. Zhu, Z. Hu, and J. Yin. Towards multi-pose guided virtual try-on network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019a.
- H. Dong, X. Liang, X. Shen, B. Wu, B.-C. Chen, and J. Yin. Fw-gan: Flow-navigated warping gan for video virtual try-on. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019b.
- Z. Dong, L. Duan, J. Song, M. J. Black, and A. Geiger. MoGA: 3D generative avatar prior for monocular gaussian avatar reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- A. Dravid, Y. Gandelsman, K.-C. Wang, R. Abdal, G. Wetzstein, A. A. Efros, and K. Aberman. Interpreting the weight space of customized diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- K. Du, Z. Liang, Y. Shen, and Z. Wang. GS-ID: Illumination decomposition on gaussian splatting via adaptive light aggregation and diffusion-guided material priors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- Y. Duan, T. Shi, Z. Zou, Y. Lin, Z. Qian, B. Zhang, and Y. Yuan. Single-shot motion completion with transformer. *arXiv preprint arXiv:2103.00776*, 2021.
- P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, D. Podell, T. Dockhorn, Z. English, and R. Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning (ICML)*, 2024.
- M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, 1996.
- J. Fan, F. Luan, J. Yang, M. Hasan, and B. Wang. Rng: Relightable neural gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- A. C. Fang and N. S. Pollard. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics (TOG)*, 2003.
- A. Fathi, A. Farhadi, and J. M. Rehg. Understanding egocentric activities. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2011.
- P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International journal of computer vision*, 2005.
- M. Fischler and R. Bolles. Random Sampling Consensus: A Paradigm for Model Fitting with Application to Image Analysis and Automated Cartography. *Communications of the ACM (CACM)*, 1981.

- D. F. Fouhey, V. Delaitre, A. Gupta, A. A. Efros, I. Laptev, and J. Sivic. People watching: Human actions as a cue for single view geometry. *International journal of computer vision*, 2014.
- K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015.
- V. Gabeur, J.-S. Franco, X. Martin, C. Schmid, and G. Rogez. Moulding humans: Non-parametric 3d human shape estimation from single images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- Q. Gan, Y. Ren, C. Zhang, Z. Ye, P. Xie, X. Yin, Z. Yuan, B. Peng, and J. Zhu. HumanDiT: Pose-guided diffusion transformer for long-form human motion video generation. *arXiv preprint arXiv:2502.04847*, 2025.
- J. Gao, C. Gu, Y. Lin, Z. Li, H. Zhu, X. Cao, L. Zhang, and Y. Yao. Relightable 3d gaussians: Realistic point cloud relighting with brdf decomposition and ray tracing. In *European Conference on Computer Vision (ECCV)*, 2024.
- I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.
- H. Grabner, J. Gall, and L. Van Gool. What makes a chair a chair? In *CVPR 2011*, 2011.
- A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.
- A. Grigorev, A. Sevastopolsky, A. Vakhitov, and V. Lempitsky. Coordinate-based texture inpainting for pose-guided human image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- C. Gu, X. Wei, Z. Zeng, Y. Yao, and L. Zhang. IRGS: Inter-reflective gaussian splatting with 2D gaussian ray tracing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- A. Guédon and V. Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- R. A. Güler, N. Neverova, and I. Kokkinos. DensePose: Dense human pose estimation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- C. Guo, S. Zou, X. Zuo, S. Wang, W. Ji, X. Li, and L. Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

- C. Guo, T. Jiang, X. Chen, J. Song, and O. Hilliges. Vid2avatar: 3d avatar reconstruction from videos in the wild via self-supervised scene decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- C. Guo, J. Li, Y. Kant, Y. Sheikh, S. Saito, and C. Cao. Vid2avatar-pro: Authentic avatar from videos in the wild via universal prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- J. Guo, D. Zhang, X. Liu, Z. Zhong, Y. Zhang, P. Wan, and D. Zhang. LivePortrait: Efficient portrait animation with stitching and retargeting control. *arXiv preprint arXiv:2407.03168*, 2024a.
- K. Guo, P. Lincoln, P. Davidson, J. Busch, X. Yu, M. Whalen, G. Harvey, S. Orts Escolano, R. Pandey, J. Dourgarian, D. Tang, A. Tkach, A. Kowdle, E. Cooper, M. Dou, S. Fanello, G. Fyffe, C. Rhemann, J. Taylor, P. Debevec, and S. Izadi. The relightables: Volumetric performance capture of humans with realistic relighting. *ACM Transactions on Graphics*, 2019.
- Y. Guo, C. Yang, A. Rao, Z. Liang, Y. Wang, Y. Qiao, M. Agrawala, D. Lin, and B. Dai. AnimateDiff: Animate your personalized text-to-image diffusion models without specific tuning. In *International Conference on Learning Representations (ICLR)*, 2024b.
- A. Gupta and L. S. Davis. Objects in action: An approach for combining action understanding and object perception. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.
- A. Gupta, S. Satkin, A. A. Efros, and M. Hebert. From 3d scene geometry to human workspace. In *CVPR 2011*, 2011.
- V. Guzov*, A. Mir*, T. Sattler, and G. Pons-Moll. Human poseitioning system (hps): 3d human pose estimation and self-localization in large scenes from body-mounted sensors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- M. Habermann, W. Xu, M. Zollhöfer, G. Pons-Moll, and C. Theobalt. Livecap: Real-time human performance capture from monocular video. *ACM Transactions on Graphics*, 2019.
- M. Habermann, L. Liu, W. Xu, G. Pons-Moll, M. Zollhoefer, and C. Theobalt. Hdhumans: A hybrid approach for high-fidelity digital humans. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2023.
- I. Habibie, D. Holden, J. Schwarz, J. Yearsley, and T. Komura. A recurrent variational autoencoder for human motion synthesis. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.
- X. Han, Z. Wu, Z. Wu, R. Yu, and L. S. Davis. Viton: An image-based virtual try-on network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

- R. Haralick, C.-N. Lee, K. Ottenberg, and M. Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision (IJCV)*, 1994.
- P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 1968.
- F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal. Robust motion in-betweening. *ACM Transactions on Graphics*, 2020.
- M. Hassan, V. Choutas, D. Tzionas, and M. J. Black. Resolving 3D human pose ambiguities with 3D scene constraints. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- M. Hassan, D. Ceylan, R. Villegas, J. Saito, J. Yang, Y. Zhou, and M. Black. Stochastic Scene-Aware motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021a.
- M. Hassan, P. Ghosh, J. Tesch, D. Tzionas, and M. J. Black. Populating 3D scenes by learning human-scene interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021b.
- J. Hasselgren, N. Hofmann, and J. Munkberg. Shape, light, and material decomposition from images using monte carlo rendering and denoising. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- S. Hauswiesner, M. Straka, and G. Reitmayr. Free viewpoint virtual try-on with commodity depth cameras. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, 2011.
- C. He, J. Saito, J. Zachary, H. Rushmeier, and Y. Zhou. Nemf: Neural motion fields for kinematic animation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- M. He, P. Clausen, A. L. Taşel, L. Ma, O. Pilarski, W. Xian, L. Rikker, X. Yu, R. Burgert, N. Yu, and P. Debevec. Diffrelight: Diffusion-based facial performance relighting. In *ACM SIGGRAPH Asia Conference Papers*, 2024a.
- T. He, Y. Xu, S. Saito, S. Soatto, and T. Tung. Arch++: Animation-ready clothed human reconstruction revisited. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- X. He, X. Li, D. Kang, J. Ye, C. Zhang, L. Chen, X. Gao, H. Zhang, Z. Wu, and H. Zhuang. MagicMan: Generative novel view synthesis of humans with 3D-aware diffusion and iterative refinement. *arXiv preprint arXiv:2408.14211*, 2024b.
- T. Helten, A. Baak, G. Bharaj, M. Muller, H.-P. Seidel, and C. Theobalt. Personalization and evaluation of a real-time depth-based full body tracker. In *International Conference on 3D Vision (3DV)*, 2013.

