Dense and Globally Consistent Multi-View Stereo

Dense and Globally Consistent Multi-View Stereo

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

M.Sc. Jian Wei

aus Heilongjiang, China

Tübingen

2016

Tag der mündlichen Qualifikation:	25.11.2016
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. Hendrik P. A. Lensch
2. Berichterstatter:	Prof. Dr. Andreas Schilling

Abstract

Multi-View Stereo (MVS) aims at reconstructing dense geometry of scenes from a set of overlapping images which are captured at different viewing angles. This thesis is devoted to addressing MVS problem by estimating depth maps, since 2D-space operations are trivially parallelizable in contrast to 3D volumetric techniques.

Typical setup of depth-map-based MVS approaches consists of per-view calculation and multi-view merging. Most solutions primarily aim at the most precise and complete surfaces for individual views but relaxing the global geometry consistency. Therefore, the inconsistent estimates lead to heavy processing workload in the merging stage and diminish the final reconstruction.

Another issue is the textureless areas where the photo-consistency constraint can not discriminate different depths. These matching ambiguities are normally handled by incorporating plane features or the smoothness assumption, that might produce segmentation effect or depends on accuracy and completeness of the calculated object edges.

This thesis deals with two kinds of input data, photo collections and high-frame-rate videos, by developing distinct MVS algorithms based on their characteristics:

For the sparsely sampled photos, we propose an advanced PatchMatch system that alternates between patch-based correlation maximization and pixel-based optimization of the cross-view consistency. Thereby we get a good trade-off between the photometric and geometric constraints. Moreover, our method achieves high efficiency by combining local pixel traversal and a hierarchical framework for fast depth propagation.

For the densely sampled videos, we mainly focus on recovering the homogeneous surfaces, because the redundant scene information enables ray-level correlation which can generate shape depth discontinuities. Our approach infers smooth surfaces for the enclosed areas using perspective depth interpolation, and subsequently tackles the occlusion errors connecting the fore- and background edges. In addition, our edge depth estimation is more robust by accounting for unstructured camera trajectories.

Exhaustively calculating depth maps is unfeasible when modeling large scenes from videos. This thesis further improves the reconstruction scalability using an incremental scheme via content-aware view selection and clustering. Our goal is to gradually eliminate the visibility conflicts and increase the surface coverage by processing a minimum subset of views. Constructing view clusters allows us to store merged and locally consistent points with the highest resolution, thus reducing the memory requirements.

All approaches presented in the thesis do not rely on high-level techniques, so they can be easily parallelized. The evaluations on various datasets and the comparisons with existing algorithms demonstrate the superiority of our methods.

Kurzfassung

Multi-View Stereo (MVS) rekonstruiert dichte Geometrien von Szenen aus einer Menge von sich überlappenden Bildern, die aus verschiedenen Blickwinkeln aufgenommen worden. Diese Arbeit widmet sich dem MVS Problem durch die Schätzung von Tiefenkarten, da Operationen im zweidimensionalen Raum im Gegensatz zum dreidimensionalen Raum einfach parallelisierbar sind.

Der typische Aufbau von MVS Ansätzen, die auf Tiefenkarten basieren, setzt sich aus einer Berechnung einer Tiefenkarte pro Blickwinkel und dem fusioniren der Tiefenkarten zusammen. Die meisten Lösungsansätze fokussieren sich hauptsächlich auf die präzise und vollständige Berechnung von Oberflächen für jeden Blickwinkel, vernachlässigen jedoch die globale Konsistenz der Geometrie. Jedoch führen diese inkonsistenten Schätzungen zu einem erhöhtem Aufwand beim Fusionieren der Tiefenkarten und verschlechtern die finale Rekonstruktion.

Eine weitere Schwierigkeit besteht darin, Mehrdeutigkeiten in der Photokonsistenz in homogenen Flächen aufzulösen. Typischerweise wird dies durch Ebenen-Features oder die Stetigkeits-Annahme gehandhabt. Dies wiederum kann Segmentierungseffekte produzieren und hängt von der Genauigkeit und Vollständigkeit von den berechneten Objektkanten ab.

Die vorliegende Arbeit behandelt zwei Arten von Eingabedaten, sammlungen von Fotos und Videos mit hoher Bildrate, durch die Entwicklung zweier unterschiedlicher MVS-Algorithmen:

Für einzelne aufgenommene Fotos schlagen wir eine erweiterte Version des PatchMatch Systems vor, welches zwischen der Maximierung von patch-basierter Korrelation und der pixel-basierten Optimierung der Konsistenz zwischen verschiedenen Blickwinkeln alterniert. Dabei erhalten wir einen guten Kompromiss zwischen fotometrischen und geometrischen Bedingungen. Zusätzlich erreicht unsere Methode eine hohe Effektivität durch die Kombination vom Traversieren lokaler Pixel und einem hierarchischen Framework für schnelle Tiefen-Propagation.

Für dicht gesampelte Videos fokussieren wir uns hauptsächlich auf die Rekonstruktion homogener Flächen. Die vorhandene redundante Szeneninformation erlaubt es, Pixel einzeln zu korrelieren, was wiederum Kontur-Diskontinuitäten genauer rekonstruiert. Unser Ansatz schätzt glatte Flächen in der Umgebung durch die Nutzung von perspektivischer Interpolation und löst Verdeckungsprobleme, bei denen Vorder- und Hintergrund fälschlicherweise verbunden werden. Zusätzlich erlaubt dies robustere Tiefenschätzung an Kanten durch Verwendung mehrerer unstrukturierter Kamera-Trajektorien.

Eine vollständige Berechnung aller Tiefenkarten ist unpraktikabel, wenn große Sze-

nen aus Videos betrachtet werden. Diese Arbeit verbessert die Skalierbarkeit von Rekonstruktionen durch ein inkrementelles Schema zur inhaltsbasierenden Blickwinkelauswahl und dem Gruppieren von Bildern, die die selben Oberflächen zeigen. Unser Ziel ist die schrittweise Eliminierung inkonsistenter Oberflächen und das Vervollständigen der Rekonstruktion bei Verwendung einer minimalen Auswahl an Blickwinkeln. Die Rekonstruktion in Gruppen erlaubt es uns, lokal konsistente und fusionierte Punkte in hoher Auflösung zu verwenden und dabei den Speicherbedarf trotzdem niedrig zu halten.

Alle in dieser Arbeit vorgestellte Ansätze basieren auf einfach parallelisierbaren Operationen. Die Auswertung auf verschiedenen Datensätzen und der Vergleich mit bereits existierenden Algorithmen zeigen die Überlegenheit der hier vorgeschlagenen Methoden.

Acknowledgments

I would like to first thank my supervisor Prof. Hendrik P. A. Lensch for giving me the position to pursue my PhD studies in the Computer Graphics group. I am grateful for the freedom to find my research direction and the support he provided throughout my studies. His insightful advices and valuable guidance had a significant influence in my work. I would also like to cordially thank Prof. Andreas Schilling for his efforts in reading and assessing the thesis.

My special thanks go to my colleague Benjamin Resch for our fruitful collaborations. Due to the close connection between our research interests, we had many helpful discussions about various academic problems on my thesis topic. This work would not have been possible without his patience and suggestions.

I am very fortunate to have had the opportunity to work with other colleagues, which include Katharina Schwarz, Fabian Groh, Patrick Wieschollek, Jieen Chen, and Dennis Bukenberger. I am also deeply grateful to the former colleagues who have now left our group including Takahiro Okabe, Christian Fuchs, Manuel Finckh, Beatriz Trinchão Andrade, and Georg Mühlbeier. Thanks to all of them for their friendship and for providing a nice work atmosphere.

Sincere thanks go to the secretaries Violaine Le Guily, Sibylle Hasse, Birgit Grieg, Sophie Freitag, and Manuela Di Paolo for the help they gave me. I thank all my friends I have known in Tübingen who have made it a memorable time for me here.

I would like to express my gratitude to the financial support of China Scholarship Council (CSC). The studies and life in Germany will be a wonderful asset for me.

I am forever indebted to my parents for their love, constant support, and unfailing faith in me. Finally and most wholeheartedly, I thank my wife Liwei Chen. Thank you for your love, optimism, as well as immense encouragement. This work is dedicated to you and our newborn son.

Research Contributions

The researches presented in Chapters 4, 5, and 6 of the thesis were assisted by Dipl.-Inf. Benjamin Resch and supervised by Prof. Dr.-Ing. Hendrik P. A. Lensch. Specific contributions of the researchers are summarized as follows:

- In **Chapter** 4, we introduce an advanced PatchMatch system for estimating depth maps from sparsely sampled photos. The fast depth propagation approach combining more local pixel traversal and a hierarchical framework was designed by the present author, which makes our depth map estimation more efficient on GPUs. The major novelty of this chapter lies in that we alternate between patch-based correlation maximization and pixel-based consistency optimization. The outline of integrating intra-view propagation and cross-view filtering was first proposed by Prof. Lensch. The present author developed this idea by proposing a novel formulation for cross-view depth filtering which respects the estimate consistency, and finding a way of getting a good trade-off between the photometric and geometric constraints that is impossible in the typical setup of depth-map-based multi-view stereo (MVS). The present author was responsible for implementing the algorithm and producing all the results. Benjamin Resch and Prof. Lensch assisted in analyzing the collected results.

- The next two chapters depict two MVS systems for densely sampled, unstructured videos. **Chapters** 5 is concerned with creating a sequence of depth maps for the traditional depth-map-based workflow, particularly targeted at recovering homogeneous areas. In view of the availability of accurate edge depth by the ray-level calculation, we infer smooth surfaces for the areas enclosed by these sparse estimates. The present author proposed this key idea and introduced a new method for visualizing and analyzing the appearance of a point hypothesis in the secondary images, as well as a subpixel-level depth sweeping algorithm. Benjamin Resch assisted in designing the careful score aggregation and two-scale secondary-image selection. Our edge depth calculation is more robust by accounting for arbitrary camera movements. The present author also designed a simpler depth interpolation scheme that is free of perspective distortion. By using the geometry reconstructed from other views, we also tackle the occlusion problem successfully. The visibility conflict invalidation was done with help of Benjamin Resch. For evaluating the performance, Benjamin Resch generated the synthetic datasets. All the implementations and results were produced by the present author.

- **Chapter** 6 extends the previous chapter's work to improve reconstruction scalability that is particularly suitable for large datasets. We do this by actively selecting the most beneficial views for incremental occlusion handling and coverage improvement. For further efficiency gains, we merge each cluster of locally consistent points into a simplified and optimized point set with the highest resolution. The content-adaptive view selection and clustering were proposed by the present author. The idea of exploiting the implicit function was from Benjamin Resch, which was further improved by the present author by respecting the depth uncertainty and only maintaining the highest-resolution points. Benjamin Resch generated the large-scale synthetic scenes and helped in capturing the real-world scenes using the flying drone. All the implementations, results, and analysis were done by the present author.

In all these chapters, some of the camera poses and feature points which are input of the proposed algorithms were obtained from the Structure from Motion work of Benjamin Resch. The author of this thesis wrote the first draft of each chapter. Benjamin Resch and Prof. Lensch provided some valuable suggestions for improving the texts.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
  - 1.1.1 Pipeline of Image-Based Reconstruction . . . 1
  - 1.1.2 Problems for General Inputs . . . 3
  - 1.1.3 Problems for Specific Inputs . . . 3
- 1.2 Contributions and Structure . . . 5

2 Background **7**
- 2.1 Shape Representations . . . 7
- 2.2 Geometry of Perspective Projection . . . 8
- 2.3 Photometric Constraint . . . 9
  - 2.3.1 Photo Consistency . . . 9
  - 2.3.2 Patch Matching . . . 10
  - 2.3.3 Ray-Based Correlation . . . 12
- 2.4 Geometric Constraint . . . 12
  - 2.4.1 Global Consistency . . . 13
  - 2.4.2 Visibility Conflicts . . . 13
  - 2.4.3 Multi-View Consistency Rating . . . 14

3 Related Research **15**
- 3.1 Dense Reconstruction . . . 15
- 3.2 Multi-View Consistency . . . 17
- 3.3 Depth-Map/Point-Cloud Merging . . . 18
- 3.4 Scalable Reconstruction . . . 19
- 3.5 Summary . . . 21

4 Globally Consistent MVS for Sparsely Sampled Photos **23**
- 4.1 Introduction . . . 23
- 4.2 System Overview . . . 24
  - 4.2.1 Initialization . . . 24
  - 4.2.2 Post-Processing . . . 26
- 4.3 Local Propagation . . . 26
  - 4.3.1 Propagation Between Pixels . . . 26
  - 4.3.2 Local Traversal Scheme . . . 26

4.4 Hierarchical Framework . . . 27
4.5 Cross-View Depth Filtering . . . 28
4.5.1 Depth Candidates . . . 28
4.5.2 Depth Variance . . . 29
4.5.3 Cross-View Filtering . . . 30
4.6 Results . . . 31
4.6.1 Parameter Settings . . . 33
4.6.2 Evaluation . . . 33
4.7 Conclusion . . . 40

5 Dense and Occlusion-Robust MVS for Unstructured Videos 43
5.1 Introduction . . . 43
5.2 System Overview . . . 44
5.2.1 Pre-Processing . . . 44
5.3 Edge Depth Estimation . . . 45
5.3.1 Pixel-Wise Color and Score Maps . . . 46
5.3.2 Robust Score Aggregation . . . 46
5.3.3 Two-Scale Secondary-Image Selection . . . 49
5.3.4 Subpixel-Level Depth Sweeping . . . 51
5.3.5 Depth Uncertainty . . . 51
5.4 Homogeneous-Area Depth Estimation . . . 52
5.4.1 Depth Diffusion . . . 53
5.4.2 Visibility Conflict Invalidation . . . 55
5.4.3 Cross-View Propagation . . . 57
5.5 Results . . . 58
5.5.1 Parameter Settings . . . 59
5.5.2 Evaluation . . . 59
5.6 Conclusion . . . 61

6 Dense and Scalable MVS from Unstructured Videos with Occlusions 63
6.1 Introduction . . . 63
6.2 System Overview . . . 64
6.3 Point Representation . . . 66
6.4 Visibility Conflict Invalidation . . . 66
6.5 Content-Aware View Selection and Clustering . . . 68
6.5.1 NBV Determination . . . 69
6.5.2 Active View Clustering . . . 70
6.6 Ray-Wise Point Merging . . . 71
6.6.1 Implicit Function . . . 71
6.6.2 Point Optimization and Simplification . . . 72
6.6.3 Associated Information Updating . . . 73

6.7 Results 74
6.7.1 Parameter Settings 75
6.7.2 Evaluation 75
6.8 Conclusion 77

7 Conclusion **81**
7.1 Contributions 81
7.2 Future Work 82

Symbols **85**

Abbreviations **87**

Bibliography **89**

Chapter 1

Introduction

This thesis focuses on one of the classic but still challenging problems in the area of computer vision, *multi-view stereo*. Typically, this problem aims at reconstructing the dense geometry of a scene from images alone, by capturing the scene at different viewpoints.

The practical applications of 3D reconstruction range from object recognition (Mustafa *et al.*, 2013; Liu *et al.*, 2014) and tracking (Li and Flierl, 2012; Xiang *et al.*, 2014), augmented reality (Yang *et al.*, 2013), to realistic scene visualization (image-based rendering) (Zhou *et al.*, 2013; Oxholm and Nishino, 2014). It is also applicable in other fields covering industry (Uslu *et al.*, 2011; Jadidi *et al.*, 2015), medicine (Ma *et al.*, 2013; Chen *et al.*, 2015), archeology (Ducke *et al.*, 2011; Russell *et al.*, 2011; Verhoeven *et al.*, 2012), etc.

Our objective is to yield convincing scene surfaces with high computational efficiency by estimating and merging a set of dense and globally consistent depth maps. This chapter first briefly introduces the motivation of this thesis in Section 1.1. Then Section 1.2 describes our contributions and the thesis structure.

1.1 Motivation

In this section, we summarize the pipeline of image-based 3D reconstruction and outline problems of the approaches using multi-view depth maps.

1.1.1 Pipeline of Image-Based Reconstruction

The pipeline of multi-view reconstruction commonly consists of two stages:

1. **Structure from Motion (SfM)** The most measurable features are first detected in each image and matched between different views. Then, the established correspondences are used to compute a camera-pose track together with 3D positions of the feature points.

2. **Multi-View Stereo (MVS)** The outputs of SfM, i.e., camera poses and feature points with inferred visibility constraints, are exploited to derive dense scene geometry for better scene visualization and understanding.

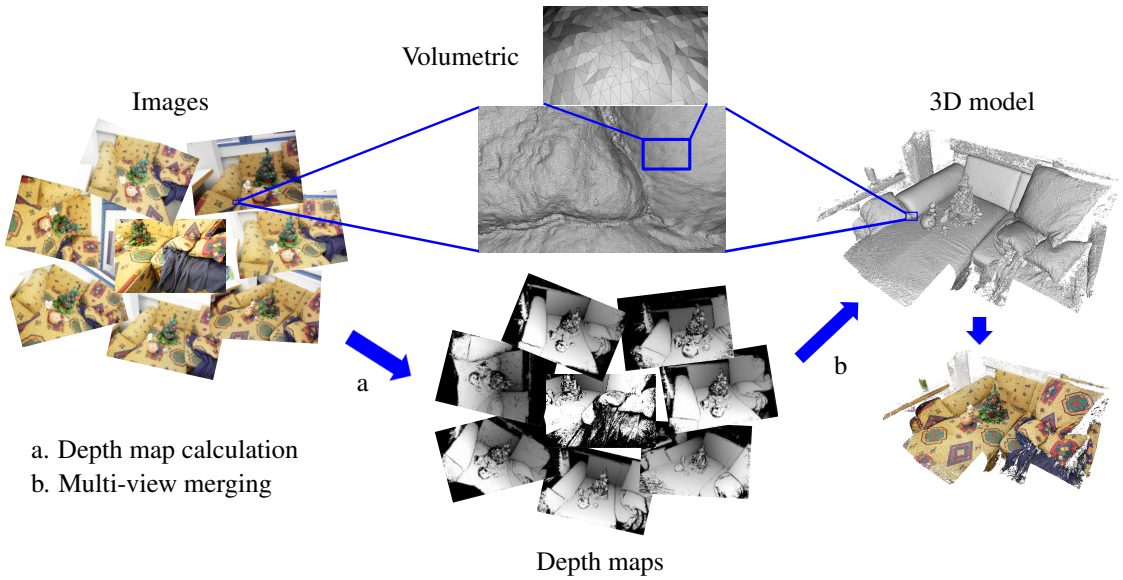


Figure 1.1: In general, MVS techniques can be classified into volumetric (top) and depth-map-based (bottom) methods. The former operates on 3D meshes and is thus more suitable for small scenes as well as single objects. The latter supports higher computational parallelism by calculating and merging view-dependent depth maps.

SfM is basically easier because it merely reconstructs very sparse feature points. There have been many approaches proposed to solve the more difficult MVS problem (see Chapter 3). Among them, the algorithms using depth maps yield comparable surface quality in contrast to the 3D-space volumetric solutions (see Figure 1.1), but they can achieve higher computational efficiency due to the trivially parallelizable image-space operations. Therefore, they are applicable to larger-scale datasets.

The depth-map-based MVS techniques represent 3D geometry using a group of view-dependent depth maps, and normally recover the scene surfaces through two stages (as shown in Figure 1.1):

1. **Depth map calculation** A set of dense depth maps are first created from individual views (see Section 3.1). Given an image (named reference image), its depth estimates are calculated by correlating each single pixel or small patch in a selected subset of other images (named secondary images) based on the *photo-consistency* constraint. The definition and two different formulations of this photometric cue are given in Section 2.3.

2. **Multi-view merging** Next, the redundant depth values from multiple views are merged to produce a refined global 3D model (see Section 3.3). In this process, the *visibility conflicts* are eliminated, and the remaining *globally consistent* surface approximations are fused together. We explain the geometric constraint in Section 2.4.

1.1.2 Problems for General Inputs

To attain impressive 3D geometry with higher coverage and less noisy surfaces, the key is to improve the quality of depth maps which are input to the merging step.

Most existing methods primarily aim at the most precise and complete estimates per view, since they generate depth maps independently by relaxing the consistency constraint. The cross-correlation process might struggle with matching ambiguities induced by lack of texture and mismatches due to image noise or occlusion. These problematic areas are usually reconstructed by enforcing local smoothness (see Section 3.1). Although surface completeness is improved, the depth estimates inevitably suffer from many globally inconsistent values. Hence, these approaches put heavy processing workload on the subsequent merging stage and the final recovery quality is diminished as well.

Therefore, a main goal of this thesis is to create dense depth maps with both high accuracy and cross-view consistency. This way, the task for the multi-view merging algorithms can become easier, and less erroneous surface estimates are removed from or fused into the reconstructed 3D model.

1.1.3 Problems for Specific Inputs

The input data for scene modeling can be acquired by taking either a collection of photographs or a high-frame-rate video stream. The former records significantly view-dependent content at sparsely sampled viewpoints by multiple shots from a standard camera. The latter acquires the scene information with denser coverage by arbitrarily moving a video camera around. These two kinds of inputs have both advantages and drawbacks for MVS as well as SfM. No known current reconstruction pipeline is able to generally achieve remarkable modeling for both cases, including ours. Fortunately, besides the typical SfM for photo collections (Snavely *et al.*, 2006), the scalable SfM for video data has been recently designed (Resch *et al.*, 2015). Likewise, this thesis addresses the MVS problem by accordingly developing respective solutions.

Sparsely Sampled Photo Collections

The large baselines of sparsely captured photos provide sufficient triangulation angles. Hence, to calculate a depth, the correspondence is searched by projecting the pixel neighborhoods (patch) to a few other images (see Section 2.3.2). Such *patch-wise* matching strategy enhances reliability of depth determination.

One of the most widely adopted algorithms, PatchMatch (reviewed in Section 3.1), accelerates the correspondence search by iterating between random-offset testing and best-estimate propagation. Although GPU parallelism enables relatively fast process of region growing, the regular pixel traversal along long scanlines does not make full use of the graphics processors. Therefore, this thesis proposes two modifications to further speed up the estimation (see Sections 4.3 and 4.4).

More importantly, in the areas with image noise or no prominent structure, the photo consistency alone might fail to assign the best depth candidate a distinguishable minimum of matching costs (defined in Section 2.3.2). Thus, the global consistency of depth maps should be balanced against the non-robust cross-correlation measure in these areas. Our solution, presented in Section 4.5.3, is based on an optical flow technique (Lang *et al.*, 2012) that improves temporal consistency by computing per-image estimates and simultaneously filtering the results along the calculated motion paths.

The sparsity of image samples leads to greater chances of a scene point for being occluded or projected outside image boundaries. Moreover, matching at patch level tends to blur the depth discontinuities at object boundaries. These disadvantages can be avoided by capturing a high-frame-rate video sequence, as discussed below.

Densely Sampled, Unstructured Videos

As camera hardware improved continually, the frame rate and image resolution of a video camera have been increased, enabling the acquisition to get simpler and faster. So video streams have received more attention than photographs. However, the small image baselines and massively growing amount of input data bring difficulties and challenges to most of classic approaches which were originally designed for photos.

The reconstruction algorithm for light fields (Kim *et al.*, 2013) exploits the data redundancy by projecting each individual pixel to many other images for depth calculation (see Section 2.3.3). Such *ray-based* scheme derives precise and sharp depth discontinuities between objects, allowing the unmeasurable homogeneous areas to be densely recovered in a fine-to-coarse manner (reviewed in Section 3.1). But as the authors claimed, they still produced some noisy estimates in comparably large textureless regions.

Therefore, another special focus of this thesis is on the recovery of large-area homogeneous surfaces. To this end, we simply diffuse the scattered edge-depth values per view, assuming flat surface in between (see Section 5.4.1). This way of depth interpolation has to deal with two potential problems, one beforehand and the other afterwards:

First, the quality of surface interpolants relies entirely on that of edge depth. Since we treat unstructured videos, the arbitrary camera trajectories might induce frequent occurrence of occlusion or out-of-image projection, which would decrease the reliability of cross-view correlation. Thus, this thesis advances the edge depth calculation of (Kim *et al.*, 2013) for more robustness against different possible cases (see Section 5.3).

Second, the view-independent interpolation probably generates wrong surfaces between fore- and background edges in front of the actual scene geometry. This is caused by occlusion of edges in the view being processed. To correct these inconsistent interpolants, we use the estimates from other views where the edges are visible and measurable (see Sections 5.4.2 and 5.4.3).

Furthermore, exhaustively estimating depth maps requires excessive time and memory consumptions, which becomes unfeasible when processing very long videos for large scenes. Only processing a view subset must guarantee that sufficient scene structures are

reconstructed for recovering comparable area of surfaces and removing all visibility conflicts. In this thesis, we achieve scalable reconstruction by content-adaptively selecting a minimum subset of views and constructing view clusters to obtain a visibility-consistent, concise point cloud (see Chapter 6).

1.2 Contributions and Structure

This thesis contributes new techniques to solve the MVS problem using either sparsely acquired photo collections or high-frame-rate unstructured videos as the input. The particular approaches employed for each kind of input are distinct due to different characteristics of the processed data. However, the goals are the same, namely calculating dense depth maps with both high accuracy and high consistency, with the target that as many per-view estimates as possible can be fused into the final 3D model. We also extend our pipeline for more practical reconstruction of large-scale scenes using a video camera.

This thesis incorporates material from three papers by the author (Wei *et al.*, 2014, 2016, 2017). The remainder of the thesis is structured as follows where the contributions of each chapter are outlined.

Chapter 2 provides a background which is needed to understand some chapters of this thesis. We describe the ways of representing 3D shapes and the principle of perspective projection. Next we explain how a depth map can be obtained using the photometric constraint. Two basic correlation algorithms are depicted that are used or improved in the thesis. To enforce global consistency of per-view estimates, we describe how consistent results and visibility conflicts can be distinguished.

Chapter 3 gives an overview of the techniques that are most related to the addressed problems, including the methods achieving dense recovery and multi-view consistency, as well as the approaches for surface merging and scalable reconstruction.

Chapter 4 introduces an advanced PatchMatch system for estimating depth maps from sparsely sampled photos. Our scheme is more efficient on GPUs by combining more local pixel traversal and a hierarchical framework. The major novelty lies in that we alternate between patch-based correlation maximization and pixel-based consistency optimization. Thereby we can get a good trade-off between the photometric and geometric constraints, that is impossible in the typical setup of depth-map-based MVS. This chapter is based on (Wei *et al.*, 2014).

The next two chapters depict two MVS systems for densely sampled, unstructured videos. **Chapters** 5 is concerned with creating a sequence of depth maps for the traditional depth-map-based workflow, particularly targeted at recovering homogeneous areas. In view of the availability of accurate edge depth by the ray-level calculation as in (Kim *et al.*, 2013), the key idea is to infer smooth surfaces for the areas enclosed by these sparse estimates. Our edge depth calculation is more robust by accounting for arbitrary camera movements. Our depth interpolation is simpler and free of perspective

distortion. By using the geometry reconstructed from other views, we also tackle the occlusion problem successfully. This chapter is based on (Wei *et al.*, 2016).

Chapter 6 extends the work of the previous chapter to improve reconstruction scalability that is particularly suitable for large datasets. We do this by actively selecting the most beneficial views for incremental occlusion handling and coverage improvement. For further efficiency gains, we merge the locally consistent points of each view cluster into a simplified and optimized point set with the highest resolution. This chapter is based on (Wei *et al.*, 2017).

Finally, **Chapter** 7 concludes the contributions. We also prospect the future work in this area.

All approaches proposed in the thesis do not rely on high-level techniques, e.g., segmentation, global optimization, ray tracing, or triangulation. Our operations are most often evaluated per patch, per pixel, or per point. Hence, we can achieve high computational efficiency through parallelization.

Chapter 2

Background

This chapter describes the background that is necessary to understand following chapters. We begin by introducing several ways of parameterizing 3D shapes in Section 2.1. Since solving the problem of image-based reconstruction is heavily based on the projections between 2D and 3D spaces, Section 2.2 first gives a brief description of some notions used in the perspective camera model and the projective geometry. For mathematical equations of camera calibration and perspective projection, please see (Hartley and Zisserman, 2004). Then, the photo consistency and two fundamental techniques for dense correspondences are depicted in Section 2.3. Finally, we define the geometric constraints in Section 2.4, which is crucial to globally consistent reconstruction.

2.1 Shape Representations

There have been various ways to parameterize the shapes of 3D scenes. Here, we describe the most commonly used depth maps, point clouds, and meshes, that are also utilized in our approaches and evaluations.

- **Depth maps** The depth measures how far the scene surfaces are from a given camera. Its definition is given in Section 2.2. In the depth map of this viewpoint, the depth information of the scene point appearing in each image pixel is stored. Since one camera only observes the front-most surfaces and may only see the scene partially, multiple overlapping depth maps are generally needed to represent the entire 3D shape. Depth maps can be easily optimized using image-oriented operations.

- **Point clouds** A point cloud represents the scene surfaces using a set of 3D points. Each point is parameterized by its position in the world space, and usually a normal vector as well as a scale value (see Chapter 6) to separately maintain the orientation and area of the corresponding surface patch. Points can be generated by 3D scanning or from depth maps. Although point clouds are not directly applicable in a number of practical cases, they can be converted to meshes or other representation forms of shapes. However, if a point cloud is sufficiently dense and accurate, it is enough for visualizing the reconstructed 3D model.

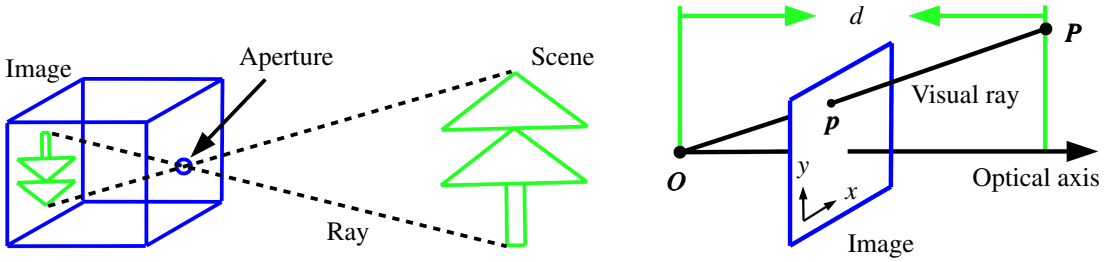


Figure 2.1: The pinhole camera model (left) and the geometry of perspective projection (right). For each pixel $\boldsymbol{p}$ on the 2D image plane, its depth d is calculated by inverting the imaging process of the 3D point $\boldsymbol{P}$ on the scene surface. $\boldsymbol{O}$ is the optical center.

- **Meshes** The meshes are composed of a group of vertices, edges, and faces. The mesh calculation is normally referred to as surface reconstruction, and meshes are widely used particularly in computer graphics. Two popular 3D elements for creating the meshes are triangles and tetrahedral.

Other representations of shapes include height maps (Vogiatzis *et al.*, 2005) or relief surfaces (Vogiatzis *et al.*, 2008), level sets (Osher and Sethian, 1988), occupancy grids (Khan *et al.*, 2007; Brandao *et al.*, 2016), planar disks (Habbecke and Kobbelt, 2007) or surfels (Weise *et al.*, 2011; Kolev *et al.*, 2014), etc.

2.2 Geometry of Perspective Projection

Recovering the 3D shape of a scene from 2D images is essentially the task of inverting the imaging process. So this section first looks at the camera models, among which the linear pinhole camera is the simplest one. As illustrated in Figure 2.1 (left), a pinhole camera is modeled by a box with an infinitesimally small aperture on one face. Light reflected from a scene travels through the aperture, and the light rays intersect with the inside of the opposite box face producing the image of the scene. The intersected box face is called *image plane*, and the aperture is approximated by lenses in an actual camera. Such a mapping from 3D scene space (or world space) to 2D image space is named *perspective projection*.

The geometry of perspective projection, projective geometry, is obtained by symmetrically placing the image plane in front of the aperture to reverse the vertically-flipped image. As presented in Figure 2.1 (right), the 3D location of this aperture is called the *optical center*, denoted as $\boldsymbol{O}$, and the distance between the image plane and $\boldsymbol{O}$ is the *focal length*. The line through $\boldsymbol{O}$ and perpendicular to the image plane is the *optical axis*.

For a scene point $\boldsymbol{P}$, let $\boldsymbol{p}$ be its 2D projection, i.e., the image pixel where $\boldsymbol{P}$ is projected to. The *depth* of $\boldsymbol{p}$ is the distance between $\boldsymbol{O}$ and $\boldsymbol{P}$ measured along the optical axis. Note that the depth's definition is view-dependent.

The 3D line passing through $\boldsymbol{O}$ and $\boldsymbol{P}$ is called the *visual ray*, all points along which project to $\boldsymbol{p}$. Since locating the actual 3D point along the visual ray of an image pixel is a one-to-many problem, necessary constraints have to be incorporated into the reconstruction. For this purpose, the following sections describe two types of cues, one photometric for correlation across images (Section 2.3), and the other geometric for global coherence of per-view estimates (Section 2.4).

2.3 Photometric Constraint

MVS algorithms mostly focus on building dense correspondences over views to recover the underlying depth information. The process of cross-correlation is constrained by the photo consistency which is defined in Section 2.3.1. Because this thesis deals with both multiple photographs and video data, we also review the respective basic methods that produce depth maps by operating on patches (Section 2.3.2) and visual rays (Section 2.3.3). See Figure 2.2 for their main differences.

2.3.1 Photo Consistency

The correspondence recovery assumes that the surfaces of the scene are diffuse (or Lambertian), namely that each surface point appears to have a color unrelated to the viewing direction in the absence of noise. Based on this relatively simple geometry, for each point estimate, its 2D projections in all visible images must exhibit the same color if accurate camera calibration has been attained.

Specifically, we denote a set of images as $\boldsymbol{I}$. For a pixel $\boldsymbol{p}$ in one image of $\boldsymbol{I}$, assume that a depth hypothesis d generates a point candidate $\boldsymbol{P}$. Suppose that $\boldsymbol{P}$ is always visible in $\boldsymbol{I}$, and is projected to the pixel $\boldsymbol{q}_i$ in each image $I_i \in \boldsymbol{I}$. An easy way to assess the similarity (or dissimilarity) of $\{I_i(\boldsymbol{q}_i)\}$ is to compute the color variance

$$e(\boldsymbol{p},d) = \frac{1}{|\boldsymbol{I}|}\sum_{I_i\in\boldsymbol{I}} (I_i(\boldsymbol{q}_i) - \bar{I}_c)^2, \tag{2.1}$$

where

$$\bar{I}_c = \frac{1}{|\boldsymbol{I}|}\sum_{I_i\in\boldsymbol{I}} I_i(\boldsymbol{q}_i) \tag{2.2}$$

is the mean color across views. The smaller the variance is, we say that the more photometrically consistent the depth is.

Equation (2.1) is an over-simplified measure of the so-called *photo consistency*. In this thesis, we call the image being processed the *reference image*, and the images used for measuring the photo consistency of its depth estimates the *secondary images*. For complicated or real-world scenes, more robust formulation and more reliable secondary images should be adopted in Equation (2.1).

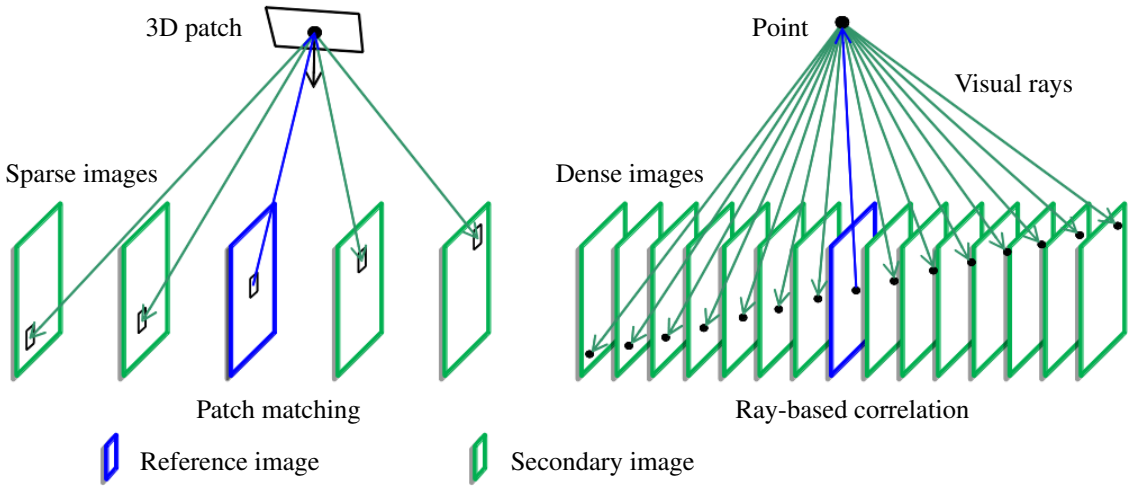


Figure 2.2: Two cross-correlation techniques. The patch matching (left) finds correspondences for a pixel neighborhood by projecting its 3D-patch candidates (each with an orientation hypothesis) into sparsely selected secondary images. The ray-based scheme (right) calculates photo consistency by comparing the appearances of densely sampled visual rays for the point hypotheses of a single pixel.

2.3.2 Patch Matching

The presence of image noise can easily lead to noisy depth estimates when finding pixel-to-pixel correspondences, e.g., Section 2.3.1. Another alternative is to match small windows, i.e., 2D patches where the pixel neighborhoods provide more local information to improve the correlation reliability. As shown in Figure 2.2 (left), each window is actually the image-space projection of a 3D-patch candidate. Since a square window in the reference image may get distorted when mapping onto another image depending on the patch orientation, an extra patch normal has to be assumed or computed.

The normalized cross-correlation (NCC) is a measure commonly used for evaluating photo consistency on patches. Given a pixel $\boldsymbol{p} = (x, y)$ in the reference image I_r, let d and $\boldsymbol{n}$ be the depth and normal of a tested surface patch whose 2D projection window W is centered at $\boldsymbol{p}$. Bailer *et al.* (2012) define $\boldsymbol{n} = (\boldsymbol{n}_x, \boldsymbol{n}_y)$ using the gradients of the tangent plane in x and y directions. The NCC score between I_r and a secondary image I_s is calculated by

$$\mathrm{NCC}(\boldsymbol{p}, d, \boldsymbol{n}, I_s) = \frac{\sum_{\boldsymbol{p}'} (I_r(\boldsymbol{p}') - \bar{I}_r)(I_s(\boldsymbol{q}') - \bar{I}_s)}{\sqrt{\sum_{\boldsymbol{p}'} (I_r(\boldsymbol{p}') - \bar{I}_r)^2} \cdot \sqrt{\sum_{\boldsymbol{q}'} (I_s(\boldsymbol{q}') - \bar{I}_s)^2}}. \tag{2.3}$$

In Equation (2.3), $\boldsymbol{p}' = (x', y') \in W$ and $\boldsymbol{q}'$ denotes its mapped pixel in I_s by using the

patch-orientation-aware depth of $\boldsymbol{p}'$:

$$d' = d \cdot \left(1 - (x - x')\boldsymbol{n}_x - (y - y')\boldsymbol{n}_y\right). \tag{2.4}$$

Equation (2.4) modifies the depth in both x and y directions, since depths are not equal in a tilted patch. $\bar{I}_r$ and $\bar{I}_s$ in Equation (2.3) represent the mean colors of the pixel regions $\{\boldsymbol{p}'\}$ and $\{\boldsymbol{q}'\}$, respectively. Subtracting the means makes the colors in the windows normalized and thus independent of the variations in image intensity or contrast.