- G. E. Henter, S. Alexanderson, and J. Beskow. Moglow: Probabilistic and controllable motion synthesis using normalising flows. *ACM Transactions on Graphics (TOG)*, 2020.
- G. Hinton. Using relaxation to find a puppet. In *Proceedings of the 2nd Summer Conference on Artificial Intelligence and Simulation of Behaviour, AISB'76*. IOS Press, 1976.
- J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.
- D. Hogg. Model-based vision: a program to see a walking person. *Image and Vision Computing*, 1983.
- D. Holden, T. Komura, and J. Saito. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)*, 2017.
- D. Holden, O. Kanoun, M. Perepichka, and T. Popa. Learned motion matching. *ACM Transactions on Graphics (TOG)*, 2020.
- Y. Hong, K. Zhang, J. Gu, S. Bi, Y. Zhou, D. Liu, F. Liu, K. Sunkavalli, T. Bui, and H. Tan. LRM: Large reconstruction model for single image to 3D. In *International Conference on Learning Representations (ICLR)*, 2024.
- D. House and D. Breen. *Cloth modeling and animation*. CRC Press, 2000.
- W.-L. Hsiao, I. Katsman, C.-Y. Wu, D. Parikh, and K. Grauman. Fashion++: Minimal edits for outfit improvement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- L. Hu, X. Gao, P. Zhang, K. Sun, B. Zhang, and L. Bo. Animate anyone: Consistent and controllable image-to-video synthesis for character animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024a.
- L. Hu, H. Zhang, Y. Zhang, B. Zhou, B. Liu, S. Zhang, and L. Nie. Gaussianavatar: Towards realistic human avatar modeling from a single video via animatable 3d gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.
- L. Hu, G. Wang, Z. Shen, X. Gao, D. Meng, L. Zhuo, P. Zhang, B. Zhang, and L. Bo. Animate anyone 2: High-fidelity character image animation with environment affordance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- S. Hu, T. Hu, and Z. Liu. GauHuman: Articulated gaussian splatting from monocular human videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024c.
- B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH Conference Proceedings*, 2024a.

- Y. Huang, M. Kaufmann, E. Aksan, M. J. Black, O. Hilliges, and G. Pons-Moll. Deep inertial poser: Learning to reconstruct human pose from sparse inertial measurements in real time. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 2018.
- Y. Huang, H. Yi, Y. Xiu, T. Liao, J. Tang, D. Cai, and J. Thies. TeCH: Text-guided reconstruction of lifelike clothed humans. In *International Conference on 3D Vision (3DV)*, 2024b.
- Y. Huang, Y. Yuan, X. Li, J. Kautz, and U. Iqbal. Adahuman: Animatable detailed 3D human generation with compositional multiview diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- Z. Huang, Y. Xu, C. Lassner, H. Li, and T. Tung. Arch: Animatable reconstruction of clothed humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- I. Hwang, B. Zhou, Y. M. Kim, J. Wang, and C. Guo. Scenemi: Motion in-betweening for modeling human-scene interactions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- M. Işık, M. Rünz, M. Georgopoulos, T. Khakhulin, J. Starck, L. Agapito, and M. Nießner. Humanrf: High-fidelity neural radiance fields for humans in motion. *ACM Transactions on Graphics*, 2023.
- Imagen Team Google. Imagen 3. *arXiv preprint arXiv:2408.07009*, 2024.
- E. Insafutdinov, M. Andriluka, L. Pishchulin, S. Tang, E. Levinkov, B. Andres, and B. Schiele. Arttrack: Articulated multi-person tracking in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- U. Iqbal, M. Garbade, and J. Gall. Pose for action-action for pose. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, 2017.
- U. Iqbal, A. Caliskan, K. Nagano, S. Khamis, P. Molchanov, and J. Kautz. Rana: Relightable articulated neural avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- S. Iwase, S. Saito, T. Simon, S. Lombardi, T. Bagautdinov, R. Joshi, F. Prada, T. Shiratori, Y. Sheikh, and J. Saragih. Relightablehands: Efficient neural relighting of articulated hand models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. A. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. J. Davison, and A. W. Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *ACM Symposium on User Interface Software and Technology (UIST)*, 2011.

- Jack-Jones. Jack and jones website. <http://www.jackjones.com>, 2020.
- M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.
- M.-H. Jeong, D.-H. Han, and H.-S. Ko. Garment capture from a photograph. *Computer Animation and Virtual Worlds*, 2015.
- C. Ji, T. Yu, K. Guo, J. Liu, and Y. Liu. Geometry-aware single-image full-body human relighting. In *European Conference on Computer Vision (ECCV)*, 2022.
- B. Jiang, Y. Hong, H. Bao, and J. Zhang. SelfRecon: Self reconstruction your digital avatar from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022a.
- H. Jiang and K. Grauman. Seeing invisible poses: Estimating 3d body pose from egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- N. Jiang, Z. Zhang, H. Li, X. Ma, Z. Wang, Y. Chen, T. Liu, Y. Zhu, and S. Huang. Scaling up dynamic human-scene interaction modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024a.
- W. Jiang, K. M. Yi, G. Samei, O. Tuzel, and A. Ranjan. Neuman: Neural human radiance field from a single video. In *European Conference on Computer Vision (ECCV)*, 2022b.
- Y. Jiang, S. Jiang, G. Sun, Z. Su, K. Guo, M. Wu, J. Yu, and L. Xu. NeuralHOFusion: Neural volumetric rendering under human-object interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022c.
- Y. Jiang, Z. Shen, Y. Hong, C. Guo, Y. Wu, Y. Zhang, J. Yu, and L. Xu. Robust dual gaussian splatting for immersive human-centric volumetric videos. *ACM Transactions on Graphics*, 2024b.
- Y. Jiang, J. Tu, Y. Liu, X. Gao, X. Long, W. Wang, and Y. Ma. GaussianShader: 3D gaussian splatting with shading functions for reflective surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024c.
- Z. Jiang, Z. Han, C. Mao, J. Zhang, Y. Pan, and Y. Liu. VACE: All-in-one video creation and editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- H. Jin, I. Liu, P. Xu, X. Zhang, S. Han, S. Bi, X. Zhou, Z. Xu, and H. Su. Tensorir: Tensorial inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- N. Jin, Y. Zhu, Z. Geng, and R. Fedkiw. A pixel-based framework for data-driven clothing. *arXiv preprint arXiv:1812.01677*, 2018.

- H. Junkawitsch, G. Sun, H. Zhu, C. Theobalt, and M. Habermann. Eva: Expressive virtual avatars from multi-view videos. In *ACM SIGGRAPH Conference Proceedings*, 2025.
- Y. Kanamori and Y. Endo. Relighting humans: Occlusion-aware inverse rendering for full-body human images. *ACM Transactions on Graphics*, 2018.
- A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of StyleGAN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- K. Karunratanakul, K. Preechakul, E. Aksan, T. Beeler, S. Suwajanakorn, and S. Tang. Optimizing diffusion noise can serve as universal motion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- M. Kaufmann, E. Aksan, J. Song, F. Pece, R. Ziegler, and O. Hilliges. Convolutional autoencoders for human motion infilling. In *International Conference on 3D Vision (3DV)*, 2020.
- N. Keetha, J. Karhade, K. M. Jatavallabhula, G. Yang, S. Scherer, D. Ramanan, and J. Luiten. Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 2023.
- R. Khrodgar, T. Bagautdinov, J. Martinez, S. Zhaoen, A. James, P. Selednik, S. Anderson, and S. Saito. Sapiens: Foundation for human vision models. In *European Conference on Computer Vision (ECCV)*, 2024.
- H. Kim, M. Jang, W. Yoon, J. Lee, D. Na, and S. Woo. Switchlight: Co-design of physics-driven architecture and pre-training framework for human portrait relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.
- D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.
- T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

- H. Kjellström, J. Romero, and D. Kragić. Visual object-action recognition: Inferring object affordances from human demonstration. *Computer Vision and Image Understanding*, 2011.
- L. Kneip, D. Scaramuzza, and R. Siegwart. A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.
- M. Kocabas, J.-H. R. Chang, J. Gabriel, O. Tuzel, and A. Ranjan. HUGS: Human gaussian splats. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- N. Kolotouros, T. Alldieck, A. Zanfir, E. Bazavan, M. Fieraru, and C. Sminchisescu. DreamHuman: Animatable 3D avatars from text. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- W. Kong et al. HunyuanVideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.
- L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM Transactions on Graphics*, 2002a.
- L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *ACM SIGGRAPH Conference Proceedings*, 2002b.
- Z. Kukelova, M. Bujnak, and T. Pajdla. Closed-Form Solutions to Minimal Absolute Pose Problems with Known Vertical Direction. In *Asian Conference on Computer Vision (ACCV)*, 2010.
- N. Kulkarni, A. Gupta, and S. Tulsiani. Canonical surface mapping via geometric cycle consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- Z. Lahner, D. Cremers, and T. Tung. Deepwrinkles: Accurate and realistic clothing modeling. In *European Conf. on Computer Vision*, 2018.
- C. Lassner and M. Zollhöfer. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- C. Lassner, G. Pons-Moll, and P. V. Gehler. A generative model of people in clothing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.
- V. Lazova, E. Insafutdinov, and G. Pons-Moll. 360-degree textures of people in clothing from a single image. In *International Conference on 3D Vision (3DV)*, 2019.
- K. Lebeda, J. E. S. Matas, and O. Chum. Fixing the Locally Optimized RANSAC. In *British Machine Vision Conference (BMVC)*, 2012.

- C. Lee, J. Lee, and T.-K. Kim. Mpmavatar: Learning 3d gaussian avatars with accurate and robust physics-based dynamics. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.
- J. Lee, J. Chai, P. S. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. In *ACM SIGGRAPH Conference Proceedings*, 2002.
- J. Lee, C. Won, H. Jung, I. Bae, and H.-G. Jeon. Fully explicit dynamic gaussian splatting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- J. Lei, Y. Wang, G. Pavlakos, L. Liu, and K. Daniilidis. GART: Gaussian articulated template models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- H. Li, E. Vouga, A. Gudym, L. Luo, J. T. Barron, and G. Gusev. 3d self-portraits. *ACM Transactions on Graphics*, 2013.
- J. Li, Y. Yin, H. Chu, Y. Zhou, T. Wang, S. Fidler, and H. Li. Learning to generate diverse dance motions with transformer. *arXiv preprint arXiv:2008.08171*, 2020a.
- J. Li, C. Cao, G. Schwartz, R. Khirodkar, C. Richardt, T. Simon, Y. Sheikh, and S. Saito. Uravatar: Universal relightable gaussian codec avatars. In *ACM SIGGRAPH Asia Conference Papers*, 2024a.
- M. Li, S. Chen, Y. Zhao, Y. Zhang, Y. Wang, and Q. Tian. Dynamic multiscale graph neural networks for 3d skeleton based human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020b.
- P. Li, W. Zheng, Y. Liu, T. Yu, Y. Li, X. Qi, X. Chi, S. Xia, Y.-P. Cao, W. Xue, W. Luo, and Y. Guo. Pshuman: Photorealistic single-image 3D human reconstruction using cross-scale multiview diffusion and explicit remeshing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- R. Li, S. Yang, D. A. Ross, and A. Kanazawa. Learn to dance with aist++: Music conditioned 3d dance generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- R. Li, J. Tanke, M. Vo, M. Zollhofer, J. Gall, A. Kanazawa, and C. Lassner. Tava: Template-free animatable volumetric actors. In *European Conference on Computer Vision (ECCV)*, 2022.
- T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 2017.
- X. Li, S. Liu, K. Kim, X. Wang, M.-H. Yang, and J. Kautz. Putting humans in a scene: Learning affordance in 3D indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019a.

- Z. Li and N. Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- Z. Li, J. Sedlar, J. Carpentier, I. Laptev, N. Mansard, and J. Sivic. Estimating 3d motion and forces of person-object interactions from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019b.
- Z. Li, Z. Chen, Z. Li, and Y. Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.
- Z. Li, Y. Sun, Z. Zheng, L. Wang, S. Zhang, and Y. Liu. Animatable and relightable gaussians for high-fidelity human avatar modeling. *arXiv preprint arXiv:2311.16096*, 2024c.
- Z. Li, Z. Zheng, L. Wang, and Y. Liu. Animatable gaussians: Learning pose-dependent gaussian maps for high-fidelity human avatar modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024d.
- Z. Liang, Q. Zhang, Y. Feng, Y. Shan, and K. Jia. GS-IR: 3D gaussian splatting for inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- G. Lin, J. Jiang, J. Yang, Z. Zheng, and C. Liang. OmniHuman-1: Rethinking the scaling-up of one-stage conditioned human animation models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- H. Y. Ling, F. Zinno, G. Cheng, and M. van de Panne. Character controllers using motion vaes. *ACM Trans. Graph.*, 2020.
- L. Ling, Y. Sheng, Z. Tu, W. Zhao, C. Xin, K. Wan, L. Yu, Q. Guo, Z. Yu, Y. Lu, X. Li, X. Sun, R. Ashok, A. Mukherjee, H. Kang, X. Kong, G. Hua, T. Zhang, B. Benes, and A. Bera. DL3DV-10K: A large-scale scene dataset for deep learning-based 3D vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- C. K. Liu, A. Hertzmann, and Z. Popović. Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics (TOG)*, 2005.
- J.-W. Liu, Y.-P. Cao, T. Yang, Z. Xu, J. Keppo, Y. Shan, X. Qie, and M. Z. Shou. HOSNeRF: Dynamic human-object-scene neural radiance fields from a single video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- L. Liu, H. Li, and Y. Dai. Efficient global 2d-3d matching for camera localization in a large-scale 3d map. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

- L. Liu, M. Habermann, V. Rudnev, K. Sarkar, J. Gu, and C. Theobalt. Neural actor: Neural free-view synthesis of human actors with pose control. *ACM Transactions on Graphics*, 2021.
- T. Lokovic and E. Veach. Deep shadow maps. In *ACM SIGGRAPH Conference Proceedings*, 2000.
- M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics*, 2015.
- Y. Lu, J. Dong, Y. Kwon, Q. Zhao, B. Dai, and F. De la Torre. GAS: Generative avatar synthesis from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *International Conference on 3D Vision (3DV)*, 2024.
- F. Luo, H. Pan, X. Yang, B. Jiang, F. Liu, and T. Huang. ShadowGS: Shadow-aware 3D gaussian splatting for satellite imagery. *arXiv preprint arXiv:2601.00939*, 2026.
- Y. Luo, Z. Rong, L. Wang, L. Zhang, T. Hu, and Y. Zhu. DreamActor-M1: Holistic, expressive and robust human image animation with hybrid guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart. Get out of my lab: Large-scale, real-time visual-inertial localization. In *Robotics: Science and Systems*, 2015.
- L. Ma, X. Jia, Q. Sun, B. Schiele, T. Tuytelaars, and L. Van Gool. Pose guided person image generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- L. Ma, Q. Sun, S. Georgoulis, L. Van Gool, B. Schiele, and M. Fritz. Disentangled person image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- M. Ma, H. Fan, and K. M. Kitani. Going deeper into first-person activity recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- Q. Ma, J. Yang, A. Ranjan, S. Pujades, G. Pons-Moll, S. Tang, and M. Black. Learning to dress 3d people in generative clothing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- Y. Ma, H. Liu, H. Wang, H. Pan, Y. He, J. Yuan, A. Zeng, C. Cai, H.-Y. Shum, W. Liu, and Q. Chen. Follow-your-emoji: Fine-controllable and expressive freestyle portrait animation. In *ACM SIGGRAPH Asia Conference Proceedings*, 2024.