As the photo consistency over all secondary images $\boldsymbol{S}(I_r)$, Bailer *et al.* (2012) calculate an occlusion-robust matching error by preferring high NCC values:

$$e(\boldsymbol{p}, d, \boldsymbol{n}) = \frac{\sum_{I_i \in \boldsymbol{S}(I_r)} 1 - \frac{\mathrm{NCC}(I_i)}{1 - \mathrm{NCC}(I_i)}}{\sum_{I_i \in \boldsymbol{S}(I_r)} \frac{1}{1 - \mathrm{NCC}(I_i)}}, \tag{2.5}$$

where $\mathrm{NCC}(I_i)$ is short for $\mathrm{NCC}(\boldsymbol{p}, d, \boldsymbol{n}, I_i)$.

Using all other images to measure the photo consistency of surface patch hypotheses (including depth and normal) is impractical: On one hand, the patches might be invisible or observed at a small angle in some views. On the other hand, the patch-level computation is more expensive than operating on single pixels. Therefore, unreliable secondary images and unnecessary calculations should be avoided for each reference image by only considering the views most suitable for correlation.

Robust Secondary-View Selection

Goesele *et al.* (2007) select secondary views by using a weighted sum of the SfM feature points which are visible in both the reference and the tested views. Based on this strategy, Bailer *et al.* (2012) differently prefer large viewing angles, and prioritize views by checking the visibilities of the reference-image content in both remaining and already selected views. The rating function to test an unselected view V_i for a reference view V_r is defined as

$$g(V_i) = \sum_{\boldsymbol{P}_f \in \boldsymbol{F}(V_r) \bigcap \boldsymbol{F}(V_i)} w_a(\boldsymbol{P}_f) \cdot w_s(\boldsymbol{P}_f) \cdot w_c(\boldsymbol{P}_f), \tag{2.6}$$

where $\boldsymbol{F}(V_r)$ and $\boldsymbol{F}(V_i)$ represent the feature points visible in V_r and V_i, respectively. Each common point $\boldsymbol{P}_f$ is weighted in three respects:

- **Angle** The function $w_a(\cdot)$ assigns $\boldsymbol{P}_f$ more weight, if its visual ray from V_i has a larger angle from the rays coming out of V_r and the views selected so far.

- **Scale** The function $w_s(\cdot)$ weights $\boldsymbol{P}_f$ more if the corresponding images I_r and I_i have similar scales, and less if I_i has a large angle but a markedly different scale from I_r.

- **Coverage** The function $w_c(\cdot)$ gives $\boldsymbol{P}_f$ more weight if it is sparsely recovered in already selected images.

See (Bailer *et al.*, 2012) for more detailed explanation and formulation. This view selection approach is used in our patch-based MVS (Chapter 4).

2.3.3 Ray-Based Correlation

Kim *et al.* (2013) proposed a ray-based reconstruction for high spatio-angular-resolution light fields. Unlike the sparsely captured photographs, such small-baseline images contain a large amount of redundant data. They utilize the extra scene information across views instead of intra-view (as in the patch matching) for reliable correlation, that enables sharp and edge-aligned depth discontinuities.

For each reference image I_r, a set of densely selected neighboring images are used as the secondary images $\boldsymbol{S}(I_r)$. Given a pixel $\boldsymbol{p}$ and a hypothetical depth d, the depth score is computed by

$$s(\boldsymbol{p},d) = \frac{1}{|\boldsymbol{S}(I_r)|} \sum_{I_i \in \boldsymbol{S}(I_r)} K\left(I_i(\boldsymbol{q}_i) - \bar{I}_c\right), \tag{2.7}$$

where $\boldsymbol{q}_i$ is the projection of $\boldsymbol{p}$ in I_i, and the kernel $K(\cdot)$ can be a Gaussian or any other bell-shaped function. The mean color is calculated using Equation (2.2) and updated by a mean-shift scheme (Comaniciu and Meer, 2002) for robustness to noise:

$$\bar{I}_c = \frac{\sum_{I_i \in \boldsymbol{S}(I_r)} K\left(I_i(\boldsymbol{q}_i) - \bar{I}_c\right) \cdot I_i(\boldsymbol{q}_i)}{\sum_{I_i \in \boldsymbol{S}(I_r)} K\left(I_i(\boldsymbol{q}_i) - \bar{I}_c\right)}. \tag{2.8}$$

As defined in Equation (2.7), Kim *et al.* (2013) simply aggregate the per-secondary-view scores by averaging because their method is targeted at the inputs with strictly constrained or uncomplicated camera trajectories. Comparatively, this thesis presents to use two-scale secondary-image selection and a more robust aggregation strategy to treat the more difficult unstructured videos (See Section 5.3.4).

2.4 Geometric Constraint

The 3D reconstruction can be constrained from another aspect, geometrically, to enforce surface consistency across views. This means that per-view estimates are judged to be consistent or conflicted with other views. In the following, we first depict this global cue (Section 2.4.1) as well as two visibility conflicts (Section 2.4.2) using two views, based on the analysis of (Merrell *et al.*, 2007). Then the consistency measure for multi-view depth maps, proposed by Bailer *et al.* (2012), is described in Section 2.4.3, that is also employed in our approach of Chapter 4 for outlier removal.

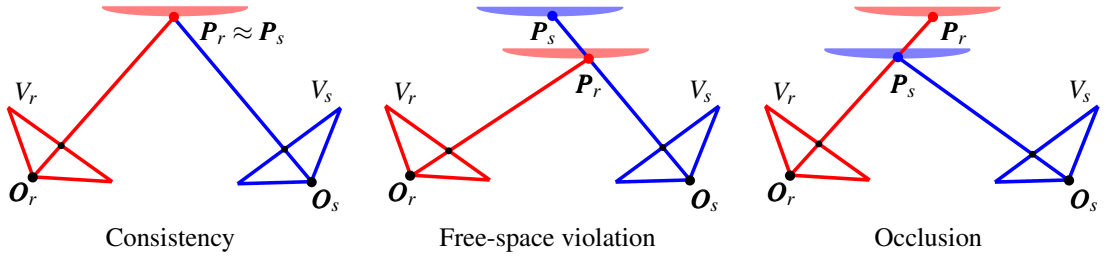


Figure 2.3: Geometric relations between two points $\boldsymbol{P}_r$ and $\boldsymbol{P}_s$, which are reconstructed from different views V_r and V_s, respectively. $\boldsymbol{O}_r$ and $\boldsymbol{O}_s$ are the corresponding camera centers. By assuming that $\boldsymbol{P}_s$ is correct, three cases may happen on $\boldsymbol{P}_r$: It is consistent with $\boldsymbol{P}_s$ (left), it violates the free space of $\boldsymbol{P}_s$ (middle), or it is occluded by $\boldsymbol{P}_s$ (right).

2.4.1 Global Consistency

The *global consistency* means that, for each scene point, the estimates at its 2D projections in all visible views should still create the same 3D point. Figure 2.3 simply formulates this constraint by considering two views V_r (the reference view) and V_s (the secondary view) with their respective camera centers at $\boldsymbol{O}_r$ and $\boldsymbol{O}_s$. Suppose that two points are reconstructed, $\boldsymbol{P}_r$ from V_r and $\boldsymbol{P}_s$ from V_s. If they are nearly in the same location (see Figure 2.3, left):

$$\frac{|d(\boldsymbol{O}_s,\boldsymbol{P}_s) - d(\boldsymbol{O}_s,\boldsymbol{P}_r)|}{d(\boldsymbol{O}_s,\boldsymbol{P}_s)} < \varepsilon, \tag{2.9}$$

we say that $\boldsymbol{P}_r$ is *consistent* or *coherent* with $\boldsymbol{P}_s$. In the equation, the function $d(\cdot,\cdot)$ computes the distance between two 3D points, and ε is a small threshold.

2.4.2 Visibility Conflicts

Assuming that the points calculated from all other views are correct, there are two possible cases representing an inconsistent estimate in a reference view:

- **Free-space violation** The 3D space between a scene surface and its observing camera is called the *free space*. The free space must be empty because otherwise this surface would be invisible in the considered view. Thereby, as illustrated in Figure 2.3 (middle), if $\boldsymbol{P}_r$ is on the visual ray through $\boldsymbol{P}_s$ and is closer than $\boldsymbol{P}_s$:

  $$d(\boldsymbol{O}_s,\boldsymbol{P}_r) < d(\boldsymbol{O}_s,\boldsymbol{P}_s), \tag{2.10}$$

  $\boldsymbol{P}_r$ is probably a wrong estimate since it *violates* the free space of $\boldsymbol{P}_s$.

- **Occlusion** On the contrary, if $\boldsymbol{P}_s$ lies on the visual ray through $\boldsymbol{P}_r$ and is closer

than $\boldsymbol{P}_r$ (see Figure 2.3, right):

$$d(\boldsymbol{O}_r, \boldsymbol{P}_s) < d(\boldsymbol{O}_r, \boldsymbol{P}_r), \tag{2.11}$$

we say that $\boldsymbol{P}_r$ is *occluded* by $\boldsymbol{P}_s$, because it is impossible that $\boldsymbol{P}_r$ is observed behind a real surface.

2.4.3 Multi-View Consistency Rating

To implement the consistency checking on multiple depth maps, Merrell *et al.* (2007) compare depth maps against each other. Rather than taking into account all other views for a reference depth map, Bailer *et al.* (2012) merely use depth maps of the secondary views found by their view-selection method (introduced in Section 2.3.2). They define a rating function:

$$r(\boldsymbol{p}) = \sum_{D_i \in \boldsymbol{S}(D_r)} 2\mathbf{B}(D_i) - \left| \sum_{D_i \in \boldsymbol{S}(D_r)} \mathbf{A}(D_i) - \sum_{D_i \in \boldsymbol{S}(D_r)} \mathbf{C}(D_i) \right|. \tag{2.12}$$

In the equation, the set of secondary-view depth maps for a reference depth map D_r is denoted as $\boldsymbol{S}(D_r)$, and $\boldsymbol{p}$ is a pixel in D_r. The *consistency rating* $r(\boldsymbol{p})$ is increased by consistency cases (represented by the indicator function $\mathbf{B} \in \{0,1\}$) and decreased by visibility conflicts. The difference between the free-space violation ($\mathbf{A} \in \{0,1\}$) and occlusion ($\mathbf{C} \in \{0,1\}$) cases is used as a penalty, because a surface point is unlikely to be too near and too far simultaneously. The depth estimate $D_r(\boldsymbol{p})$ is the more globally consistent, the higher $r(\boldsymbol{p})$ is.

Since the sampled secondary views observe the reference-view content at markedly different angles, there is very little chance that an inconsistent outlier appears in more than one depth map. Therefore, $D_r(\boldsymbol{p})$ is identified as a consistent estimate if

$$r(\boldsymbol{p}) \geq \begin{cases} 0 & \text{if } |\boldsymbol{S}(D_r)| < 2, \\ 2 & \text{if } |\boldsymbol{S}(D_r)| > 5, \\ 1 & \text{else.} \end{cases} \tag{2.13}$$

Chapter 3

Related Research

In this chapter, we summarize some existing work that are most relevant to this thesis. First, Sections 3.1 and 3.2 respectively go over the techniques for reconstructing scene geometry with high completeness and multi-view consistency. Then Section 3.3 reviews the approaches for merging multiple depth maps or point sets. The scalable reconstruction methods designed for large scenes are surveyed in Section 3.4. Finally, a summary is given in Section 3.5 to highlight the techniques that are used or developed in the thesis.

3.1 Dense Reconstruction

3D Patch Expansion

Dense geometry can be reconstructed by expending the existing 3D surface patches which are typically associated with image features. Furukawa and Ponce (2010) spread each patch by considering a group of neighboring image cells that satisfy certain criteria. The expansion procedure of (Locher *et al.*, 2016) works by assuming planar surfaces. A greedy strategy is used in (Habbecke and Kobbelt, 2007). The approach of Jancosek *et al.* (2009) is based on the patch quality which respects the patch pose, the reference image, and its feasibility information. In these approaches, the seed areas directly grow in 3D space. This avoids computation, cleaning up, and merging of overlapping depth maps which potentially contain noisy and redundant data.

PatchMatch

PatchMatch is a stochastic technique, originally proposed by Barnes *et al.* (2009) for finding dense approximate nearest-neighbor correspondences between the patches in a single image. By iterating between best-estimate propagation and random optimization, this method can obtain dense results faster than a brute-force search and even some acceleration algorithms (Mount and Arya, 1998; McNames, 2001). PatchMatch has been effectively applied to handle the multi-view matching problem for photo collections.

To propagate the depth estimates, Goesele *et al.* (2007) prioritize the candidate pixels based on their matching confidences to avoid spreading into unreliable regions. As in

(Bleyer *et al.*, 2011; Besse *et al.*, 2012), a simpler strategy is utilizing a top-left to bottom-right traversal order by considering the left and upper neighbors of each pixel in odd iterations followed by a reversed order in even iterations considering the right and lower neighbors. The above pixel scanning methods are difficult to parallelize. Bailer *et al.* (2012) proposed a more GPU-friendly scheme. Their propagation operates along the image rows and columns by considering three neighbors for each pixel. However, as presented in Chapter 4, there is still space for modifying their method to yield more efficiency gains.

Multi-Scale Calculation

The matching ambiguities, e.g., due to poor textures, can be solved by restricting the computations in a multi-scale manner. Kim *et al.* (2013) presented a fine-to-coarse strategy for dense reconstruction from high spatio-angular light fields. Beginning from the accurate edge depth at the original image resolution, they estimate depth for the unrecovered areas by iterating between image downscaling and depth calculation which uses the estimates at the finer scale as the bounds. Conversely, Kang and Medioni (2014) adopt a coarse-to-fine discrete-continuous variational method. Though these approaches can obtain dense depth maps, depth discontinuities might be produced in large homogeneous areas, including our method depicted in Chapter 4.

Diffusion and Edge-Aware Filtering

If sufficient object edges or surface textures are already reconstructed accurately, their estimates can be used to interpolate smooth surfaces for the areas in between. Stefanoski *et al.* (2013) simply diffuse the estimates of image features to get a dense depth map. The domain transform filter introduced by Lang *et al.* (2012) is also employed (Rzeszutek and Androutsos, 2013, 2015) which can preserve the sharp depth discontinuities. Unlike the techniques of patch expansion and PatchMatch, diffusion and edge-aware filtering lead to less computational effort, since the photo consistency is not measured for the depth hypotheses. However, as discussed in Section 5.4.1, these 2D-space interpolation methods may generate distorted surfaces in 3D space.

Planarity Assumption

Simultaneous Localization and Mapping (SLAM) is a real-time version of SfM. Like SfM, most of the SLAM algorithms merely create sparse points conveying little semantics. Some advanced SLAM methods (Chekhlov *et al.*, 2007; Gee *et al.*, 2007) increase surface completeness by replacing the point features with 3D planes. These two kinds of features can also be combined (Martinez-Carranza and Calway, 2010; Kundu *et al.*, 2014). (Sinha *et al.*, 2009) and (Furukawa *et al.*, 2009) calculate piecewise planar depth maps by combing planes with 3D lines. Gallup *et al.* (2010) distinguish whether a region

is planar using image segmentation. Bódis-Szomorú *et al.* (2015) proposed a dense MVS algorithm that produces superpixel meshes. The planarity assumption tends to create the over-segmentation effect. In (Zhang *et al.*, 2009), this effect is only confined in the initialization which is followed by an estimate refinement step using bundle optimization.

Other Approaches

In other dense MVS approaches, a smoothness term is incorporated into the energy minimization problem (Yu and Gallup, 2014; Kang and Medioni, 2014, 2015), and dynamic programming is also used (Pollefeys *et al.*, 1998, 2004). Hernandez and Schmitt (2004) fuse the texture and silhouette information. In the patch matching, Narayanan *et al.* (1998) adopt multiple image baselines and Bradley *et al.* (2008) use scaled windows. Jancosek and Pajdla (2011) recover the difficult surfaces which are poorly sampled by the input point cloud by using the Visual-Hull (Laurentini, 1994).

3.2 Multi-View Consistency

Energy Minimization

An early work (Kolmogorov and Zabih, 2002) obtain consistent estimates by addressing the global energy minimization problem via graph cuts. In this approach, the input images are treated symmetrically and the visibility constraint is encoded properly. Campbell *et al.* (2008) introduce an unknown state for those pixels with inconsistent depth and use a subsequent global regularization step to recover the surfaces in these places. The iterative approach (Zhang *et al.*, 2009) performs bundle optimization. In (Furukawa and Ponce, 2010), the global consistency and a weak form of regularization are enforced to eliminate the incorrect patches after each expansion step. Schoenberger *et al.* (2016) effectively combines the photometric and geometric constraints for view selection and post-processing in depth and normal estimation.

Space Carving

Spacing carving is a volumetric technique that reconstructs surfaces in the discretized 3D space. Pan *et al.* (2009) presented a probabilistic space carving scheme. The tetrahdra intersected by a visual ray are labeled as free space (see Section 2.4.2), and all others as occupied (inside the objects). Then the border of tetrahdra is extracted as the surface. The binary labeling is solved in (Vu *et al.*, 2012) by using a minimum *s-t* cut. The space carving strategy has been extended to work incrementally (Yu and Lhuillier, 2012; Hoppe *et al.*, 2013; Sugiura *et al.*, 2013). Like other volumetric approaches, such technique is only suitable for modeling single objects or small scenes due to the expensive memory requirements.

Cross-View Propagation and Filtering

Global consistency can be achieved by spreading the per-view information across views. Bleyer *et al.* (2011) proposed a modified PatchMatch pipeline for stereo video. The estimates are propagated in three domains: spatial, cross-view, and temporal. The consistency problem also exists in other fields, e.g., optical/scene flow estimation and segmentation from videos (Lang *et al.*, 2012; Vogel *et al.*, 2014; Hur, 2016). Lang *et al.* (2012) introduced a temporal smoothness assumption for getting consistent optical flow. They update the estimates by iterating between spatial and temporal filtering passes. Their filtering in the temporal domain follows the calculated motion vectors.

3.3 Depth-Map/Point-Cloud Merging

Depth Map Merging

As introduced in Section 1.1, the depth-map-based MVS algorithms need to merge the per-view estimates to derive the scene geometry. In order to reconstruct an accurate and smooth model, the clearly wrong estimates should be filtered out beforehand. Several systems (Merrell *et al.*, 2007; Pollefeys *et al.*, 2008) calculate noisy, photo-consistent depth maps using a simple, fast approach for real-time performance. And, most of the PatchMatch methods also compute the depth view-independently. Therefore, a large amount of inconsistent estimates are produced, that are subsequently removed by a consistency checking (Merrell *et al.*, 2007; Bailer *et al.*, 2012). The video-oriented scheme of Kang and Medioni (2015) removes the wrong estimates which are inconsistent among small-baseline neighboring views and the conflicts violating the visibility constraint across large-baseline non-neighbors. Li *et al.* (2010) proposed to utilize bundle optimization. Hu and Mordohai (2012) use a least commitment method to defer the hard decisions.

Signed Distance Function

Some volumetric methods construct signed distance function for merging point clouds. Volumetric Range Image Processing (VRIP) (Curless and Levoy, 1996) uses regular voxel grid since the point scale is not considered (see Section 6.3 for the point scale definition). (Calakli and Taubin, 2011) respects the scale information but it seeks the globally optimal surfaces. Schroers *et al.* (2012) adopt the closest signed distances which are more accurate than the directional signed distances.

Implicit Function

Another type of point merging techniques builds upon implicit function. Ohtake *et al.* (2003) exploit the function for fitting multiple points. Fuhrmann and Goesele's meth-

ods respect the point scale by utilizing discretized implicit function in (Fuhrmann and Goesele, 2011) and extending it to floating scale in (Fuhrmann and Goesele, 2014). Our point merging presented in Section 6.6 is based on the latter work, because the continuous function allows us to evaluate the surface hypotheses anywhere.