- N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black. AMASS: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- C. Malleson, M. Volino, A. Gilbert, M. Trumble, J. Collomosse, and A. Hilton. Real-time full-body motion capture from video and imus. In *International Conference on 3D Vision (3DV)*, 2017.
- W. Mao, M. Liu, M. Salzmann, and H. Li. Learning trajectory dependencies for human motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 1978.
- J. Martinez, M. J. Black, and J. Romero. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics*, 2017.
- D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, S. Sridhar, G. Pons-Moll, and C. Theobalt. Single-shot multi-person 3d pose estimation from monocular rgb. In *International Conference on 3D Vision (3DV)*, 2018.
- Y. Men, Y. Yao, M. Cui, and L. Bo. MIMO: Controllable character video synthesis with spatial decomposed modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- M. Mihajlovic, S. Prokudin, S. Tang, R. Maier, F. Bogo, T. Tung, and E. Boyer. SplatFields: Neural gaussian splats for sparse 3d and 4d reconstruction. In *European Conference on Computer Vision (ECCV)*, 2024.
- B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020.
- A. Mir, T. Alldieck, and G. Pons-Moll. Learning to transfer texture from clothing images to 3d humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

- A. Mir, A. Moreau, H. Dharmo, Z. Zhang, G. Pons-Moll, and E. Pérez-Pellitero. Gas-pacho: Gaussian splatting for controllable humans and objects. *arXiv preprint arXiv:2503.09342*, 2025a.
- A. Mir, J. Wang, R. A. Guler, C. Guo, G. Pons-Moll, and B. Zhou. Aha! animating human avatars in diverse scenes with gaussian splatting. *arXiv preprint arXiv:2511.09827*, 2025b.
- N. Moenne-Loccoz, A. Mirzaei, O. Perel, R. de Lutio, J. M. Esturo, G. State, S. Fidler, N. Sharp, and Z. Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes. *ACM Transactions on Graphics*, 2024.
- R. Mokady, A. Hertz, K. Aberman, Y. Pritch, and D. Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- G. Moon, T. Shiratori, and S. Saito. Expressive whole-body 3D gaussian avatar. In *European Conference on Computer Vision (ECCV)*, 2024.
- A. Moreau, J. Song, H. Dharmo, R. Shaw, Y. Zhou, and E. Pérez-Pellitero. Human gaussian splatting: Real-time rendering of animatable avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- A. Moreau, M. Brahimi, R. Shaw, A. Papaioannou, T. Tanay, Z. Zhang, and E. Pérez-Pellitero. Better together: Unified motion capture and 3d avatar reconstruction. *arXiv preprint arXiv:2503.09293*, 2025.
- T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 2022.
- R. Natsume, S. Saito, Z. Huang, W. Chen, C. Ma, H. Li, and S. Morishima. Sic-lope: Silhouette-based clothed people. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- NavVis. Navvis m6. <https://www.navvis.com/m6>, 2020.
- M. Nazarczuk, S. Catley-Chandar, T. Tanay, Z. Zhang, G. Slabaugh, and E. Pérez-Pellitero. Vidar: Video diffusion-aware 4d reconstruction from monocular inputs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.
- A. Neophytou and A. Hilton. A layered model of human body and garment deformation. In *International Conference on 3D Vision (3DV)*, 2014.
- N. Neverova, R. A. Güler, and I. Kokkinos. Dense pose transfer. In *European Conference on Computer Vision (ECCV)*, 2018.
- N. Neverova, D. Novotny, M. Szafraniec, V. Khalidov, P. Labatut, and A. Vedaldi. Continuous surface embeddings. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

- R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality*, 2011.
- H. N. Ng and R. L. Grimsdale. Computer graphics techniques for modeling cloth. *IEEE Computer Graphics and Applications*, 1996.
- M. Omran, C. Lassner, G. Pons-Moll, P. Gehler, and B. Schiele. Neural body fitting: Unifying deep learning and model based human pose and shape estimation. In *International Conference on 3D Vision (3DV)*, 2018.
- OpenAI. ChatGPT (GPT-5). <https://chatgpt.com/>, 2025. Large language model, August 2025 version.
- OpenAI. Sora 2. <https://openai.com/index/sora-2/>, 2025.
- B. N. Oreshkin, A. Valkanas, F. G. Harvey, L.-S. Ménard, F. Bocquetlet, and M. J. Coates. Motion inbetweening via deep δ -interpolator. *arXiv preprint arXiv:2201.06701*, 2022.
- P. Pan, Z. Su, C. Lin, Z. Fan, Y. Zhang, Z. Li, T. Shen, Y. Mu, and Y. Liu. HumanSplat: Generalizable single-image human gaussian splatting with structure priors. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- R. Pandey, S. Orts-Escolano, C. LeGendre, C. Haene, S. Bouaziz, C. Rhemann, P. Debevec, and S. Fanello. Total relighting: Learning to relight portraits for background replacement. *ACM Transactions on Graphics*, 2021.
- H. Pang, H. Zhu, A. Kortylewski, C. Theobalt, and M. Habermann. Ash: Animatable gaussian splats for efficient and photoreal human rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- H. E. Pang, S. Liu, Z. Cai, L. Yang, T. Zhang, and Z. Liu. DISCO4D: Disentangled 4D human generation and animation from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- K. Park, K. Rematas, A. Farhadi, and S. M. Seitz. Photoshape: Photorealistic materials for large-scale shape collections. *ACM Trans. Graph.*, 2018.
- A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

- C. Patel, L. Zhouyingcheng, and G. Pons-Moll. Tailornet: Predicting clothing in 3d as a function of human pose, shape and garment style. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- M. Paulich, M. Schepers, N. Rudigkeit, and G. Bellusci. *Xsens MTw Awinda: Miniature Wireless Inertial-Magnetic Motion Tracker for Highly Accurate 3D Kinematic Applications*, 2018.
- G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. A. Osman, D. Tzionas, and M. J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- H.-Y. Peng, J.-P. Zhang, M.-H. Guo, Y.-P. Cao, and S.-M. Hu. CharacterGen: Efficient 3D character generation from single images with multi-view pose canonicalization. *ACM Transactions on Graphics*, 2024.
- S. Peng, J. Dong, Q. Wang, S. Zhang, Q. Shuai, X. Zhou, and H. Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- M. Petrovich, M. J. Black, and G. Varol. Action-conditioned 3D human motion synthesis with transformer VAE. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- M. Petrovich, M. J. Black, and G. Varol. TEMOS: Generating diverse human motions from textual descriptions. In *European Conference on Computer Vision (ECCV)*, 2022.
- E. Pinyoanuntapong, M. U. Saleem, K. Karunratanakul, P. Wang, H. Xue, C. Chen, C. Guo, J. Cao, J. Ren, and S. Tulyakov. Maskcontrol: Spatio-temporal control for masked motion synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele. Poselet conditioned pictorial structures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.
- L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. Gehler, and B. Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

- G. Pons-Moll, A. Baak, T. Helten, M. Müller, H.-P. Seidel, and B. Rosenhahn. Multisensor-fusion for 3d full-body human motion capture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.
- G. Pons-Moll, J. Taylor, J. Shotton, A. Hertzmann, and A. Fitzgibbon. Metric regression forests for correspondence estimation. *International Journal of Computer Vision (IJCV)*, 2015.
- G. Pons-Moll, S. Pujades, S. Hu, and M. Black. ClothCap: Seamless 4D clothing capture and retargeting. *ACM Transactions on Graphics*, 2017.
- B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *International Conference on Learning Representations (ICLR)*, 2023.
- T. Popa, Q. Zhou, D. Bradley, V. Kraevoy, H. Fu, A. Sheffer, and W. Heidrich. Wrinkling captured garments using space-time data-driven deformation. *Computer Graphics Forum*, 2009.
- X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- A. Pumarola, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer. Unsupervised person image synthesis in arbitrary poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- A. R. Punnakal, A. Chandrasekaran, N. Athanasiou, A. Quiros-Ramirez, and M. J. Black. BABEL: Bodies, action and behavior with english labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- Z. Qian, S. Wang, M. Mihajlovic, A. Geiger, and S. Tang. 3dgs-avatar: Animatable avatars via deformable 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- L. Qiu, X. Gu, P. Li, Q. Zuo, W. Shen, J. Zhang, K. Qiu, W. Yuan, G. Chen, Z. Dong, and L. Bo. LHM: Large animatable human reconstruction model for single image to 3D in seconds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025a.
- L. Qiu, P. Li, Q. Zuo, X. Gu, Y. Dong, W. Yuan, S. Zhu, X. Han, G. Chen, and Z. Dong. PF-LHM: 3D animatable avatar reconstruction from pose-free articulated human images. *arXiv preprint arXiv:2506.13766*, 2025b.
- L. Qiu, S. Zhu, Q. Zuo, X. Gu, Y. Dong, J. Zhang, C. Xu, Z. Li, W. Yuan, L. Bo, G. Chen, and Z. Dong. Anigs: Animatable gaussian avatar from a single image with inconsistent gaussian reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025c.
- A. Raj, P. Sangkloy, H. Chang, J. Hays, D. Ceylan, and J. Lu. Swapnet: Image based garment transfer. In *European Conference on Computer Vision (ECCV)*, 2018.

- W. T. Reeves, D. H. Salesin, and R. L. Cook. Rendering antialiased shadows with depth maps. In *ACM SIGGRAPH Conference Proceedings*, 1987.
- P. S. Reitsma and N. S. Pollard. Evaluating motion graphs for character animation. *ACM Transactions on Graphics (TOG)*, 2007.
- RenderPeople. Renderpeople. <https://renderpeople.com/>, 2025. Accessed: 2025-09-17.
- H. Rhodin, C. Richardt, D. Casas, E. Insafutdinov, M. Shafiei, H.-P. Seidel, B. Schiele, and C. Theobalt. Egocap: egocentric marker-less motion capture with two fisheye cameras. *ACM Transactions on Graphics (TOG)*, 2016.
- D. Roetenberg, H. Luinge, and P. Slycke. Moven: Full 6dof human motion tracking using miniature inertial sensors. *Xsen Technologies, December*, 2007.
- G. Rogez, J. S. Supancic, and D. Ramanan. First-person pose recognition using egocentric workspaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. In *European Conference on Computer Vision (ECCV)*, 2002.
- C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 2004.
- L. Rout, Y. Chen, N. Ruiz, C. Caramanis, S. Shakkottai, and W.-S. Chu. Semantic image inversion and editing using rectified stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2025.
- A. Safonova and J. K. Hodgins. Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 2007.
- A. Safonova, J. K. Hodgins, and N. S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics (ToG)*, 2004.
- C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, R. Gontijo-Lopes, B. K. Ayan, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li. PIFu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

- S. Saito, T. Simon, J. Saragih, and H. Joo. PIFuHD: Multi-level pixel-aligned implicit function for high-resolution 3D human digitization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- S. Saito, G. Schwartz, T. Simon, J. Li, and G. Nam. Relightable gaussian codec avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- I. Sárándi and G. Pons-Moll. Neural localizer fields for continuous 3D human pose and shape estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- K. Sarkar, M. C. Buehler, G. Li, D. Wang, D. Vicini, J. Riviere, Y. Zhang, S. Orts-Escolano, P. Gotardo, T. Beeler, and A. Meka. LitNeRF: Intrinsic radiance decomposition for high-quality view synthesis and relighting of faces. In *ACM SIGGRAPH Asia Conference Papers*, 2023.
- P. Sarlin, F. Debraine, M. Dymczyk, R. Siegwart, and C. Cadena. Leveraging deep visual descriptors for hierarchical efficient localization. In *Conference on Robot Learning*, 2018.
- P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperGlue: Learning Feature Matching with Graph Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- T. Sattler, B. Leibe, and L. Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE transactions on pattern analysis and machine intelligence*, 2016.
- T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixe. Understanding the limitations of cnn-based absolute camera pose regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- M. Savva, A. X. Chang, P. Hanrahan, M. Fisher, and M. Nießner. PiGraphs: Learning Interaction Snapshots from Observations. *ACM Transactions on Graphics (TOG)*, 2016.
- J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.
- J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

- R. Shao, Y. Pang, Z. Zheng, J. Sun, and Y. Liu. Human4DiT: 360-degree human video generation with 4D diffusion transformer. *ACM Transactions on Graphics*, 2024.
- A. Shapiro, A. Feng, R. Wang, H. Li, M. Bolas, G. Medioni, and E. Suma. Rapid avatar capture and simulation using commodity depth sensors. *Computer Animation and Virtual Worlds*, 2014.
- R. Shaw, M. Nazarczuk, J. Song, A. Moreau, S. Catley-Chandar, H. Dhamo, and E. Perez-Pellitero. SWinGS: Sliding windows for dynamic 3D gaussian splatting. In *European Conference on Computer Vision (ECCV)*, 2024.
- J. Shin, J. Lee, S. Lee, M.-G. Park, J.-M. Kang, J. H. Yoon, and H.-G. Jeon. Canonical-fusion: Generating drivable 3d human avatars from multiple images. In *European Conference on Computer Vision (ECCV)*, 2024.
- T. Shiratori, H. S. Park, L. Sigal, Y. Sheikh, and J. K. Hodgins. Motion capture from body-mounted cameras. *ACM Transactions on Graphics*, 2011.
- J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Efficient human pose estimation from single depth images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.
- A. Shysheya, E. Zakharov, K.-A. Aliev, R. Bashirov, E. Burkov, K. Isakov, A. Ivakhnenko, Y. Malkov, I. Pasechnik, D. Ulyanov, A. Vakhitov, and V. Lempitsky. Textured neural avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- A. Siarohin, E. Sangineto, S. Lathuilière, and N. Sebe. Deformable gans for pose-based human image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- A. Siarohin, S. Lathuilière, S. Tulyakov, E. Ricci, and N. Sebe. First order motion model for image animation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- G. Sim and G. Moon. PERSONA: Personalized whole-body 3D avatar with pose-driven deformations from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021a.
- S. Song, W. Zhang, J. Liu, and T. Mei. Unsupervised person image generation with semantic parsing transformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021b.