Other Approaches

A simpler method (Bradley *et al.*, 2008) builds a Volumn-Surface Tree (Boubekeur *et al.*, 2006) and computes the averages of the point positions in each leaf node. But the density of the simplified points relies on the node size. The Poisson Surface Reconstruction (Kazhdan and Hoppe, 2013) defines the point scale as the density of the accumulated points, thus struggling with high-frequency noise. Múcke *et al.* (2011) generate a global confidence map and recover surfaces via graph cuts. This approach has difficulty in finding the exact maximum from the unsigned confidences. Moving least-squares (Shen *et al.*, 2005; Cuccuru *et al.*, 2009) and height maps Zheng *et al.* (2012) are also used for this task.

3.4 Scalable Reconstruction

Large-Scale SfM and SLAM

The SfM and SLAM problems are aimed at video sequences and have been addressed at large scales. The SLAM system (Engel *et al.*, 2014) locally tracks the camera paths and builds globally consistent maps (pose graphs of the keyframes) for the whole environment. In (Bourmaud and Mgret, 2015), the camera movements and large scenes are divided into multiple submaps which are estimated by solving the so-called know rotation problem. A loopy belief propagation scheme is subsequently used to align the estimates of submaps. The SfM pipeline (Resch *et al.*, 2015) exploits the strong internal coherence in video data instead of the traditionally used large image baselines. The locally reconstructed camera tracks, loop closing, and linking between the tracks are integrated in this approach.

Incremental and Online MVS

Some systems incrementally increase completeness of the global model represented by patches (Kang and Medioni, 2015; Locher *et al.*, 2016), triangles (Newcombe and Davison, 2010), surfels (Weise *et al.*, 2009; Kolev *et al.*, 2014), or points (Keller *et al.*, 2013). For memory reduction, voxel hashing (Nießner *et al.*, 2013) and binary voxel grid (Reichl *et al.*, 2015) are adopted. In (Pizzoli *et al.*, 2014), a combination of Bayesian estimation and convex optimization is used. Fioraio *et al.* (2015) presented an online subvolume registration method. (Kang and Medioni, 2015) maintains and updates a set of patch tracks during the live scanning. Similarly to SLAM, the performance of these solutions relies

largely on the simultaneously computed camera poses. And most of them concentrate on the modeling of RGB-D input or small scenes. Sugiura *et al.* (2013) employ an incremental tetrahedra carving scheme. Kuhn and Mayer (2015) progressively decompose the large, sparse point cloud for parallel reconstruction. A total variation prior is also used recently (Kuhn *et al.*, 2016). Locher *et al.* (2016) utilize a prioritized region growing algorithm. By adaptively expending and branching 3D patches, the density, resolution, and precision of the reconstructed point cloud are all improved. Unlike these methods, our approach proposed in Chapter 6 focuses on the recovery of large-area homogeneous surfaces. Moreover, our diffusion-based occlusion handling has less computational cost, and can be parallelized more easily.

(Next-Best-) View Selection

To reduce overhead of computation and memory, the views can be pre-selected from all available (Goesele *et al.*, 2007; Snavely *et al.*, 2008) or the clustered (Furukawa *et al.*, 2010; Mauro *et al.*, 2014a) views. Goesele *et al.* (2007) select views based on image resolution and baselines. Snavely *et al.* (2008) build skeletal graphs and Mauro *et al.* (2014a) use an integer linear programming model. Riemenschneider *et al.* (2014) predict the best views for semantic labeling exploiting the 3D mesh model from MVS.

Pre-selection neglects the scene's geometric properties. This problem can be solved by the Next-Best-View (NBV) strategy. Instead of requiring user intervention (Hoppe *et al.*, 2012), some approaches work in a content-aware way. In (Haner and Heyden, 2012), covariance propagation is implemented and the views decreasing the estimate uncertainty are preferred. The triangle-mesh-based work (Dunn and Frahm, 2009) additionally incorporates image resolution and 2D visual saliency. In (Mauro *et al.*, 2014b), the distance to object and viewing angle with the already selected views are considered. Hornung *et al.* (2008) find the NBV by maximizing the scene's completeness and visibility within a voxel grid, and refine unreliable estimates using the views that improve the photo consistency. The optical-flow-guided approach (Newcombe and Davison, 2010) exploits a base mesh, and ray casting is used in (Mendez *et al.*, 2016). Some approaches (Newcombe and Davison, 2010; Mauro *et al.*, 2014b,a) directly operate on SfM points.

View Clustering

The objective of existing view clustering methods is independent and parallel processing of multiple view clusters. Zaharescu *et al.* (2008) cluster views based on a coarse initial geometry. In (Furukawa *et al.*, 2010), cluster division and view addition are performed iteratively to enforce the size and coverage constraints. In (Ladikos *et al.*, 2009), graph partitioning and spectral clustering are exploited by considering both visibility information and camera configuration. Mauro *et al*. perform graph-based dominant set clustering in (Mauro *et al.*, 2013), and utilize leveraged affinity propagation in (Mauro *et al.*, 2014a).

3.5 Summary

The following is a summary of the existing techniques on which the new approaches introduced in this thesis are based:

- Our MVS algorithm for photographs (Chapter 4) builds upon the PatchMatch approach of Bailer *et al.* (2012). We integrate a more local depth propagation scheme and the coarse-to-fine framework of Kang and Medioni (2014) for more efficient region growing. Moreover, we borrow the idea of temporal filtering from (Lang *et al.*, 2012) to produce coherent depth values over camera views, that lighten the burden of depth map merging algorithms.

- The proposed dense depth map estimation scheme for videos (Chapter 5) is based on the edge-first strategy of Kim *et al.* (2013). Their ray-wise edge depth calculation is enhanced for more robustness against occlusion and out-of-image projection. For textureless regions, we exploit the smoothness assumption instead of the photo-consistency constraint to remove wrong depth discontinuities, but avoids the over-segmentation effect of planarity-based methods. Our depth diffusion handles the perspective distortions produced by classic interpolation algorithms. Our occlusion removal is more efficient on GPUs than the techniques of energy minimization and space carving.

- Our scalable MVS system (Chapter 6) belongs to the techniques of incremental reconstruction and view selection. Differently, we target at progressively handling the potential occlusions by solving the NBV problem, and our NBV solution works on dense surface points rather than the SfM features. Furthermore, in contrast to the view pre-clustering methods, we achieve high efficiency by automatically generating view (or point) clusters after the partial reconstruction. For point merging, we adopt the implicit function of (Fuhrmann and Goesele, 2014). This approach is modified to output a concise point set with the highest surface resolution instead of a triangle mesh with mixed resolution. And the point uncertainty is respected in our isosurface search.

Chapter 4

Globally Consistent MVS for Sparsely Sampled Photos

This chapter focuses on creating depth maps for traditional MVS input, large-baseline and unorganized photographs. The results can be post-transformed into a point cloud or 3D surface meshes. We adopt the PatchMatch technique of Bailer *et al.* (2012) but with various modifications. The main properties of our approach are two-fold: faster propagation of optimal estimates and, more importantly, higher cross-view depth consistency.

4.1 Introduction

Bailer *et al.* (2012) extend the core PatchMatch to support scale-robust reconstruction and slanted patches. They alternate between left-/rightward and up-/downward propagations along quite long scanlines. Though fast estimation are achieved by exploiting GPU parallelism, the iterative long propagation steps still slow down their entire system.

A crucial drawback of this method is that the per-view depth maps are computed *independently*, so inconsistent outliers may exist and probably grow during depth propagation, producing unstable estimates across views. This leads to a large amount of estimates removed by consistency checking in the depth map merging stage, and diminishes the reconstruction quality. Likewise, this problem applies to most other depth-map-based methods, as reviewed in Section 3.2.

In this chapter, we propose a novel PatchMatch system that combines correlation certainty and depth consistency:

1) We use a parallel, local propagation scheme to grow estimates along short scanlines, and a faster coarse-to-fine strategy to fill in larger holes.

2) Our main objective is to enforce global consistency among the depth data of different views. Lang *et al.* (2012) obtain temporal coherence of optical flow by filtering along motion paths, which simultaneously uses and calculates per-pixel results. Inspired by this work, we impose a cross-view filtering stage based on the free-space constraint (see Section 2.4.2) and variance filtering. Our algorithm alternates between optimizations of the photometric and geometric consistency measures. Thereby, we can balance

the depth consistency against the unreliable patch correlation in poorly textured areas. Moreover, filtering across views also fills holes using the depth from other views. With noisy patches and spikes excluded earlier, our approach puts less stress on the subsequent fusion steps, thus producing denser surfaces with less outliers.

Introduction of the proposed approach will start from a brief description of our workflow in Section 4.2.

4.2 System Overview

Figure 4.1 shows our advanced PatchMatch workflow for depth map creation. The main contributions compared with (Bailer *et al.*, 2012) are highlighted in the figure.

After initializing a sparse depth map for each view, all input images and depth maps are down-scaled. Next, we start the hierarchical estimation. At each scale, a propagation-filtering-propagation strategy is used. Concretely, the propagation steps spread the best estimates of both depth and normal (see Section 2.3.2) by traversing pixels and minimizing matching errors along short paths. The cross-view filtering rejects inconsistent outliers and spreads reliable depth data to different views. Then the depth and normal maps are up-scaled for the estimation at the next consecutive scale. Finally, visibility conflicts are removed and the depth maps are smoothed.

As depicted in Figure 4.1, we use three scales with scaling factor of 2, each performing three cross-view filtering passes. In the random optimization steps, arbitrarily sampled depths and normals are tested on the up-scaled results (see (Bailer *et al.*, 2012) for more details).

We use the state-of-the-art SfM implementation Bundler (Snavely *et al.*, 2006) to derive camera poses and feature points with available visibility information. Each reference image selects at most 6 secondary images using the scale-robust method of Bailer *et al.* (2012) (see Section 2.3.2).

The rest of this section describes how we initialize and post-process the estimates in our approach. Then the detailed introduction of our contributions is organized as follows: We describe the local propagation in Section 4.3 and the hierarchical framework in Section 4.4. Our major contribution, incorporation of cross-view depth filtering, will be presented in Section 4.5.3.

4.2.1 Initialization

Given a reference view, let D_k, N_k, and E_k be its depth, normal, and matching-error maps at scale k, respectively. For each pixel $\boldsymbol{p}$, if it corresponds to a SfM feature, we initialize its finest-scale depth $D_0(\boldsymbol{p})$ by projecting this feature point from 3D space to 2D, and let $D_0(\boldsymbol{p}) = 0$ otherwise. Its patch normal is initialized fronto-parallel at the coarsest scale, i.e., $N_2(\boldsymbol{p}) = \{0,0\}$ (see Section 2.3.2 for the representation of normal). Before calculation at each scale, E_k is re-calculated using the existing depth and normal values.

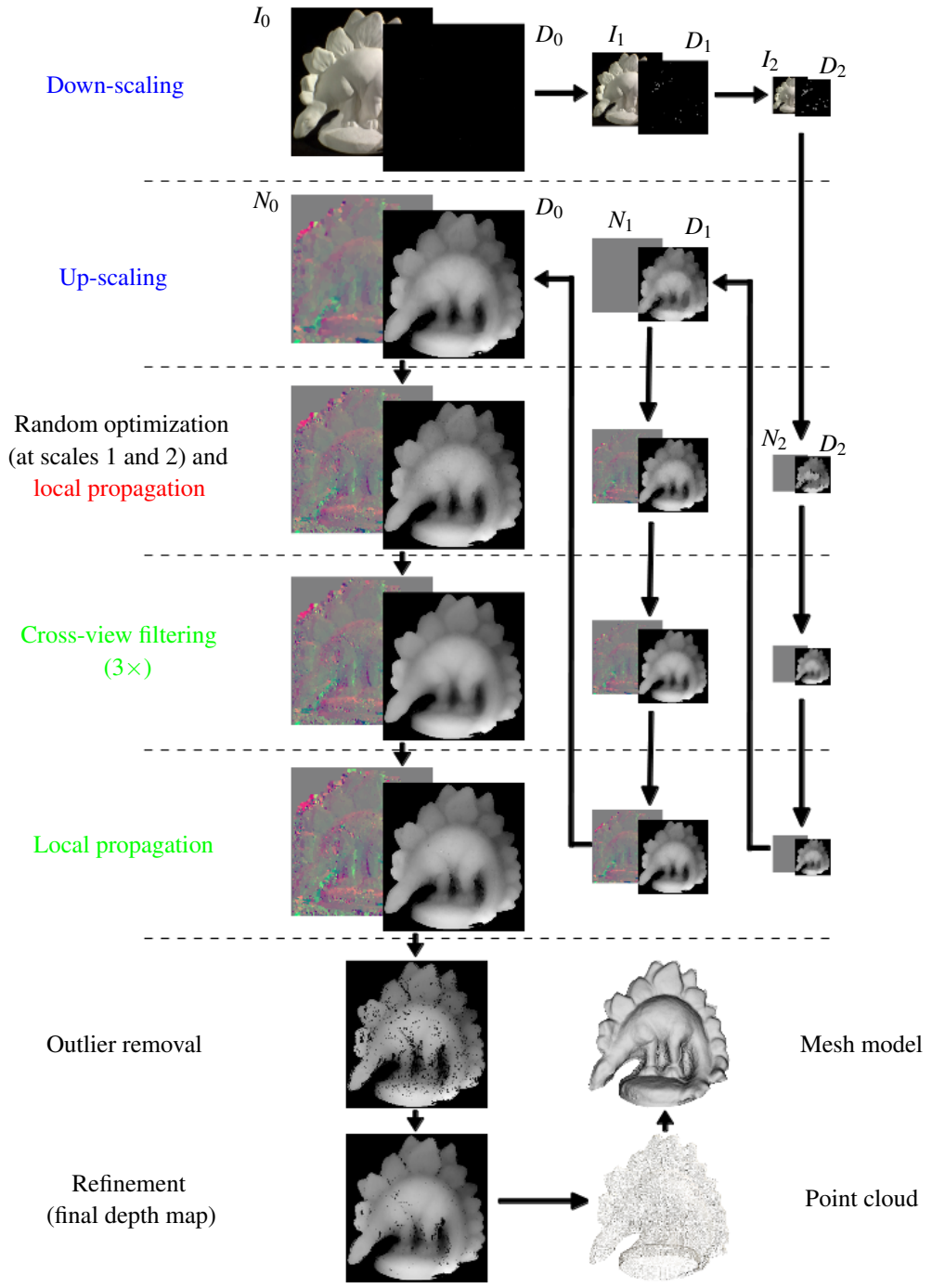


Figure 4.1: Our processing pipeline for one view of Dino dataset. I_k, D_k, and N_k correspondingly represent the image, depth map, and normal map at scale k. I_0 is the input image, and D_0 is initialized from Bundler (Snavely *et al.*, 2006). Our key steps, i.e., main differences from (Bailer *et al.*, 2012), are highlighted: hierarchical framework (blue), local propagation (red), and cross-view filtering with an additional propagation pass (green). Background areas are removed by thresholding.

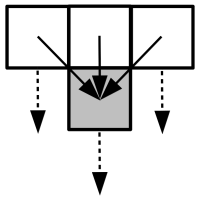
4.2.2 Post-Processing

With the resulting depth maps, inconsistent outliers conflicting Equation (2.13) are eliminated. After that, we use a domain transform filter (Gastal and Oliveira, 2011) with $\sigma_s = 30$ and $\sigma_r = 0.03$ to refine the results by filling the holes and then filtering the noise. We perform three complete filtering passes (horizontal+vertical 1D passes for each) for both tasks. In the hole filling the existing depth estimates are not altered. To avoid involving inconsistent values, we also remove outliers before each complete filtering pass and after the last pass.

4.3 Local Propagation

The PatchMatch technique calculates a dense depth map by iteratively propagating the seed's data or newly created estimates, which may come from initialization, random optimization, or the cross-view filtering in our approach. In this section, we first describe how estimates are propagated between pixels, and then explain our local pixel-traversal scheme for fast propagation within the whole depth map.

4.3.1 Propagation Between Pixels



In the propagation, good (depth and normal) estimates are dispersed into the neighborhoods if the correlation measure is improved. Each pixel tests the patch candidates from at most three neighbors (assigned depth), which are called *query pixels* and selected depending on the traversal direction. An example of downward propagation is shown on the right.

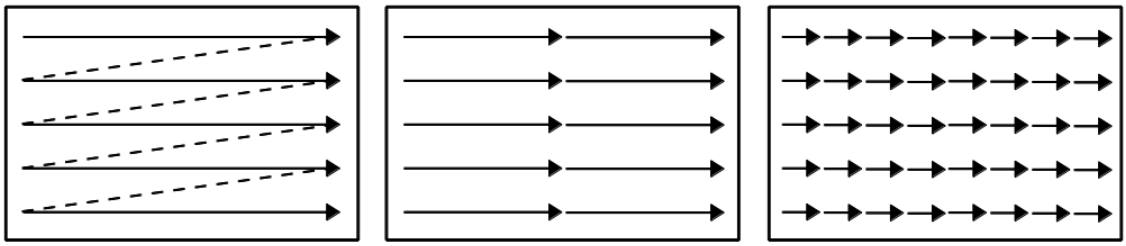
More concretely, assume a pixel $\boldsymbol{p} = (x, y)$ in depth map D_k, and one of its query pixels $\boldsymbol{q} = (x', y')$. A depth hypothesis for $\boldsymbol{p}$ is calculated based on the surface patch at $\boldsymbol{q}$, which is similar to Equation (2.4):

$$d = D_k(\boldsymbol{q}) \cdot \left(1 - 2^k(x' - x)\boldsymbol{n}_x - 2^k(y' - y)\boldsymbol{n}_y\right), \tag{4.1}$$

where $N_k(\boldsymbol{q}) = (\boldsymbol{n}_x, \boldsymbol{n}_y)$ is the normal estimate of $\boldsymbol{q}$, and 2^k considers the effect of image scaling (see Section 4.4). If $\boldsymbol{p}$ has no estimate or we get a smaller matching error $e := e(\boldsymbol{p}, d, N_k(\boldsymbol{q}))$ using Equation (2.5), i.e., $e < E_k(\boldsymbol{p})$, we update its depth, normal, and matching error by $D_k(\boldsymbol{p}) \leftarrow d$, $N_k(\boldsymbol{p}) \leftarrow N_k(\boldsymbol{q})$, and $E_k(\boldsymbol{p}) \leftarrow e$.

4.3.2 Local Traversal Scheme

In order to spread estimates over all pixels, the propagation should be performed along some form of scanline, e.g., simply from top-left to bottom-right for rightward propaga-



(Besse *et al.*, 2012) (Bailer *et al.*, 2012) Our local propagation

Figure 4.2: Different traversal schemes for rightward propagation.

tion (see Figure 4.2, left). Bailer *et al.* (2012) traverse pixels following parallel scanlines with length of half the image height (width) in the vertical (horizontal) propagation (see Figure 4.2, middle). They scan down- and upward (left- and rightward) on GPUs simultaneously, and achieve faster convergence. However, the speed of this scheme is still limited by too few scanlines to keep the GPU busy.

Therefore, we shorten the traversal distance such that more GPU threads can be assigned. As shown in Figure 4.2 (right), in the vertical (horizontal) propagation, we select an eighth of the image height (width) as the length of each scanline. In every other iteration vertical and horizontal propagations are applied alternately.

4.4 Hierarchical Framework

For textureless regions with few initial estimates, one propagation alone at the original scale is generally insufficient due to the locality of short scanlines. We solve this problem by down-scaling the depth map and spreading the sparse data into the neighborhoods. This way, one propagation at the coarsest scale can fill most of the holes. Then the estimates are used for the consecutive finer scale by up-scaling. In practice, we also down-scale the images and up-scale the estimated normal maps. This hierarchical framework is illustrated in Figure 4.1. Another benefit is the reduced overall time of depth estimation, since the scaling process is negligible compared with the speed-up of propagation (see Section 4.6.2).