- O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *ACM SIGGRAPH Conference Proceedings*, 2004.
- S. Starke, H. Zhang, T. Komura, and J. Saito. Neural state machine for character-scene interactions. *ACM Trans. Graph.*, 2019.
- S. Starke, Y. Zhao, T. Komura, and K. Zaman. Local motion phases for learning multi-contact character movements. *ACM Trans. Graph.*, 2020.
- S. Starke, Y. Zhao, F. Zinno, and T. Komura. Neural animation layering for synthesizing martial arts movements. *ACM Trans. Graph.*, 2021.
- J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.
- G. Sun, X. Chen, Y. Chen, A. Pang, P. Lin, Y. Jiang, L. Xu, J. Wang, and J. Yu. Neural free-viewpoint performance rendering under complex human-object interactions. In *Proceedings of the ACM International Conference on Multimedia*, 2021.
- T. Sun, J. T. Barron, Y.-T. Tsai, Z. Xu, X. Yu, G. Fyffe, C. Rhemann, J. Busch, P. E. Debevec, and R. Ramamoorthi. Single image portrait relighting. *ACM Transactions on Graphics*, 2019.
- SuperSplat. Supersplat. <https://supersplat.at/>, 2025.
- L. Svärm, O. Enqvist, F. Kahl, and M. Oskarsson. City-scale localization for cameras with known vertical direction. *IEEE transactions on pattern analysis and machine intelligence*, 2016.
- O. Taheri, N. Ghorbani, M. J. Black, and D. Tzionas. GRAB: A dataset of whole-body human grasping of objects. In *European Conference on Computer Vision (ECCV)*, 2020.
- O. Taheri, V. Choutas, M. J. Black, and D. Tzionas. Goal: Generating 4d whole-body motion for hand-object grasping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii. Inloc: Indoor visual localization with dense matching and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- J. Tan, D. Xiang, S. Tulsiani, D. Ramanan, and G. Yang. DressRecon: Freeform 4D human reconstruction from monocular video. In *International Conference on 3D Vision (3DV)*, 2025.

- M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, J. Kerr, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, and A. Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH Conference Proceedings*, 2023.
- Y. Tao, Z. Zheng, Y. Zhong, J. Zhao, D. Quionhai, G. Pons-Moll, and Y. Liu. Simulcap : Single-view human performance capture with cloth simulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- F. Taubner, R. Zhang, M. Tuli, and D. B. Lindell. CAP4D: Creating animatable 4D portrait avatars with morphable multi-view diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- J. Taylor, J. Shotton, T. Sharp, and A. W. Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- G. Tevet, S. Raab, B. Gordon, Y. Shafir, D. Cohen-Or, and A. H. Bermano. Human motion diffusion model. In *International Conference on Learning Representations (ICLR)*, 2023.
- C. Toft, E. Stenborg, L. Hammarstrand, L. Brynte, M. Pollefeys, T. Sattler, and F. Kahl. Semantic match consistency for long-term visual localization. In *European Conference on Computer Vision (ECCV)*, 2018.
- Tom-Tailor. Tom-tailor website. <http://www.tom-tailor.eu>, 2020.
- D. Tome, T. Alldieck, P. Peluse, G. Pons-Moll, L. Agapito, H. Badino, and F. de la Torre. Selfpose: 3d egocentric pose estimation from a headset mounted camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
- A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla. 24/7 place recognition by view synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.
- M. Trumble, A. Gilbert, C. Malleson, A. Hilton, and J. Collomosse. Total capture: 3d human pose estimation fusing video and inertial sensors. In *Proceedings of 28th British Machine Vision Conference*, 2017.
- S. Tu, Z. Xing, X. Han, Z.-Q. Cheng, Q. Dai, C. Luo, and Z. Wu. StableAnimator: High-quality identity-preserving human image animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid. Learning from synthetic humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- G. Varol, D. Ceylan, B. Russell, J. Yang, E. Yumer, I. Laptev, and C. Schmid. Bodynet: Volumetric inference of 3d human body shapes. In *European Conf. on Computer Vision*, 2018.

- D. Vlastic, R. Adelsberger, G. Vannucci, J. Barnwell, M. Gross, W. Matusik, and J. Popović. Practical motion capture in everyday surroundings. *ACM Transactions on Graphics (TOG)*, 2007.
- T. von Marcard, G. Pons-Moll, and B. Rosenhahn. Human pose estimation from video and IMUs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2016.
- T. von Marcard, B. Rosenhahn, M. Black, and G. Pons-Moll. Sparse inertial poser: Automatic 3d human pose estimation from sparse imus. *Computer Graphics Forum 36(2), Proceedings of the 38th Annual Conference of the European Association for Computer Graphics (Eurographics)*, 2017.
- T. von Marcard, R. Henschel, M. Black, B. Rosenhahn, and G. Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *European Conf. on Computer Vision*, 2018.
- F. Walch, C. Hazirbas, L. Leal-Taixé, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-Based Localization Using LSTMs for Structured Feature Correlation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.
- Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.
- B. Wang, H. Zheng, X. Liang, Y. Chen, and L. Lin. Toward characteristic-preserving image-based virtual try-on network. In *European Conference on Computer Vision (ECCV)*, 2018.
- J. Wang, H. Xu, J. Xu, S. Liu, and X. Wang. Synthesizing long-term 3d human motion and interaction in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021a.
- J. Wang, S. Yan, B. Dai, and D. Lin. Scene-aware generative network for human motion synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021b.
- J. Wang, Y. Rong, J. Liu, S. Yan, D. Lin, and B. Dai. Towards diverse and natural scene-aware 3d human motion synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025a.
- J. Wang, J. Pu, Z. Qi, J. Guo, Y. Ma, N. Huang, Y. Chen, X. Li, and Y. Shan. Taming rectified flow for inversion and editing. In *International Conference on Machine Learning (ICML)*, 2025b.
- J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence*, 2007.

- L. Wang, X. Zhao, J. Sun, Y. Zhang, H. Zhang, T. Yu, and Y. Liu. StyleAvatar: Real-time photo-realistic portrait avatar from a single video. In *ACM SIGGRAPH Conference Proceedings*, 2023.
- S. Wang, T. Simon, I. Santesteban, T. Bagautdinov, J. Li, V. Agrawal, F. Prada, S.-I. Yu, P. Nalbone, M. Gramlich, R. Lubachersky, C. Wu, J. Romero, J. Saragih, M. Zollhoefer, A. Geiger, S. Tang, and S. Saito. Relightable full-body gaussian codec avatars. In *ACM SIGGRAPH Conference Proceedings*, 2025c.
- T. Y. Wang, H. Su, Q. Huang, J. Huang, L. Guibas, and N. J. Mitra. Unsupervised texture transfer from images to model collections. *ACM Trans. Graph.*, 2016.
- X. Wang, R. Girdhar, and A. Gupta. Binge watching: Scaling affordance learning from sitcoms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- X. Wang, S. Zhang, C. Gao, J. Wang, X. Zhou, Y. Zhang, L. Yan, and N. Sang. UniAnimate: Taming unified video diffusion models for consistent human image animation. *Science China Information Sciences*, 2025d.
- X. Wang, S. Zhang, L. Tang, Y. Zhang, C. Gao, Y. Wang, and N. Sang. UniAnimate-DiT: Human image animation with large-scale video diffusion transformer. *arXiv preprint arXiv:2504.11289*, 2025e.
- X. Wang, Y. Zhao, B. Ye, X. Shan, W. Lyu, L. Qi, K. C. K. Chan, Y. Li, and M.-H. Yang. Holigs: Holistic gaussian splatting for embodied view synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025f.
- Y. Wang, F. Hong, S. Yang, L. Jiang, W. Wu, and C. C. Loy. MEAT: Multiview diffusion model for human generation on megapixels with mesh attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025g.
- Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004.
- H. Wei, Z. Yang, and Z. Wang. AniPortrait: Audio-driven synthesis of photorealistic portrait animation. *arXiv preprint arXiv:2403.17694*, 2024.
- L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li. Dense human body correspondences using convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- J. Weil. The synthesis of cloth objects. In *ACM SIGGRAPH Conference Proceedings*, 1986.
- J. Wen, X. Zhao, Z. Ren, A. G. Schwing, and S. Wang. GoMAvatar: Efficient animatable human modeling from monocular video using gaussians-on-mesh. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