We base our up-scaling procedure on the joint bilateral upsampler (Kopf *et al.*, 2007), that guides data interpolation by considering the information from the high-resolution image. For an image at scale k, denoted as I_k, let S_k represent its depth map D_k or a channel of its normal map N_k. We up-scale S_{k+1} to S_k by calculating

$$S_k(\boldsymbol{p}) = \frac{\sum_{\boldsymbol{p}'} w(\boldsymbol{p},\boldsymbol{p}') \cdot S_{k+1}(\boldsymbol{p}')}{\sum_{\boldsymbol{p}'} w(\boldsymbol{p},\boldsymbol{p}')}, \tag{4.2}$$

where $\boldsymbol{p}' \in W$ and W is a neighborhood window around pixel $\boldsymbol{p}$. Here, $\boldsymbol{p}'$ may be a pixel

at finer scale k or the corresponding pixel at coarser scale $k+1$. The weight

$$w(\boldsymbol{p},\boldsymbol{p}') = s(||\boldsymbol{p}-\boldsymbol{p}'||)\cdot r\left(I_k(\boldsymbol{p})-I_k(\boldsymbol{p}')\right) \tag{4.3}$$

depends on the Gaussian functions $s(\cdot)$ and $r(\cdot)$ with $\sigma_s = 0.4$ and $\sigma_r = 0.2$, decreasing with larger spatial distance and larger intensity difference, respectively. We down-scale images and depth maps in a similar way using

$$S_{k+1}(\boldsymbol{p}) = \frac{\sum_{\boldsymbol{p}'} w(\boldsymbol{p},\boldsymbol{p}')\cdot S_k(\boldsymbol{p}')}{\sum_{\boldsymbol{p}'} w(\boldsymbol{p},\boldsymbol{p}')}, \tag{4.4}$$

$$w(\boldsymbol{p},\boldsymbol{p}') = s(||\boldsymbol{p}-\boldsymbol{p}'||)\cdot r\left(I_{k+1}(\boldsymbol{p})-I_{k+1}(\boldsymbol{p}')\right). \tag{4.5}$$

Because the sparse depth data often belong to isolated pixels in the initial depth maps, most of depth values in the first round are just copied into their neighbors during down-scaling, except in the case that two spread regions interfere.

4.5 Cross-View Depth Filtering

So far, our propagation is per view, which may lead to global instabilities of depth values. Inspired by the temporally consistent optical flow estimation (Lang *et al.*, 2012), after propagation of all views at the coarsest scale, we perform a cross-view filtering for each reference view to improve the depth consistency. Then a second propagation spreads the optimized estimates. The results are updated in each propagation-filtering-propagation iteration at finer scales. See Figure 4.1 for our development.

The temporal filtering of Lang *et al.* (2012) follows optical flow vectors to avoid averaging across object edges. Similarly, our cross-view filtering considers the projection relationships of pixels between views using the depth information. For each depth, we find the corresponding pixels in the secondary views, and project them back into the reference view obtaining new depth candidates. These candidates are weighted by the depth difference between the reference and secondary views to get an optimized depth. In some cases, this depth projection from secondary views can even fill holes in the reference, spawning further, more consistent propagation.

In Section 4.5.1, we describe how to obtain the depth candidates. Section 4.5.2 defines another depth-consistency measure, that enables higher precision than Equation (2.12) and is utilized for weighting the candidates in the cross-view filtering (Section 4.5.3).

4.5.1 Depth Candidates

Given a depth map D_r of the reference view, let $\boldsymbol{S}(D_r)$ be the depth maps of its secondary views. For a pixel $\boldsymbol{p}$ assigned depth in D_r, we can obtain two depth candidates $d_{i\to r}(\boldsymbol{p})$ and $d_{r\to i\to r}(\boldsymbol{p})$ using each secondary depth map $D_i \in \boldsymbol{S}(D_r)$, as shown in Figure 4.3.

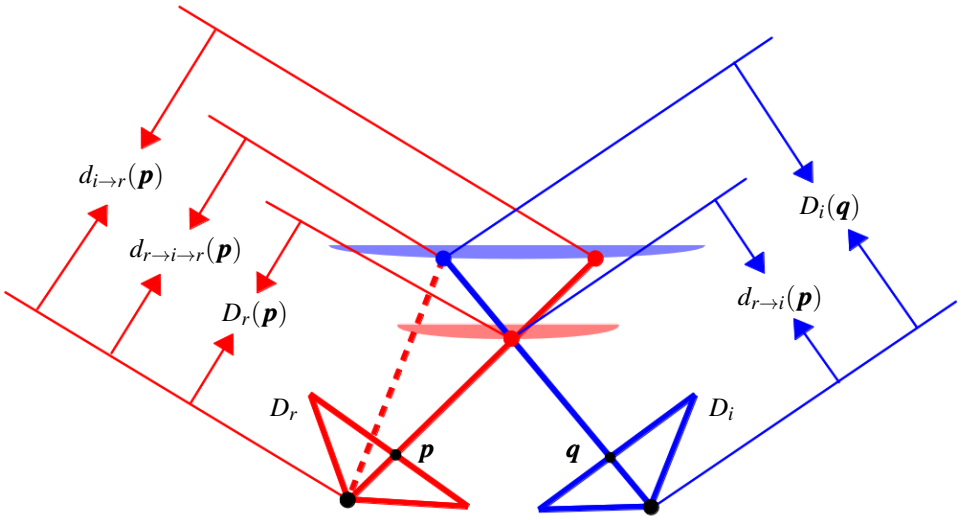


Figure 4.3: The two depth candidates, $d_{i\to r}(\boldsymbol{p})$ and $d_{r\to i\to r}(\boldsymbol{p})$, for a pixel $\boldsymbol{p}$ in the reference depth map D_r, which are obtained from a secondary depth map D_i.

To get $d_{i\to r}(\boldsymbol{p})$, we back-project all 3D-point estimates of D_i to D_r and pick the least distant depth at position $\boldsymbol{p}$. To obtain $d_{r\to i\to r}(\boldsymbol{p})$, we back-project the points of D_r to D_i, obtaining a depth $d_{r\to i}(\boldsymbol{p})$ and a position $\boldsymbol{q}$ in the view of D_i. Then the depth $D_i(\boldsymbol{q})$ is projected back to D_r to get the new candidate.

4.5.2 Depth Variance

The consistency-rating values computed by Equation (2.12) can only be integer and depend on the number of selected secondary views. Therefore, we introduce a more precise way of measuring depth coherence for our cross-view filtering (Section 4.5.3) while adopting Equation (2.12) in the post-processing for outlier removal.

Specifically, if the estimate $D_r(\boldsymbol{p})$ is not conflicted with the secondary views, the relative depth difference in the reference view

$$\Delta d_{i\to r}(\boldsymbol{p}) = \frac{D_r(\boldsymbol{p}) - d_{i\to r}(\boldsymbol{p})}{D_r(\boldsymbol{p})} \tag{4.6}$$

should be close to 0 for each D_i (see Section 2.4.1). That means, all the values of $\{d_{i\to r}(\boldsymbol{p})\}$ should have a low variance relative to $D_r(\boldsymbol{p})$. Likewise, we also consider the relative depth difference in the secondary view

$$\Delta d_{r\to i}(\boldsymbol{p}) = \frac{D_i(\boldsymbol{q}) - d_{r\to i}(\boldsymbol{p})}{D_i(\boldsymbol{q})}, \tag{4.7}$$

since in this case each $d_{r\to i}(\boldsymbol{p})$ should also have a close value to the estimate $D_i(\boldsymbol{q})$.

By using the minimum of $\Delta d_{i\to r}(\boldsymbol{p})^2$ and $\Delta d_{r\to i}(\boldsymbol{p})^2$ for each D_i, the *depth variance* at pixel $\boldsymbol{p}$ is defined as

$$v(\boldsymbol{p}) = \frac{1}{|\boldsymbol{S}(D_r)|} \sum_{D_i \in \boldsymbol{S}(D_r)} \min\left(\Delta d_{i\to r}(\boldsymbol{p})^2, \Delta d_{r\to i}(\boldsymbol{p})^2\right). \tag{4.8}$$

We say that the depth $D_r(\boldsymbol{p})$ is sufficiently consistent if $v(\boldsymbol{p})$ is lower than a threshold [1]:

$$v(\boldsymbol{p}) < \tau_1. \tag{4.9}$$

4.5.3 Cross-View Filtering

We enhance the depth consistency via a weighted average strategy. After collecting all candidates $\{d_{i\to r}(\boldsymbol{p})\}$ and $\{d_{r\to i\to r}(\boldsymbol{p})\}$, an optimized depth for pixel $\boldsymbol{p}$ is calculated by

$$\hat{d}(\boldsymbol{p}) = \frac{w_r(\boldsymbol{p}) \cdot D_r(\boldsymbol{p}) + \sum_{D_i \in \boldsymbol{S}(D_r)} \left(w_{i\to r}(\boldsymbol{p}) \cdot d_{i\to r}(\boldsymbol{p}) + w_{r\to i\to r}(\boldsymbol{p}) \cdot d_{r\to i\to r}(\boldsymbol{p})\right)}{w_r(\boldsymbol{p}) + \sum_{D_i \in \boldsymbol{S}(D_r)} \left(w_{i\to r}(\boldsymbol{p}) + w_{r\to i\to r}(\boldsymbol{p})\right)}. \tag{4.10}$$

In the equation, the weighting functions are defined as follows:

1. If $\boldsymbol{p}$ has no estimate, i.e., $D_r(\boldsymbol{p}) = 0$, or its depth is significantly inconsistent, i.e., $v(\boldsymbol{p})$ is higher than another looser threshold[1]

   $$\tau_2 > \tau_1, \tag{4.11}$$

   the candidates $\{d_{r\to i\to r}(\boldsymbol{p})\}$ are unavailable or unreliable. Hence $\hat{d}(\boldsymbol{p})$ is obtained simply using the average of the candidates $\{d_{i\to r}(\boldsymbol{p})\}$ by letting $w_r(\boldsymbol{p}) = 0$, $w_{i\to r}(\boldsymbol{p}) = 1$, and $w_{r\to i\to r}(\boldsymbol{p}) = 0$. In this case, our cross-view filtering fills the hole here or corrects the estimate with a consistent, optimized value.

2. Otherwise, we simply set $w_r(\boldsymbol{p}) = 1$ for the estimate $D_r(\boldsymbol{p})$, and define the other two weights as Gaussian functions considering the relative depth differences:

   $$w_{i\to r}(\boldsymbol{p}) = \exp\left(-\frac{\Delta d_{i\to r}(\boldsymbol{p})^2}{\sigma_d(\boldsymbol{p})^2}\right), \tag{4.12}$$

   $$w_{r\to i\to r}(\boldsymbol{p}) = \exp\left(-\frac{|\Delta d_{i\to r}(\boldsymbol{p}) \cdot \Delta d_{r\to i\to r}(\boldsymbol{p})|}{\sigma_d(\boldsymbol{p})^2}\right). \tag{4.13}$$

   When computing $w_{r\to i\to r}(\boldsymbol{p})$, we incorporate another relative depth difference

   $$\Delta d_{r\to i\to r}(\boldsymbol{p}) = \frac{D_r(\boldsymbol{p}) - d_{r\to i\to r}(\boldsymbol{p})}{D_r(\boldsymbol{p})}. \tag{4.14}$$

[1] See Section 4.6.1 for our parameter settings.

In Equations (4.12) and (4.13), we use $\sigma_d(\boldsymbol{p})$ to control the resulting smoothness. A smaller value makes the new depth gain less support from different depths, and a larger value produces more smooth depth map but also increases the possibility of spreading depth across the object edges. It is calculated by

$$\sigma_d(\boldsymbol{p}) = \gamma \cdot v(\boldsymbol{p}). \tag{4.15}$$

This definition takes the consistency of the estimate into account, enforcing more filtering on the depth with a higher variance, and γ controls the overall effect of the filtering.

The above filtering may lead to slight shifting for some inliers which were accurate before. To avoid this, after obtaining the new depth $\hat{d}(\boldsymbol{p})$, we perform three random shiftings (see (Bailer *et al.*, 2012)) around it to get more depths and calculate the matching error (Equation (2.5)) for each. If the smallest error

$$e_{\min} < \lambda, \tag{4.16}$$

λ being a threshold[1], we update the estimate $D_r(\boldsymbol{p})$ with the corresponding depth of $e_{\min}$ since the correlation measure is more robust in this case. Otherwise, the pixel $\boldsymbol{p}$ is likely located within a textureless area, so we depend more on the cross-view filtering and update $D_r(\boldsymbol{p}) \leftarrow \hat{d}(\boldsymbol{p})$.

Figure 4.4 presents our results on Dino dataset after cross-view filtering using different γ values with and without the random checking. Obviously, a smaller value makes less filtering effect, while a larger value tends to homogeneously smooth the depth data and leads to difficulty in finding back the details even if using additional random checking. When adopting the trade-off value of γ with the random checking, we get better alignment between depth discontinuities and object edges compared to the result without.

Depth-Consistency Constraints in Propagation

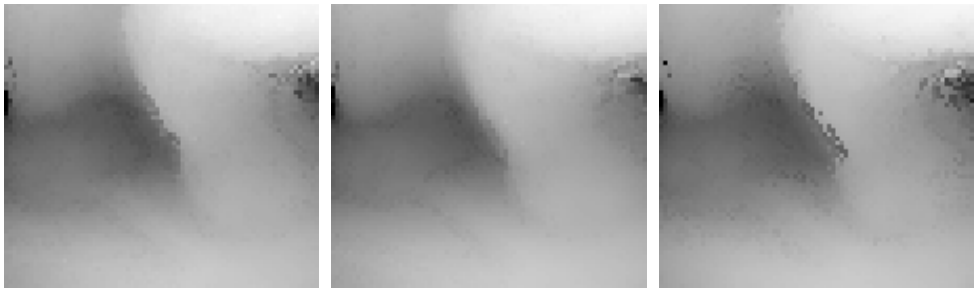
A naive depth propagation (Section 4.3) may decrease the strong depth consistency achieved by the previous cross-view filtering step, particularly when the photo-consistency constraint is unreliable. Therefore, in the depth propagation, for each consistent estimate with a depth variance lower than $1.5\tau_1$, we only update the data if matching error of the new surface patch is 1.2 times smaller than that of the former estimate.

4.6 Results

Four unorganized image sets have been tested to evaluate the performance of our approach (see Figure 4.5 and Table 4.1). Our procedures introduced in Sections 4.3, 4.4, and 4.5.3 were all executed on GPUs. We also implemented the GPU-based work of Bailer *et al.* (2012) (BAI) for comparison. With the estimated depth maps, we utilized



Image region | No cross-view filtering | $\gamma = 0.5$ with RC



$\gamma = 2$ with RC | $\gamma = 10$ with RC | $\gamma = 2$ without RC

Figure 4.4: The first two are an image region and its depth at scale 1 without cross-view filtering. The next three show the filtering results using $\gamma = 0.5$, 2, and 10, all with the subsequent random checking (RC). The last one is the result using $\gamma = 2$ but without RC.



Fountain-P11 | Dino (front) | Dino (back)



Temple (front) | Temple (back) | Sofa

Figure 4.5: Sample images of the photo collections used for evaluating our approach.

the Moving Least Square (MLS) algorithm (Cuccuru *et al.*, 2009) for selection and optimization of the produced 3D points, and the Delaunay triangulation (Labatut *et al.*, 2009) for final mesh-surface reconstruction.

4.6.1 Parameter Settings

The patch size for calculating matching error, the parameter γ for cross-view filtering (see Equation (4.15)), and the matching error threshold λ (see Equation (4.16)) are sensitive to the image size and content of the datasets. Their values for different datasets are list in Table 4.1. For the Temple scene with low image resolution, sufficient texture, and frequent occlusions, we adopted small patch size for reliable matching and small γ value to preserve the accurate estimates from patch matching. For Fountain-P11 as well as Sofa with high image resolution, and Dino which lacks enough textures, we utilized large patches for more information in patch matching and large γ values for less inconsistent outliers. We set $\lambda = 0.1$ for Dino to depend more on cross-view filtering instead of non-robust patch matching due to the poor textures, while larger λ values were used for others. The two thresholds τ_1 (see Equation (4.9)) and τ_2 (see Equation (4.11)) of the variance are independent of the datasets, so they were fixed in our experiments: $\tau_1 = 4 \times 10^{-6}$ and $\tau_2 = 36 \times 10^{-6}$.

4.6.2 Evaluation

Performance of our individual processing steps was evaluated on Fountain-P11, referring to Figure 4.6 for relative error maps of the center-view depth maps, Table 4.3 for a comparison of statistics, as well as Table 4.2 for the runtimes. The ground-truth depth maps are provided in (Tola *et al.*, 2010). We measure inaccuracy of each depth estimate d by computing its relative error using the corresponding ground-truth value d_{gt}:

$$e = \frac{|d - d_{gt}|}{d_{gt}}. \tag{4.17}$$

The completeness in the given results relates the number of recovered pixels to the image size. Figure 4.7 also compares mean consistency rating (see Equation (2.12)) and mean variance (see Equation (4.8)) of the depth maps which measure the multi-view depth coherence.

Local propagation (LP). Table 4.2 shows that local propagation with shorter scanlines is faster but might result in more holes if applied alone as shown in Figures 4.6 and 4.7, as well as Table 4.3. It has little effect on the accuracy though.

Hierarchical framework (HF). Table 4.2 also shows that our hierarchical estimation achieves a remarkable speed-up at coarse scales, and is faster than BAI and Only LP if a cross-view filtering and a second propagation are used (the time for scaling is almost

Name	Image number	Image resolution	Patch size	γ	λ
Fountain-P11 (Strecha *et al.*, 2008)	11	3072×2048	7×7	2.0	0.3
Dino (Seitz *et al.*, 2006)	363	640×480	7×7	2.0	0.1
Temple (Seitz *et al.*, 2006)	312	640×480	3×3	1.0	0.3
Sofa (Bailer *et al.*, 2012)	82	1853×1236	7×7	2.0	0.3

Table 4.1: Characteristics of the tested datasets and the corresponding parameter settings.

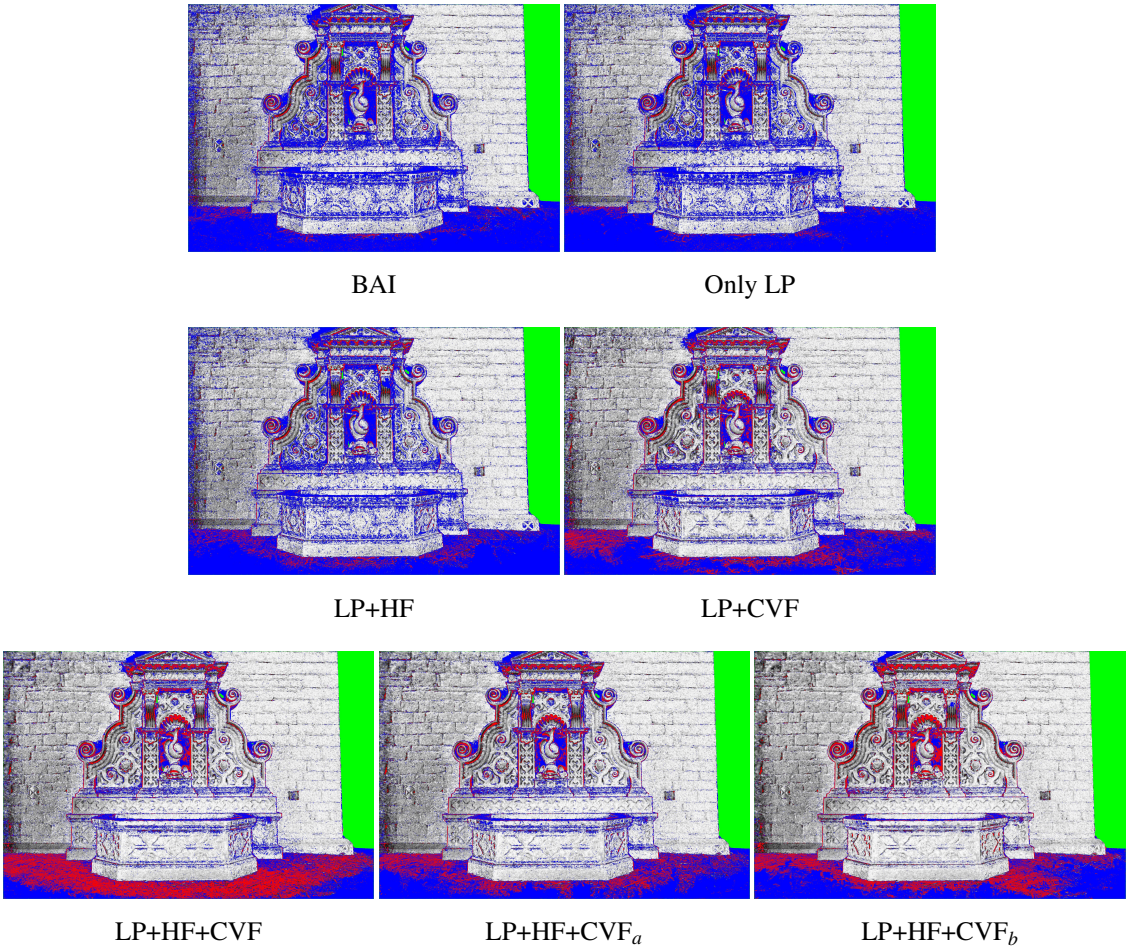


Figure 4.6: Relative error maps for the center view of Fountain-P11 after outlier removal, but not refined. LP+HF+CVF$_a$ uses cross-view filtering only for post-processing, and LP+HF+CVF$_b$ uses propagation-filtering at each scale without the second propagation step. The blue pixels have no estimate, the red have an error (see Equation (4.17)) larger than 0.003, the green have no ground truth data, and the pixels with an error between 0 and 0.003 are marked gray 255 to 0.