- C.-Y. Weng, B. Curless, and I. Kemelmacher-Shlizerman. Photo wake-up: 3d character animation from a single photo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- C.-Y. Weng, B. Curless, P. P. Srinivasan, J. T. Barron, and I. Kemelmacher-Shlizerman. HumanNeRF: Free-viewpoint rendering of moving people from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- L. A. Westover. *Splatting: a parallel, feed-forward volume rendering algorithm*. PhD thesis, University of North Carolina at Chapel Hill, 1992.
- T. Weyand, I. Kostrikov, and J. Philbin. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision (ECCV)*, 2016.
- L. Williams. Casting curved shadows on curved surfaces. In *ACM SIGGRAPH Conference Proceedings*, 1978.
- B. Wu et al. HunyuanVideo 1.5 technical report. *arXiv preprint arXiv:2511.18870*, 2025a.
- G. Wu, T. Yi, J. Fang, L. Xie, X. Zhang, W. Wei, W. Liu, Q. Tian, and X. Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- Q. Wu, J. Martinez Esturo, A. Mirzaei, N. Moenne-Loccoz, and Z. Gojcic. 3dgut: Enabling distorted cameras and secondary rays in gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025b.
- Y. Wu, J. Wang, Y. Zhang, S. Zhang, O. Hilliges, F. Yu, and S. Tang. Saga: Stochastic whole-body grasping with contact. In *European Conference on Computer Vision (ECCV)*, 2022.
- Z. Wu, J. Chen, L. Li, S. Wu, Z. Zhu, K. Xu, M. R. Oswald, and J. Song. 3D gaussian inverse rendering with approximated global illumination. *arXiv preprint arXiv:2504.01358*, 2025c.
- X. Xie, B. L. Bhatnagar, and G. Pons-Moll. Chore: Contact, human and object reconstruction from a single rgb image. In *European Conference on Computer Vision (ECCV)*, 2022a.
- X. Xie, B. L. Bhatnagar, and G. Pons-Moll. Visibility aware human-object interaction tracking from single rgb camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- X. Xie, B. L. Bhatnagar, J. E. Lenssen, and G. Pons-Moll. Template free reconstruction of human-object interaction with procedural interaction generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024a.
- Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022b.

- Y. Xie, V. Jampani, L. Zhong, D. Sun, and H. Jiang. Omnicontrol: Control any joint at any time for human motion generation. In *International Conference on Learning Representations (ICLR)*, 2024b.
- Y. Xie, H. Xu, G. Song, C. Wang, Y. Shi, and L. Luo. X-Portrait: Expressive portrait animation with hierarchical motion attention. In *ACM SIGGRAPH Conference Proceedings*, 2024c.
- Y. Xiu, J. Yang, X. Cao, D. Tzionas, and M. J. Black. ECON: Explicit clothed humans optimized via normal integration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- H. Xu, T. Alldieck, and C. Sminchisescu. H-nerf: Neural radiance fields for rendering and temporal reconstruction of humans in motion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- S. Xu, G. Chen, Y.-X. Guo, J. Yang, C. Li, Z. Zang, Y. Zhang, X. Tong, and B. Guo. VASA-1: Lifelike audio-driven talking faces generated in real time. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.
- W. Xu, A. Chatterjee, M. Zollhoefer, H. Rhodin, D. Mehta, H.-P. Seidel, and C. Theobalt. Monoperfcap: Human performance capture from monocular video. *ACM Transactions on Graphics*, 2018.
- W. Xu, A. Chatterjee, M. Zollhoefer, H. Rhodin, P. Fua, H.-P. Seidel, and C. Theobalt. Mo²Cap² : Real-time mobile 3d motion capture with a cap-mounted fisheye camera. *IEEE Transactions on Visualization and Computer Graphics*, 2019.
- Y. Xu, B. Chen, Z. Li, H. Zhang, L. Wang, Z. Zheng, and Y. Liu. Gaussian head avatar: Ultra high-fidelity head avatar via dynamic gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.
- Z. Xu, Z. Huang, J. Cao, Y. Zhang, X. Cun, Q. Shuai, Y. Wang, L. Bao, J. Li, and F. Tang. AnchorCrafter: Animate cyber-anchors selling your products via human-object interacting video generation. *arXiv preprint arXiv:2411.17383*, 2024c.
- Z. Xu, J. Zhang, J. H. Liew, H. Yan, J.-W. Liu, C. Zhang, J. Feng, and M. Z. Shou. MagicAnimate: Temporally consistent human image animation using diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024d.
- Z. Xu, Z. Yu, Z. Zhou, J. Zhou, X. Jin, F.-T. Hong, X. Ji, J. Zhu, C. Cai, S. Tang, Q. Lin, X. Li, and Q. Lu. HunyuanPortrait: Implicit condition control for enhanced portrait animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- L. Xue, C. Guo, C. Zheng, F. Wang, T. Jiang, H.-I. Ho, M. Kaufmann, J. Song, and O. Hilliges. HSR: holistic 3d human-scene reconstruction from monocular videos. In *European Conference on Computer Vision (ECCV)*, 2024.

- J. Yang, J.-S. Franco, F. Hétroy-Wheeler, and S. Wuhrer. Analyzing clothing layer deformation statistics of 3d human motions. In *European Conf. on Computer Vision*, 2018a.
- S. Yang, Z. Pan, T. Amert, K. Wang, L. Yu, T. Berg, and M. C. Lin. Physics-inspired garment recovery from a single-view image. *ACM Transactions on Graphics (TOG)*, 2018b.
- X. Yang, X. Chen, D. Gao, S. Wang, X. Han, and B. Wang. HAVE-FUN: Human avatar reconstruction from few-shot unconstrained images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- Z. Yang, J. Teng, W. Zheng, M. Ding, S. Huang, J. Xu, Y. Yang, W. Hong, X. Zhang, G. Feng, D. Yin, Y. Zhang, W. Wang, Y. Cheng, B. Xu, X. Gu, Y. Dong, and J. Tang. CogVideoX: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations (ICLR)*, 2025.
- B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.
- C. Ye, Y. Nie, J. Chang, Y. Chen, Y. Zhi, and X. Han. Gaustudio: A modular framework for 3d gaussian splatting and beyond. *arXiv preprint arXiv:2403.19632*, 2024.
- C. Yeshwanth, Y.-C. Liu, M. Nießner, and A. Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- W. Yifan, F. Serena, S. Wu, C. Öztireli, and O. Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. In *ACM SIGGRAPH Asia Conference Proceedings*, 2019.
- H. Yonemoto, K. Murasaki, T. Osawa, K. Sudo, J. Shimamura, and Y. Taniguchi. Egocentric articulated pose tracking for action recognition. In *International Conference on Machine Vision Applications (MVA)*, 2015.
- R. Yu, X. Wang, and X. Xie. Vtnfp: An image-based virtual try-on network with body and clothing feature preservation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- Y. Yuan and K. Kitani. 3d ego-pose estimation via imitation learning. In *European Conference on Computer Vision (ECCV)*, 2018.
- Y. Yuan and K. Kitani. Ego-pose estimation and forecasting as real-time pd control. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- Y. Yuan and K. Kitani. Dlow: Diversifying latent flows for diverse human motion prediction. In *European Conference on Computer Vision (ECCV)*, 2020.

- Y. Yuan, U. Iqbal, P. Molchanov, K. Kitani, and J. Kautz. Glamr: Global occlusion-aware human mesh recovery with dynamic cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- Y. Yuan, J. Song, U. Iqbal, A. Vahdat, and J. Kautz. PhysDiff: Physics-guided human motion diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- Zalando. Zalando website. <http://www.zalando.com>, 2020.
- A. Zanfir, E. Marinoiu, and C. Sminchisescu. Monocular 3d pose and shape estimation of multiple people in natural scenes—the importance of multiple scene constraints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018a.
- M. Zanfir, A.-I. Popa, A. Zanfir, and C. Sminchisescu. Human appearance transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b.
- M. Zeng, J. Zheng, X. Cheng, and X. Liu. Templateless quasi-rigid shape modeling with implicit loop-closure. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.
- Y. Zhan, T. Shao, H. Wang, Y. Yang, and K. Zhou. Interactive rendering of relightable and animatable gaussian avatars. *IEEE Transactions on Visualization and Computer Graphics*, 2025a.
- Y. Zhan, Q. Zhu, M. Niu, M. Ma, J. Zhao, Z. Zhong, X. Sun, Y. Qiao, and Y. Zheng. Tomie: Towards explicit exoskeleton for the reconstruction of complicated 3d human avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025b.
- C. Zhang, S. Pujades, M. J. Black, and G. Pons-Moll. Detailed, accurate, human shape estimation from clothed 3d scan sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- H. Zhang, S. Starke, T. Komura, and J. Saito. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)*, 2018a.
- J. Zhang, H. Luo, H. Yang, X. Xu, Q. Wu, Y. Shi, J. Yu, L. Xu, and J. Wang. Neurdome: A neural modeling pipeline on multi-view human-object interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- J. Y. Zhang, S. Pepose, H. Joo, D. Ramanan, J. Malik, and A. Kanazawa. Perceiving 3d human-object spatial arrangements from a single image in the wild. In *European Conference on Computer Vision (ECCV)*, 2020a.
- L. Zhang, Q. Zhang, M. Wu, J. Yu, and L. Xu. Neural video portrait relighting in real-time via consistency modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021a.

- R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b.
- S. Zhang, Y. Zhang, Q. Ma, M. J. Black, and S. Tang. PLACE: Proximity learning of articulation and contact in 3D environments. In *International Conference on 3D Vision (3DV)*, 2020b.
- X. Zhang, B. L. Bhatnagar, S. Starke, V. Guzov, and G. Pons-Moll. Couch: Towards controllable human-chair interactions. In *European Conference on Computer Vision (ECCV)*, 2022.
- Y. Zhang and S. Tang. The wanderings of odysseus in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- Y. Zhang, M. Hassan, H. Neumann, M. J. Black, and S. Tang. Generating 3d people in scenes without people. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020c.
- Y. Zhang, M. J. Black, and S. Tang. We are more than our joints: Predicting how 3D bodies move. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021b.
- Y. Zhang, J. Gu, L.-W. Wang, H. Wang, J. Cheng, Y. Zhu, and F. Zou. MimicMotion: High-quality human motion video generation with confidence-aware pose guidance. In *International Conference on Machine Learning (ICML)*, 2025.
- Z. Zhang, Z. Yang, and Y. Yang. SIFU: Side-view conditioned implicit function for real-world usable clothed human reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- B. Zhao, X. Wu, Z.-Q. Cheng, H. Liu, Z. Jie, and J. Feng. Multi-view image generation from a single-view. In *2018 ACM Multimedia Conference on Multimedia Conference*, 2018.
- J. Zhao and H. Zhang. Thin-plate spline motion model for image animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- K. Zhao, S. Wang, Y. Zhang, T. Beeler, , and S. Tang. Compositional human-scene interaction synthesis with semantic control. In *European Conference on Computer Vision (ECCV)*, 2022.
- K. Zhao, Y. Zhang, S. Wang, T. Beeler, and S. Tang. DIMOS: Synthesizing diverse human motions in 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- K. Zhao, G. Li, and S. Tang. DartControl: A diffusion-based autoregressive motion model for real-time text-driven motion control. In *International Conference on Learning Representations (ICLR)*, 2025a.

- Y. Zhao, C. Wu, B. Huang, Y. Zhi, C. Zhao, J. Wang, and S. Gao. Surfel-based gaussian inverse rendering for fast and relightable dynamic human reconstruction from monocular video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025b.
- I. Zheng, G. Tang, A. Doronin, P. D. Teal, and F.-L. Zhang. SSD-GS: Scattering and shadow decomposition for relightable 3D gaussian splatting. In *International Conference on Learning Representations (ICLR)*, 2026.
- N. Zheng, X. Song, Z. Chen, L. Hu, D. Cao, and L. Nie. Virtually trying on new clothing with arbitrary poses. In *Proceedings of the 27th ACM International Conference on Multimedia*, 2019.
- S. Zheng, B. Zhou, R. Shao, B. Liu, S. Zhang, L. Nie, and Y. Liu. Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- Z. Zheng, T. Yu, H. Li, K. Guo, Q. Dai, L. Fang, and Y. Liu. Hybridfusion: Real-time performance capture using a single depth sensor and sparse imus. In *European Conference on Computer Vision (ECCV)*, 2018.
- Z. Zheng, T. Yu, Y. Liu, and Q. Dai. PaMIR: Parametric model-conditioned implicit representation for image-based human reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- Z. Zheng, X. Zhao, H. Zhang, B. Liu, and Y. Liu. AvatarReX: Real-time expressive full-body avatars. *ACM Transactions on Graphics*, 2023.
- S. Zhi, M. Bloesch, S. Leutenegger, and A. J. Davison. Scenecode: Monocular dense semantic reconstruction using learned encoded scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- Y. Zhong, Z. Li, D. Z. Chen, L. Hong, and D. Xu. Taming video diffusion prior with scene-grounding guidance for 3d gaussian splatting from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- J. Zhou, Y. Wu, S. Li, M. Wei, C. Fan, W. Chen, W. Jiang, and F. Wang. RealisDance-DiT: Simple yet strong baseline towards controllable character animation in the wild. *arXiv preprint arXiv:2504.14977*, 2025.
- H. Zhu, F. Zhan, C. Theobalt, and M. Habermann. Trihuman: A real-time and controllable tri-plane representation for detailed human geometry and appearance synthesis. *ACM Transactions on Graphics*, 2024a.
- S. Zhu, R. Urtasun, S. Fidler, D. Lin, and C. Change Loy. Be your own prada: Fashion synthesis with structural coherence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

- S. Zhu, J. L. Chen, Z. Dai, Q. Su, Y. Xu, X. Cao, Y. Yao, H. Zhu, and S. Zhu. Champ: Controllable and consistent human image animation with 3D parametric guidance. In *European Conference on Computer Vision (ECCV)*, 2024b.
- Y. Zhuang, J. Lv, H. Wen, Q. Shuai, A. Zeng, H. Zhu, S. Chen, Y. Yang, X. Cao, and W. Liu. IDOL: Instant photorealistic 3D human creation from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- W. Zielonka, T. Bagautdinov, S. Saito, M. Zollhöfer, J. Thies, and J. Romero. Drivable 3D gaussian avatars. In *International Conference on 3D Vision (3DV)*, 2025.
- L. Ziwei, Y. Sijie, L. Ping, W. Xiaogang, and T. Xiaoou. Fashion landmark detection in the wild. In *European Conference on Computer Vision (ECCV)*, 2016.
- M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In *ACM SIGGRAPH Conference Proceedings*, 2001.
- M. Zwicker, H. Pfister, J. van Baar, and M. Gross. EWA splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3), 2002.