Step		BAI	Only LP	LP+HF	LP+CVF	LP+HF+CVF
Downscaling				8.4		8.4
1st	Propagation	174.3	142.2	13.6	142.0	13.6
	Cross-View Filtering				151.2	10.8
	Propagation				226.9	16.0
	Upscaling			0.4		0.4
2nd	Propagation	1126.6	880.3	224.5	1000.7	250.0
	Cross-View Filtering				193.3	49.2
	Propagation				951.9	228.7
	Upscaling			2.1		2.1
3rd	Propagation	418.0	410.2	417.4	450.6	458.2
	Cross-View Filtering				204.1	214.0
	Propagation				279.9	280.6
Outlier removal		42.2	41.2	44.2	48.1	51.1
Refinement			121.7	144.1	151.3	189.5
Overall		1984.4	1866.8	1079.1	4020.3	1844.1

Table 4.2: Runtimes (sec.) of each step using BAI and different combinations of our individual processing steps, when reconstructing all views of Fountain-P11. The results are not finally refined in BAI.

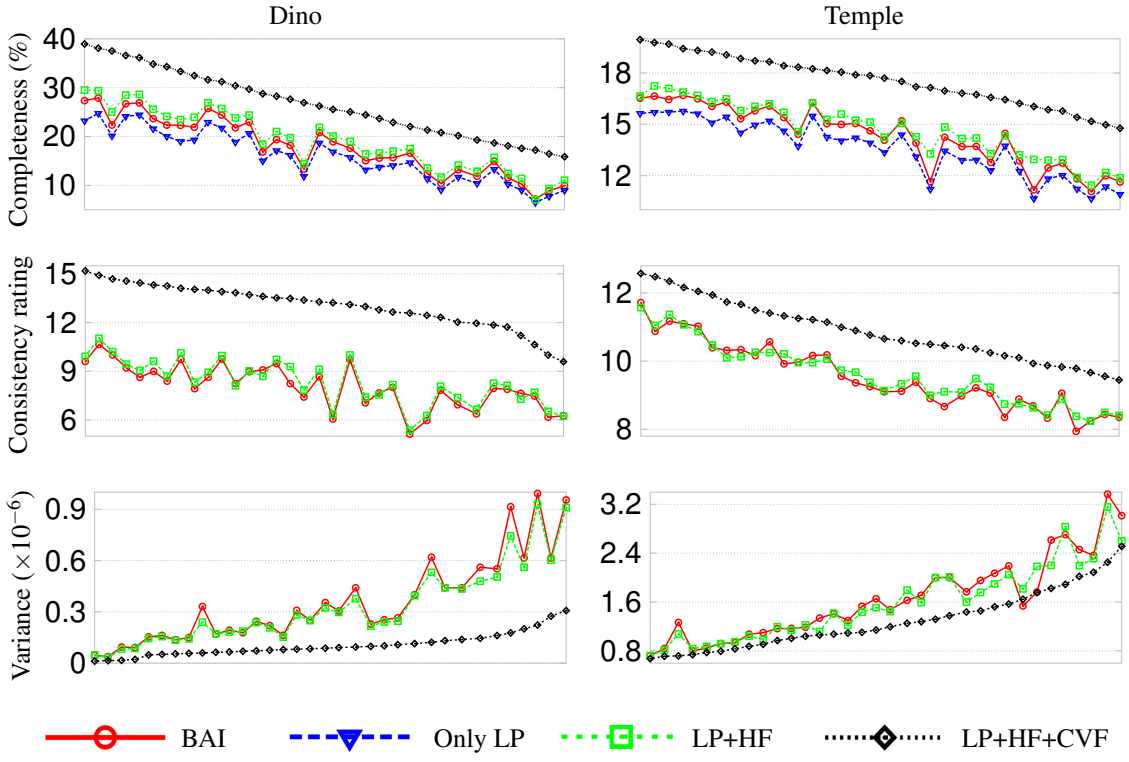


Figure 4.7: Comparisons of completeness, mean consistency rating, and mean variance for some sampled views of Dino and Temple. The views are sorted in preferred orders of LP+HF+CVF's results.

Measurement	BAI	Only LP	LP+HF	LP+CVF	LP+HF+CVF	LP+HF+CVF$_a$	LP+HF+CVF$_b$
Mean rel. error ($\times 10^{-3}$) $\downarrow$	1.663	1.414	1.236	2.407	1.732	1.505	2.062
Completeness (%) $\uparrow$	64.0	63.9	66.9	74.6	79.6	75.9	80.5
Mean consistency rating $\uparrow$	9.083	9.019	9.124	9.611	9.556	9.253	10.090
Mean variance ($\times 10^{-6}$) $\downarrow$	1.790	1.722	1.626	1.602	1.092	1.179	1.099
Mean rel. error of LP+HF+CVF on pixels of other methods ($\times 10^{-3}$) $\downarrow$	1.102	1.068	1.142	1.292		1.319	1.368

Table 4.3: Statistics obtained by the methods in Figure 4.6. The first row measures their relative errors on all reconstructed pixels, and the last shows the errors of our approach (LP+HF+CVF) only on the pixels reconstructed by other methods. The arrows indicate preferred directions. The proposed algorithm significantly improves all measures.

negligible). Furthermore, comparing Only LP and LP+HF in Table 4.3 the hierarchical strategy improves the depth accuracy and density. Though the holes are filled regardless of accuracy by our coarse-to-fine strategy, the subsequent random optimization and propagation steps improve the estimates. Its hole-filling effect is also shown in Figures 4.1 and 4.7.

Cross-view filtering (CVF). Comparing the first three columns, the column LP+CVF, and LP+HF+CVF in Table 4.3, the cross-view filtering improves reconstruction density and depth coherence in terms of both consistency rating and variance at the same time. This is underlined in Figures 4.6 and 4.7 where our algorithm shows better results for all tested views in Dino and Temple scenes.

We also evaluate the effectiveness of including cross-view filtering in the inner loop rather than using the consistency optimization as a post-processing filter at the finest scale (LP+HF+CVF$_a$ in Figure 4.6 and Table 4.3). Only used as a post-process, coverage, consistency and variance are reduced. Similar results would appear if one included similar weighting in the mesh fusion step. The benefit of the propagation step after the cross-view filtering becomes obvious in the last column (LP+HF+CVF$_b$) of Table 4.3 where the second propagation step is missing, resulting in higher coherence and higher density but lower accuracy.

As shown in the first row of Table 4.3, we obtain larger values of overall relative error using the methods with the cross-view filtering than those of the methods without. However, as presented in the last row of the table, significantly higher accuracies are achieved if only the pixels reconstructed by other methods are considered. This demonstrates that, our combined approach can find a desirable balance between depth accuracy and multi-view coherence by iterating between the correlation and consistency optimization, because high performance of either cue alone can not guarantee correct reconstruction of the 3D models.

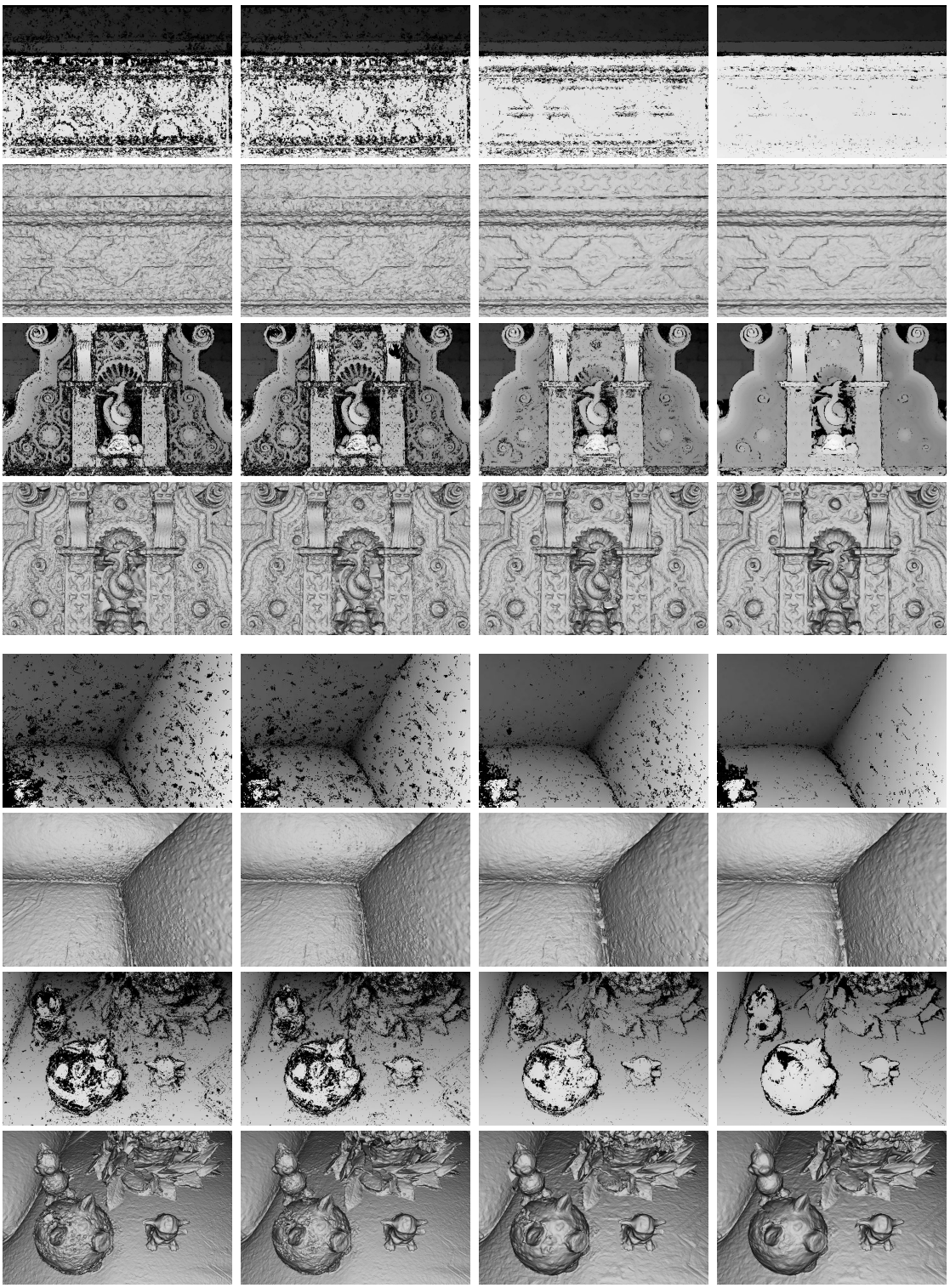
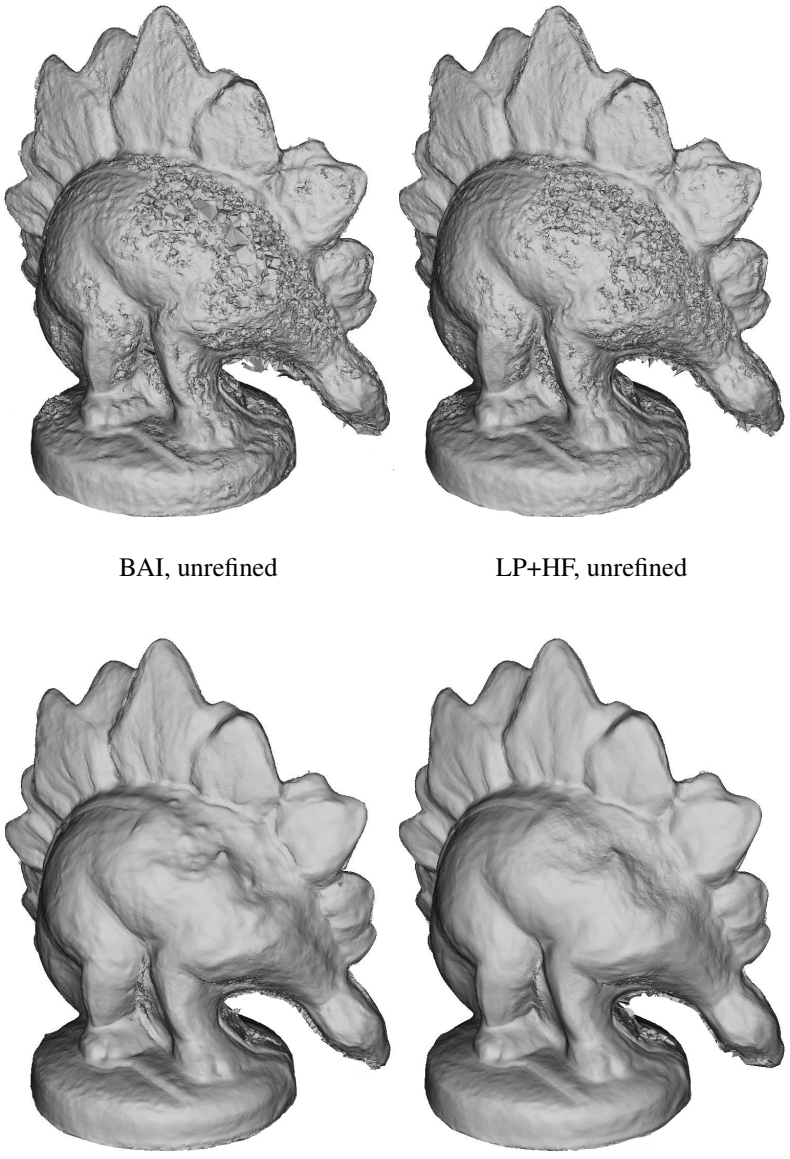


Figure 4.8: Outlier-removed depth map regions (odd rows) and reconstructed surfaces (even rows) of Fountain-P11 (top four rows) and Sofa (bottom four rows).



BAI, unrefined

LP+HF, unrefined

LP+HF+CVF, unrefined

LP+HF+CVF, refined

Sort By	Dino Full 363 views Acc [mm]	Comp [%]
Savinov	0.26	99.8
Schoenberger	0.26	97.8
Galliani	0.31	99.9
Furukawa 2	0.32	99.9
Furukawa 3	0.33	99.8
Schroers	0.33	99.7
Guillemaut	0.35	100
ICCV2015_1020	0.35	99.5
Semerjian	0.39	99.9
3DV2014_21	0.39	96.3
Yichao Li	0.4	94.9
Kuhn-GCPR-2013	0.41	95.3
Vogiatzis	0.42	99.0
Wei-BMVC-2014	0.42	98.1
Habbecke	0.43	99.7
Shiwei Li	0.44	99.8
Goesele 2007	0.46	96.7
Gervasoni	0.46	99.3
Lambert3	0.48	99.4
Hernandez	0.49	99.6
Langguth-SMVS	0.49	96.9
Furukawa	0.52	99.2
Zach2	0.55	98.7
Generalized-SSD	0.55	98.1
Goesele	0.56	80.0
Hongxing	0.62	96.3
ICCV2015_293	0.62	99.0
Lambert	0.76	99.0
Hornung	0.79	95.1
Jancosek-3DIM09	0.91	73.8

Middlebury benchmark

Figure 4.9: Dino models reconstructed by different methods and the evaluation of our full pipeline (LP+HF+CVF, refined) on the Middlebury benchmark (sorted by accuracy).



BAI, unrefined



LP+HF, unrefined



LP+HF+CVF, unrefined



LP+HF+CVF, refined

Sort By	Temple Full 312 views Acc [mm]	Comp [%]
Wei-BMVC-2014	0.34	99.4
Hernandez	0.36	99.7
Mücke	0.36	99.7
Schoenberger	0.37	98.9
Fuhrmann-SG14	0.39	99.4
Galliani	0.39	99.2
Campbell	0.41	99.9
Savinov	0.41	99.7
Shiwei Li	0.41	99.6
Goesele	0.42	98.0
Ummenhofer	0.42	99.3
Goesele 2007	0.42	98.2
Guillemaut	0.43	99.0
3DV2014_21	0.43	96.9
ICCV2015_1020	0.45	99.2
Yichao Li	0.46	96.4
Langguth-SMVS	0.47	98.7
Lambert3	0.48	99.7
Furukawa 3	0.49	99.6
Vogiatzis2	0.5	98.4
Zach2	0.51	98.8
Kuhn-GCPR-2013	0.52	96.2
ICCV2015_293	0.52	99.2
Generalized-SSD	0.53	99.4
Furukawa 2	0.54	99.3
Gervasoni	0.54	98.9
Lambert	0.55	99.7
Schroers	0.57	99.1
Hornung	0.58	98.7
Semerjian	0.62	97.8
Jancosek-3DIM09	0.65	85.8
Furukawa	0.65	98.7
Habbecke	0.66	98.0
Beljan	0.79	92.2
Hongxing	0.83	95.7
Vogiatzis	1.07	90.7

Middlebury benchmark

Figure 4.10: Temple models reconstructed by different methods and the evaluation of our full pipeline (LP+HF+CVF, refined) on the Middlebury benchmark (sorted by accuracy).

Depth map merging and 3D meshing. Figure 4.8 presents the normalized depth estimates and reconstructed geometries for sampled regions of Fountain-P11 and Sofa. Our combined approach yields higher quality of mesh models with smooth surfaces due to the denser and more consistent depth maps, and without loss of details. The figure also shows that the depth map refinement step further improves the results. We can get the same conclusion from Figures 4.9 and 4.10 where the reconstructed complete Dino and Temple models are shown.

To evaluate the benefit of our superior depth consistency in different depth map merging and meshing schemes, Figure 4.11 shows parts of the Dino and Temple surfaces (using our refined depth maps) reconstructed by: the MLS algorithm combined with Delaunay triangulation, MLS with subsequent Poisson surface reconstruction (PSR) (Kazhdan *et al.*, 2006), and the depth map fusion work of Fuhrmann and Goesele (2011). Our method with cross-view filtering produces better surfaces in all three cases on both datasets.

Comparisons to other MVS methods. We benchmark the reconstructions of our full pipeline (LP+HF+CVF) using the Middlebury evaluation website [2] which considers accuracy, completeness, and processing time. For Dino (see Figure 4.9, right), we achieved an accuracy of 0.42mm at 98.1% completeness, demonstrating that our work is competitive with other state-of-art methods, in particular, better than some region-growing-based (Goesele *et al.*, 2007; Jancosek *et al.*, 2009) and depth-map-fusion (Zach, 2008; Kuhn *et al.*, 2013) techniques. For Temple (see Figure 4.10, right), we achieved an accuracy of 0.34mm at 99.4% completeness. Our high accuracy is ranked *first* among all evaluated MVS algorithms. We reconstructed Dino in 63 mins and Temple in 27 mins, placing our work among the most efficient approaches.

4.7 Conclusion

In this chapter, we have proposed a hierarchical depth estimation algorithm using local propagation and cross-view filtering for 3D reconstruction from photo collections. The method is accelerated by parallel propagation along short scanlines. The coarse-to-fine estimation produces denser surfaces at reduced cost. The main focus is on optimizing the cross-view consistency of depth maps at all scales. Inconsistent estimates are removed and reliable, averaged candidates are propagated to neighboring views potentially helping in recovering the correct depth where otherwise would be a hole. The results show that all of our improvements lead to faster estimation, significantly denser and more consistent depth maps as well as more convincing 3D models.

[2] `http://vision.middlebury.edu/mview/`

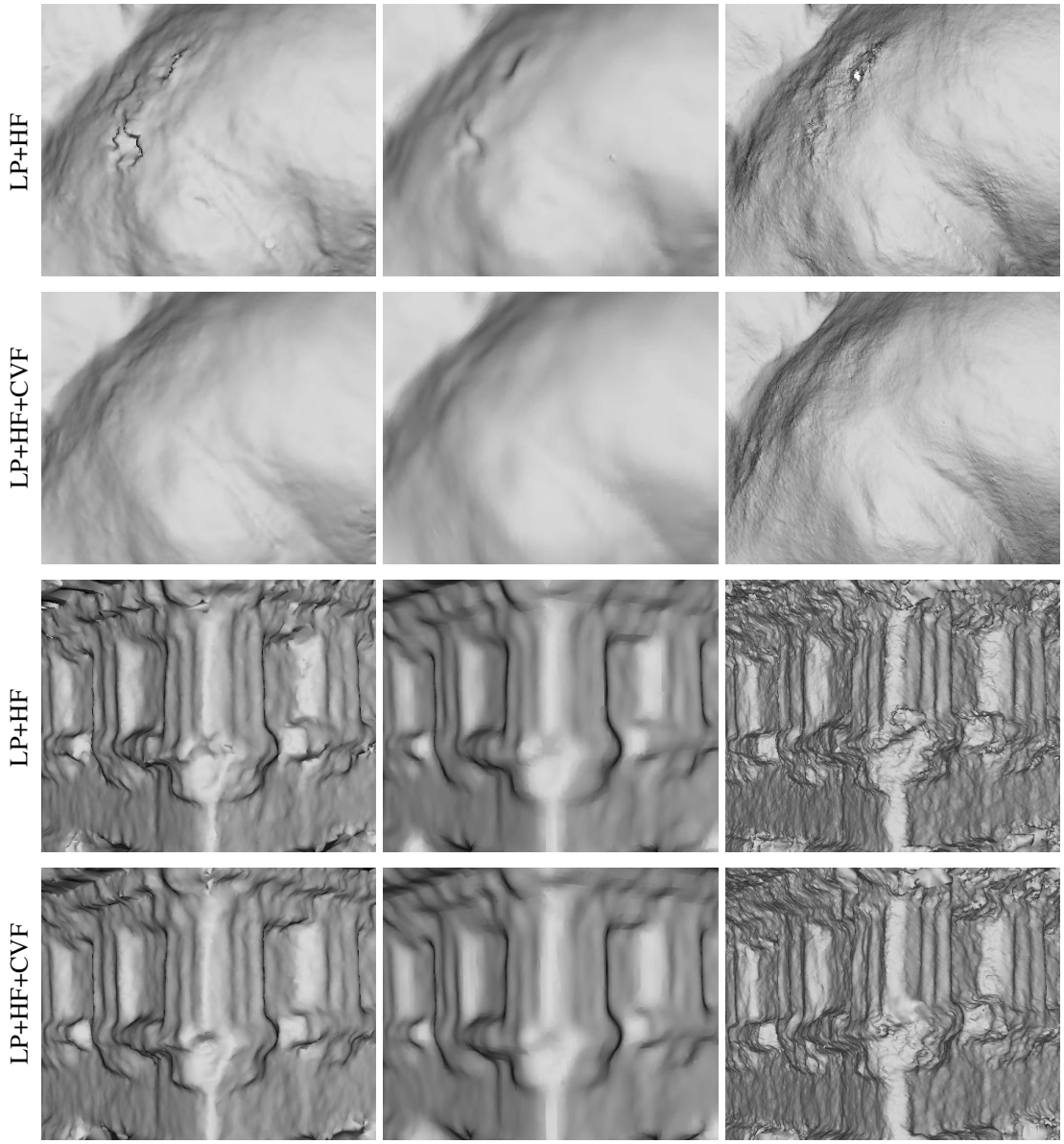


Figure 4.11: Recovered Dino (top two rows) and Temple (bottom two rows) surfaces using different merging and meshing methods on our refined depth maps without and with cross-view filtering.

Chapter 5

Dense and Occlusion-Robust MVS for Unstructured Videos

This chapter is devoted to generating a depth map sequence for the videos with arbitrary camera paths. Our main goal is to tackle the homogeneous areas where depth estimation subject to the photo-consistency constraint alone is difficult. We additionally impose a smoothness assumption and the visibility constraint for both dense and occlusion-robust depth maps.

5.1 Introduction

Video streams can be captured by mobile phones, hand-held video cameras, or flying drones. Dealing with this type of input data is more challenging than the photo collections because of the heavily increased temporal redundancy, and also than the light fields due to the unstructured camera trajectories. The light-filed-based approach (Kim *et al.*, 2013) efficiently exploits the extra scene information for ray-wise fine-to-coarse calculation, but still struggles on recovering large areas of homogeneous surfaces. And, the algorithm proposed in Chapter 4 alleviates the problem of poor texture in photos via more strict constraint on the cross-view consistency of depth estimates, but the issue is not completely solved, i.e., the homogeneous areas are not flat or incomplete.

This chapter borrows the edge-first framework from (Kim *et al.*, 2013), whereas we fit the *smoothest* possible surface for textureless areas, even if the actual geometry is curved. The reason for this smooth assumption is that planar geometry is very common in the real-world (see the examples in Figure 5.1). Note that correct recovery of shading is beyond the scope of this thesis. Unlike most of existing methods for dense MVS (see Section 3.1), our approach concentrates on image-space operations at pixel level, thus supporting high computational parallelism.

Our pipeline is separated into two parts: First, we compute depth for the reliably detectable regions by advancing (Kim *et al.*, 2013) towards higher depth resolution and more robustness on arbitrary camera movements. Next we diffuse edge depth to fill up homogeneous areas with perspectively correct interpolants, and afterwards handle



Figure 5.1: Planar surfaces are regularly seen in both indoor and outdoor scenes.

inconsistent surfaces. Section 5.2 gives an overview of the presented system, and the details will be described afterwards.

5.2 System Overview

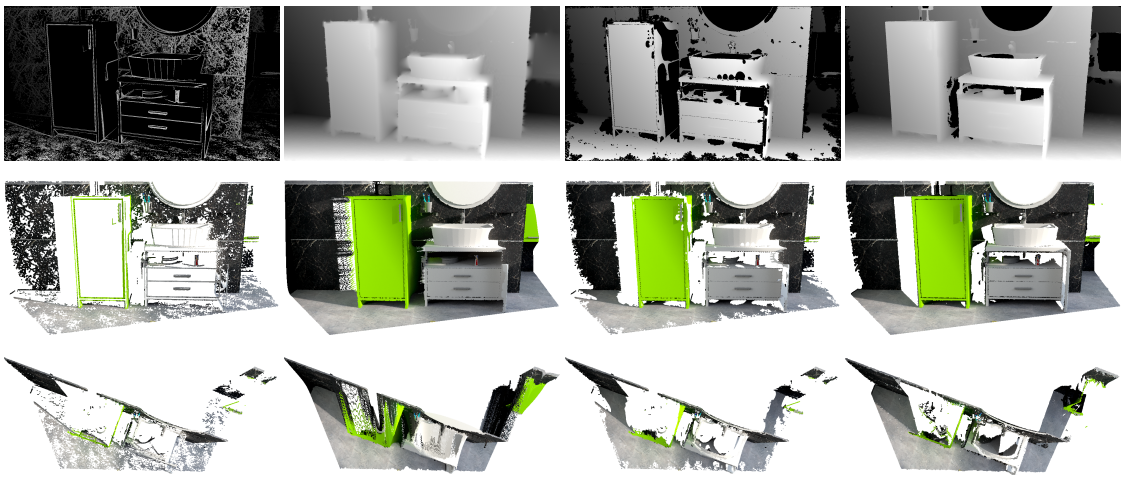
The pipeline of our system is presented in Figure 5.2 with the results of each step. The depth values at object edges are first calculated by considering individual visual rays to allow for sharp discontinuities, as in (Kim *et al.*, 2013), whereas the edge-depth accuracy is improved by advanced score evaluation and two-scale secondary-image selection. Then we perform perspective diffusion to create smooth surfaces between edges.

Depth diffusion per view probably produces interpolants that connect fore- and background edges but occlude the real surfaces. As the occluded surfaces can be seen by another view, we invalidate these incorrect values by comparing their projected depth and corresponding edge depth (if available) in other views. Since the edge depths are sparse, we need to spread the reliably detected invalid pixels to the whole wrong surfaces between discontinuous objects. We achieve this using a diffusion-based approximation of the depth uncertainty. The invalidation stage creates holes where edge depths have been wrongly interpolated. We produce the final depth maps with both high completeness and high consistency by propagating the closest valid depth over views.

Since our approach requires reliable edge depth to avoid removing too many of the surfaces estimated by diffusion, this chapter first focuses on improving (Kim *et al.*, 2013) in Section 5.3, and then describes the individual steps for recovering homogeneous areas in Section 5.4. Before these, Section 5.2.1 depicts how the input data is pre-processed in our reconstruction.

5.2.1 Pre-Processing

The SfM algorithm of Resch *et al.* (2015) is first executed to compute the camera extrinsics of all video frames, 3D positions of the feature points, and their associated visibility



Edge reconstruction | After diffusion | After invalidation | After propagation

Figure 5.2: Depth maps of one view (the top row) after individual steps of our reconstruction pipeline, and the corresponding point clouds viewed from the front (middle) and top (bottom) perspectives. The proposed approach successfully produces smooth and dense surfaces, with the wrong interpolants occluding the real surfaces removed.

information. Then we select a dense subset of images from the small-baseline frames for sufficient triangulation angles. Our approach estimates the depth maps at intervals of four image samples (see Section 5.3.3 for the reason). The selected images are pre-smoothed by a 7×7 Gaussian filer with standard deviation of 0.5 to remove noise and aliasing. Edge pixels are extracted from each image sample if the magnitude of its color gradient calculated using a 3×3 Sobel operator is larger than 2.5.

5.3 Edge Depth Estimation

Object edges are typically aligned with color discontinuities, and thus easy to detect and to reconstruct from images by imposing the photometric constraint. Therefore, we first reconstruct the high-contrast edges for individual images, producing a set of edge depth maps $\boldsymbol{D}^e$. For each reference image, the edge depth values are determined using 100 neighboring images [3] (50 preceding and 50 following, respectively). A set of corresponding uncertainty maps $\boldsymbol{U}^e$ for the edge depth is also calculated, which will be used in our visibility conflict invalidation (see Section 5.4.2).

Section 5.3.1 first investigates the influence of unconstrained camera paths upon depth estimation using pixel-dependent color and score maps. We accordingly introduce a careful score aggregation method (Section 5.3.2) and a two-scale secondary-image selection

[3] See Section 5.5.1 for our parameter settings.

scheme (Section 5.3.3) for camera-trajectory robustness. Section 5.3.4 also presents how we achieve subpixel precision and remove outliers. The uncertainty of each edge depth is calculated in Section 5.3.5.

5.3.1 Pixel-Wise Color and Score Maps

To attain precise depth estimates, we have to carefully deal with occlusion and out-of-image projection particularly for arbitrary camera trajectories. Our edge depth calculation is based on an analysis of color and score maps for each pixel. These maps are built in depth-view space (see Figure 5.3), that enables us to easily visualize the appearance of the projected point in each secondary image with a certain depth. They help us to define an aggregated depth score that distinguishes the correct depth more robustly (see the score profiles in Figure 5.4).

Let's consider the ith image I_i as the reference image. We calculate scores for 1024 depth hypotheses[3] by selecting its 100 neighborhoods $\{I_{i-50},\dots,I_{i+50}\}\setminus\{I_i\}$ as the secondary images. Considering a pixel $\boldsymbol{p}$ in I_i, let $\boldsymbol{q}$ be its projection in the jth secondary image S_j using the kth hypothetical depth d_k. To ensure that all projections have constant density along the epipolar line, the hypotheses are distributed evenly in inverse depth along the visual ray. Using each pair of (d_k, S_j) on $\boldsymbol{p}$, we build a 1024×100 size *color map* with the projection color $S_j(\boldsymbol{q})$ in *depth-view space*, i.e., the vertical axis represents depth values and the horizontal axis secondary views, as well as a corresponding *score map* with the depth score calculated using the Gaussian kernel[3]:

$$s(\boldsymbol{p}, d_k, S_j) = \exp\left(-\frac{1}{2{\sigma_c}^2}|I_i(\boldsymbol{p}) - S_j(\boldsymbol{q})|^2\right). \tag{5.1}$$

Figure 5.3 shows the generated maps for four representative pixels. Overall, there should exist a cross shape in each map: Significantly more same colors and score peaks are distributed around the center view and a certain hypothesis which is typically the real depth. The clarity of this property depends on the image contrast, the secondary-image selection scheme, and the visibilities in secondary views. Pixel $\boldsymbol{p}_1$ gains a wide score peak distribution due to weak color contrast. Higher contrast (e.g., $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$) or sparser image selection (i.e., the image set covers a wider range of viewing directions) results in a more recognizable distribution. The cross shape is interrupted in view direction where occlusion ($\boldsymbol{p}_3$) or out-of-image projection ($\boldsymbol{p}_4$) occurs. These desirable features are well-incorporated in our edge depth calculation.

5.3.2 Robust Score Aggregation

Edge depth is determined by aggregating for a certain hypothesis the per-secondary-image depth scores (i.e., the scores in one row of the proposed score map) and finding the depth receiving the highest sum. As Figure 5.4 shows, simple averaging (Equation (2.7))

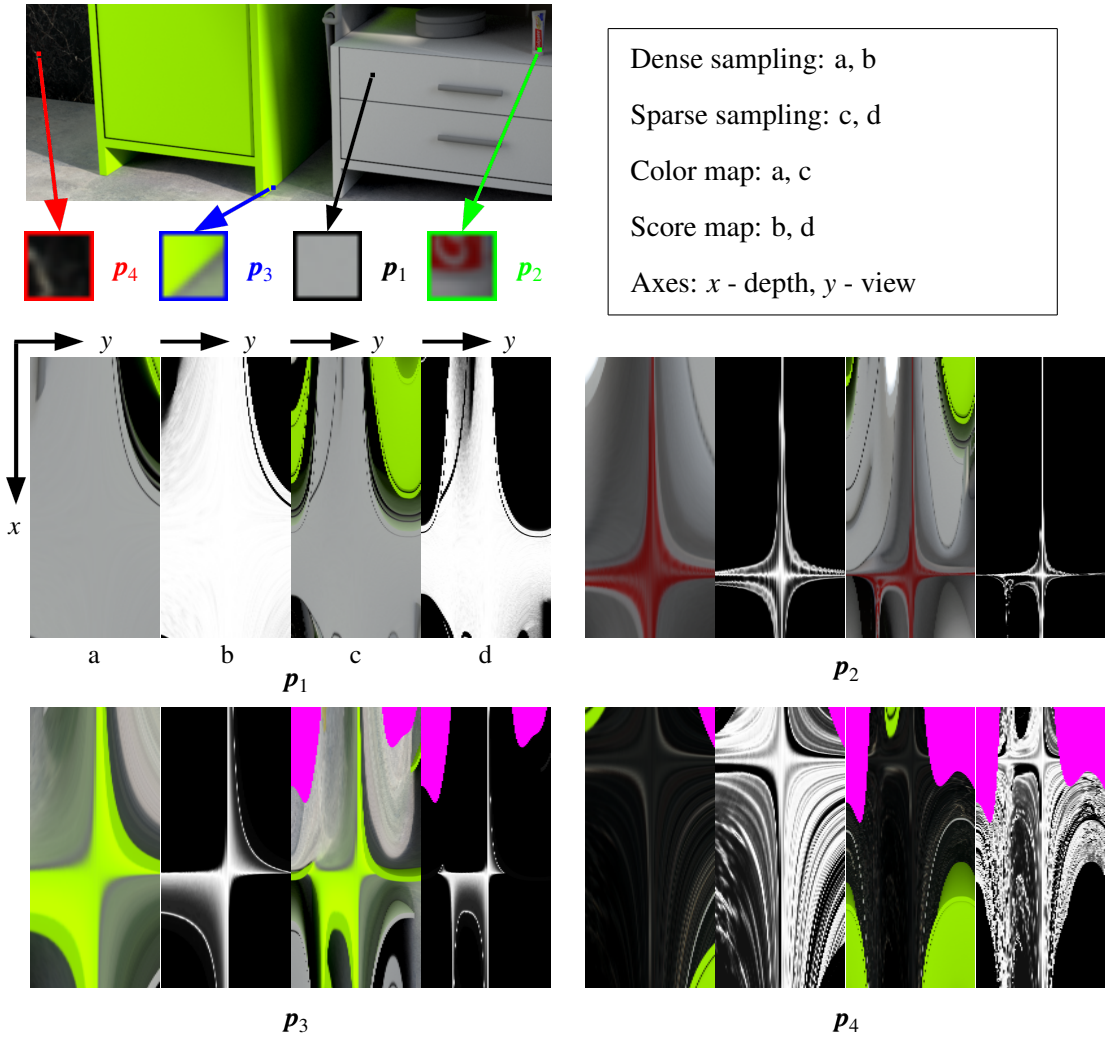


Figure 5.3: Depth-view-space analysis. For each marked pixel (top left, the 15×15 pixel patch is enlarged), we present its color and score maps once with densely sampled and once with sparsely distributed secondary images (see Section 5.3.3 for the secondary-image selection). In cases of out-of-image projection, the corresponding areas are marked magenta (see the maps of $\boldsymbol{p}_3$ and $\boldsymbol{p}_4$). The maps are widened for better visualization.

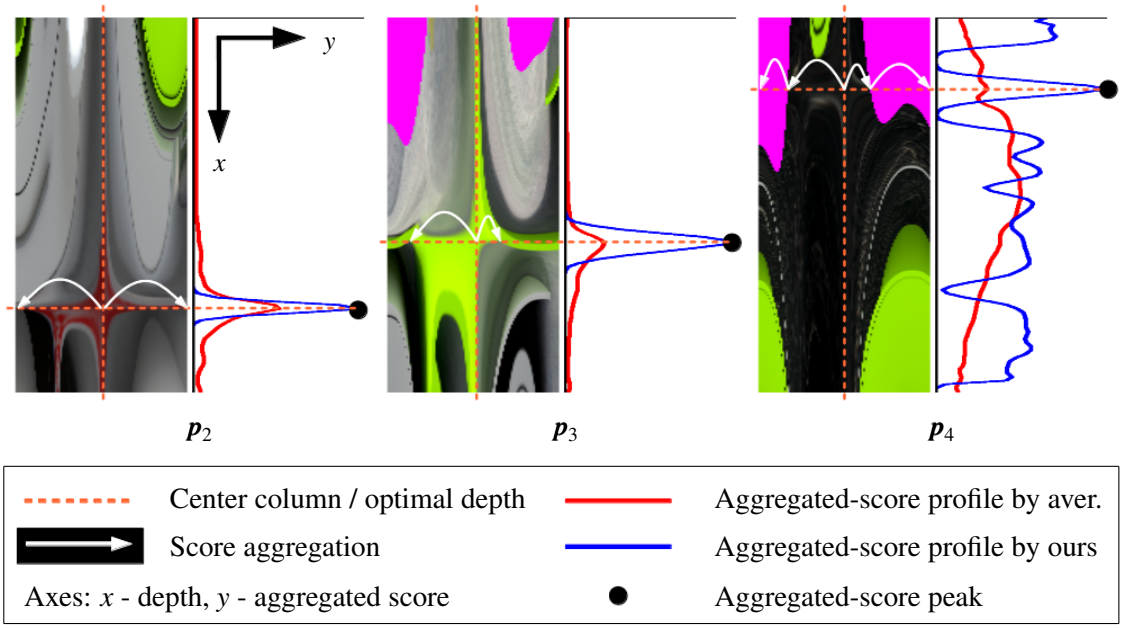


Figure 5.4: Sparse-scale color maps of the three edge pixels in Figure 5.3 and the corresponding profiles of per-depth aggregated scores. We robustly add up the corresponding depth scores in different cases (see Section 5.3.2). Thus the obtained score profiles can detect the optimal depth more reliably than simple averaging (Equation (2.7)).

leads to score profiles with ambiguous peaks for the pixels that are occluded (e.g., $\boldsymbol{p}_3$) or out-of-image ($\boldsymbol{p}_4$) in some views. Kim *et al.* (2013) handle this problem with mean-shift iterations (Equation (2.8)) but at the expense of computation time.

We define a robust and faster scheme by more carefully adding up the scores while moving from the nearest to outer views (preceding and following, respectively). This way, we can more reliably distinguish different cases (e.g., occlusion, out-of-boundary projection, or the same surface point) and accordingly treat the remaining projections. This aggregation process equates to accumulating the scores from the center to left and right within a line of the pixel-wise score map.

Specifically, let the 100 depth scores of pixel $\boldsymbol{p}$ for hypothesis d_k be aggregated to $s^{\Sigma} \leftarrow 0$, $n_m \leftarrow 0$ be the number of measurable images, and $n_h \leftarrow 0$ the number of out-of-image projections. When we *sequentially* test d_k in the preceding secondary images $\{I_{i-1}, \ldots, I_{i-50}\}$, three cases are motivated in the jth image S_j:

1. If the projection moves out of the image (eg, $\boldsymbol{p}_4$ in Figure 5.4), we hallucinate the minimum score $s_{\min}$ found so far, which is initialized with $s_{\min} \leftarrow 1$, for the remainder by

$$s^{\Sigma} = s^{\Sigma} + (50 - j)s_{\min} \quad \text{and} \quad n_h = n_h + (50 - j). \tag{5.2}$$

This produces a lower score average than as if the projections were available but still high enough to be reasonable for the same surface point.

2. If the projection is available and the depth score $s := s(\boldsymbol{p}, d_k, S_j)$ is acceptable[3] (see $\boldsymbol{p}_2$ in Figure 5.4), we update $s_{\min}$ by

$$s_{\min} \leftarrow \min\left((1+\alpha)s_{\min}, s\right) \quad \text{with} \quad \alpha < 1. \tag{5.3}$$

We use $(1+\alpha)s_{\min}$ to approximate a relatively bad match by allowing a score decrease of $\alpha s_{\min}$ along a noisy score profile[3]. Then the score is aggregated by

$$s^{\Sigma} = s^{\Sigma} + s_{\min} \quad \text{and} \quad n_m = n_m + 1. \tag{5.4}$$

We add up $s_{\min}$ instead of s to avoid adding higher scores for the remaining images if the actual score is low. Because this low score might be introduced by a depth discontinuity and the remaining projections might be on a foreground surface with the same color.

3. If the depth score is unacceptable, it represents a discontinuity (see $\boldsymbol{p}_3$ in Figure 5.4). We stop aggregating because the surfaces of the remaining projections can be arbitrary.

The aggregation is then performed on the following secondary images $\{I_{i+1}, \ldots, I_{i+50}\}$ in the same way. We let $s^{\Sigma} = 0$ if $n_m \leq 2$, and normalize it otherwise by

$$s^{\Sigma} = \frac{f s^{\Sigma}}{n_m + n_h}, \tag{5.5}$$

where $f = \sqrt{n_m}$ is a correction factor to prefer the hypotheses generating more promising matches. Figure 5.4 shows that our score aggregation creates more distinctive score profiles for pixels $\boldsymbol{p}_2$ as well as $\boldsymbol{p}_3$, and even more correct for $\boldsymbol{p}_4$ than averaging.

5.3.3 Two-Scale Secondary-Image Selection

A dense set of neighboring images has less chances of a surface point being occluded or projected outside the view (e.g., $\boldsymbol{p}_3$ and $\boldsymbol{p}_4$ in Figure 5.3) but might provide narrow baselines leading to imprecise depth estimates. We increase triangulation angles by introducing another secondary-image set that selects the frames more sparsely and more widely spread. Hereby we can refine the depth range at the dense scale and determine depth more reliably at the sparse scale.

To this end, as mentioned in Section 5.2.1, we sample the video frames into a dense image subset $\boldsymbol{I}^d$ and then further sample $\boldsymbol{I}^d$ into a sparse subset $\boldsymbol{I}^s$ with a four times wider spacing, i.e., $I_i^s \in \boldsymbol{I}^s$ corresponds to $I_{4i}^d \in \boldsymbol{I}^d$. Our depth map estimation is performed on $\boldsymbol{I}^s$. As illustrated in Figure 5.5, our goal is to guarantee that the sparse set of secondary

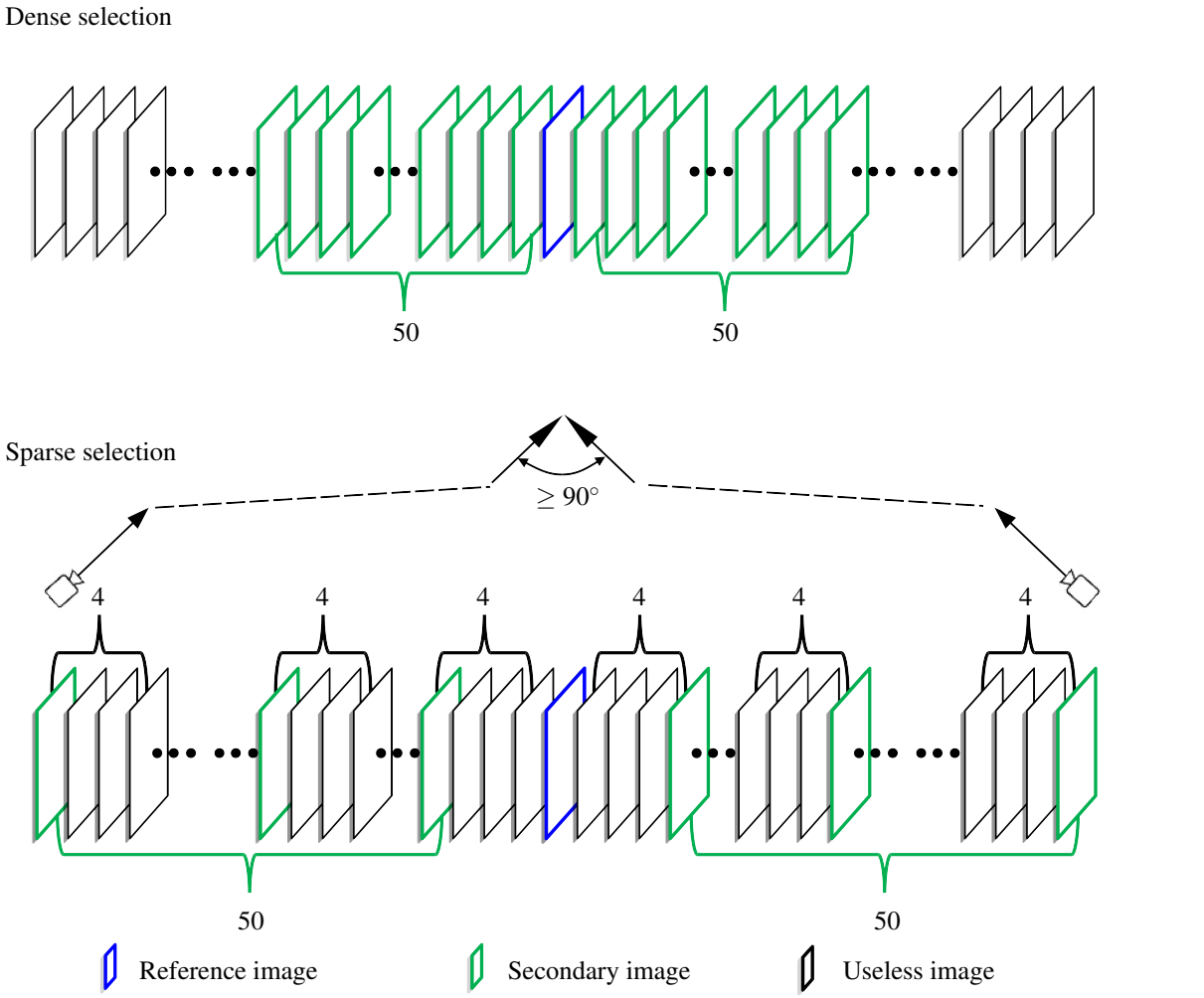


Figure 5.5: Two-scale secondary-image selection. We select images from the input video frames so that, for each reference image, its 4×100 neighborhoods have a viewing angle no smaller than 90°. Among these neighborhoods, we use 100 nearest images as the densely sampled secondary images (top), and select each secondary image in every four for sparser sampling (bottom).

images for each reference image features approximately the same change in view direction of about 90° in total. This means, each pair of neighboring dense image samples should have a viewing angle no smaller than $\frac{90^\circ}{4\times100}$. This is achieved using the visibility constraints of the SfM points from (Resch *et al.*, 2015).

In the following, we introduce how we obtain subpixel depth precision via iterative depth range refinement on both image subsets.

5.3.4 Subpixel-Level Depth Sweeping

We get subpixel precision by iteratively testing 1024 depth hypotheses within the gradually refined depth range based on the score profile (visualized in Figure 5.4), until convergence or bad matching is reached. This process is done per pixel.

For a reference-image edge pixel, we initialize its depth range using the depths of the nearest and farthest visible SfM points in that view. We first refine the depth range *in one pass* by referring to the *dense* secondary images and then *iteratively* refine it by using the *sparse* secondary-image samples. Particularly, in the tth iteration, we first determine the optimal depth $\hat{d}$ within the current depth range. If $\hat{d}$ generates only a few measurable matches, i.e., $n_m < 10$ in Equation (5.5), we stop sweeping and leave the pixel assigned no depth. Otherwise, let the corresponding aggregated score of $\hat{d}$ be s_t^Σ. If $|s_t^\Sigma - s_{t-1}^\Sigma| < 0.001$, the sweeping is converged and this pixel is assigned $\hat{d}$. Otherwise, we update the depth range with the minimum and maximum of all the hypotheses whose aggregated score is high enough, i.e., higher than $0.9s_t^\Sigma$.

Edge Depth Post-Processing

The pixel-level operation probably results in outliers due to insufficient contrast, aliasing artifact, or reflection. To detect the unreliable depth, we shift the pixel projections in all sparse-scale secondary images along the epipolar lines by one pixel and calculate a new aggregated score. If this value is higher or close to the score of the depth estimate, the optimal depth is ambiguous and thus removed. We use 0.02 as the threshold. We also eliminate the isolated points with large 3D distance from neighborhoods and then perform a 7×7 median filter for smoothness.

5.3.5 Depth Uncertainty

We calculate the standard deviation for each depth estimate to measure its uncertainty. The uncertainty values of edge depth will be reused in Section 5.4.2 for adjusting tolerances when depth inconsistencies are searched in homogeneous areas. Moreover, in Section 6.6 we also respect the uncertainties for point merging.

The score profile of a confident depth should have a sharp peak (e.g., pixel $\boldsymbol{p}_2$ in

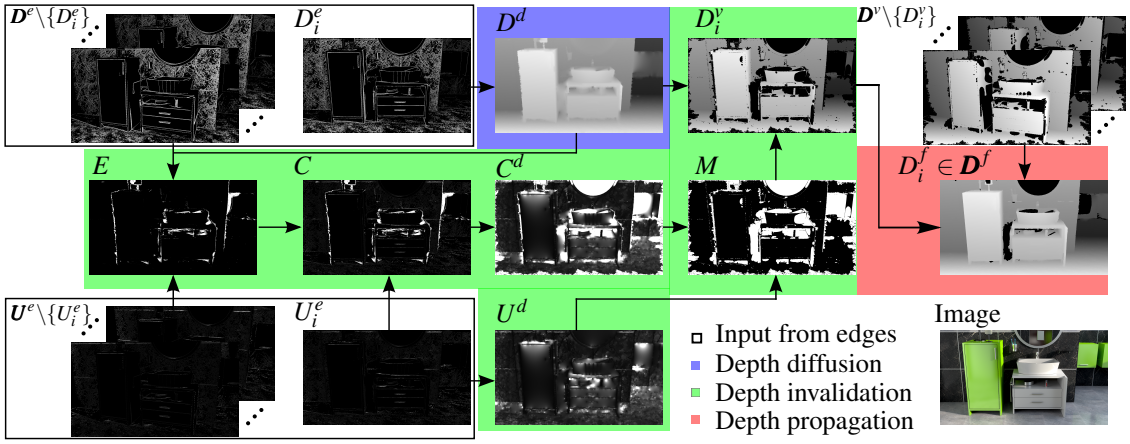


Figure 5.6: Homogeneous-area depth estimation for the ith image (see bottom right). With all edge depth maps $\boldsymbol{D}^e$ and the corresponding edge uncertainty maps $\boldsymbol{U}^e$, we produce the dense and consistent depth map $D_i^f \in \boldsymbol{D}^f$ via three stages: diffusion, invalidation, and propagation. See Section 5.4 for details. The values in the uncertainty and error maps (E, C, and C^d) have the same scale. To visualize the errors, the uncertainty maps are very dark.

Figure 5.4). Thus, we define the uncertainty of a depth estimate d as

$$u = \sqrt{\frac{\sum_k (d - d_k)^2 s_k^\Sigma}{1024}}. \tag{5.6}$$

s_k^Σ is the aggregated score of the kth depth d_k after the first pass of depth range refinement. After calculating the uncertainties for edge depth, each uncertainty map is also smoothed by a 7×7 median filter.

5.4 Homogeneous-Area Depth Estimation

In many scenes, object edges already capture the main characteristic structures of the scene. Thus we propose to handle remaining homogeneous areas by diffusing those sparse depth estimates, assuming flat surfaces in between. In addition to completeness, we also favor visibility consistency of the interpolated surfaces.

For this purpose we adopt the workflow depicted in Figure 5.6, that begins from Section 5.4.1 by diffusing each edge depth map $D_i^e \in \boldsymbol{D}^e$ to a complete depth map D^d. In Section 5.4.2 the wrong interpolants occluding the real surfaces are invalidated from D^d to create a corresponding valid depth map $D_i^v \in \boldsymbol{D}^v$. In Section 5.4.3, we improve the completeness and consistency of the final depth map $D_i^f \in \boldsymbol{D}^f$ by propagating the frontmost valid depth values across view, i.e., over $\boldsymbol{D}^v$.

5.4.1 Depth Diffusion

This section first investigates the problem of classic diffusion scheme on depth data (Stefanoski *et al.*, 2013), and next proposes a modified method to correct the distorted surfaces perspectively.

Classic Diffusion

The classic approach performs iterative convolution with a 4-point stencil. Let the diffused result of edge depth map D_i^e in the tth iteration be D_t^d, and $D_0^d \leftarrow D_i^e$ for initialization. For a pixel $\boldsymbol{p} = (x, y)$, if $\boldsymbol{p}$ has no edge depth, the diffusion problem is solved using the Jacobi iterations

$$D_{t+1}^d(\boldsymbol{p}) = \frac{1}{\delta^\Sigma} \sum_{\boldsymbol{p}'} \delta\left(D_t^d(\boldsymbol{p}')\right) D_t^d(\boldsymbol{p}') \tag{5.7}$$

with

$$\boldsymbol{p}' \in \{(x-w, 0), (x+w, 0), (0, y-w), (0, y+w)\} \tag{5.8}$$

and

$$\delta^\Sigma = \sum_{\boldsymbol{p}'} \delta\left(D_t^d(\boldsymbol{p}')\right) \tag{5.9}$$

for normalization, and $D_{t+1}^d(\boldsymbol{p}) = D_t^d(\boldsymbol{p})$ otherwise to preserve the depth discontinuity. w is the stencil size. The indicator function $\delta(\cdot)$ is evaluated to one for a known depth and zero for others. If no available depth is found for diffusion, i.e., $\delta^\Sigma = 0$, the result at $\boldsymbol{p}$ remains unchanged.

Figure 5.7 illustrates the horizontal diffusion for $\boldsymbol{p}$ from two pixels $\boldsymbol{p}_1 = (x-w, y)$ and $\boldsymbol{p}_2 = (x+w, y)$ with depth values d_1 and d_2, respectively. By using the isotropic diffusion in Equation (5.7), $\boldsymbol{p}$ would be assigned the average depth $d_w = \frac{1}{2}(d_1 + d_2)$. However, due to the perspective distortion the corresponding 3D interpolant $\boldsymbol{P}_w$ lies behind the point $\boldsymbol{P}_r$, which is on a flat surface connecting the 3D points of $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, i.e., $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$. This is because the 2D pixel grid is not exactly projected to an uniform grid on the slanted surface. A solution to this problem is adopting an anisotropic strategy.

Perspective Diffusion

Instead of finding a weighting function, we employ a simpler method by extending Equation (5.7) to the visual-ray space. We note in Figure 5.7 that, since $\boldsymbol{P}_r$ is the intersection between the visual ray $\overrightarrow{\boldsymbol{OP}_r}$ and the 3D line $\overline{\boldsymbol{P}_1\boldsymbol{P}_2}$, both can be calculated, $\boldsymbol{p}$ can get its depth d_r by back-projecting $\boldsymbol{P}_r$ onto the image plane.

Therefore, we first formulate this perspective strategy to allow for that the initial depths are scattered. Then we incorporate the interpolants respectively obtained in horizontal and vertical directions into the 2D diffusion task.

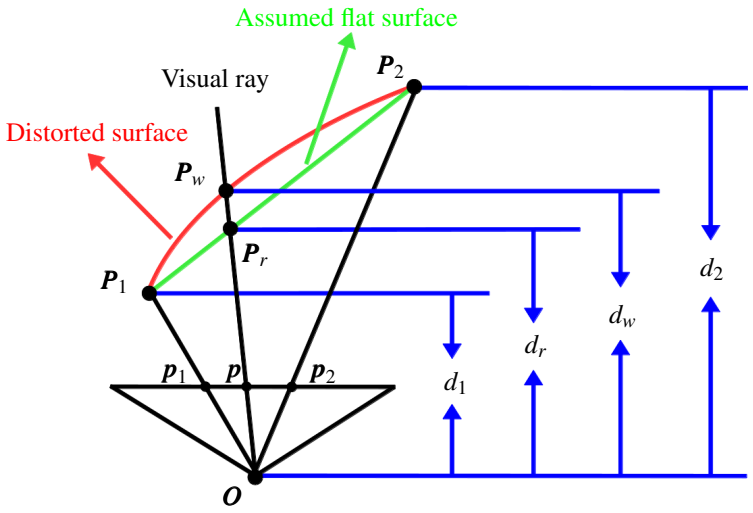


Figure 5.7: Depth diffusion from two pixels $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ to the mid-pixel $\boldsymbol{p}$. Point $\boldsymbol{P}_w$ is the projected interpolant using the classic isotropic method (Equation (5.7)), which defines the diffused depth as $d_w = \frac{1}{2}(d_1 + d_2)$. $\boldsymbol{P}_r$ is the assumed real point on the principle surface (green), and d_r is the corresponding real depth. $\boldsymbol{O}$ represents the camera center.

In order to assign $\boldsymbol{p}$ the only available depth if one of $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ has no estimate, we include the indicator function $\delta(\cdot)$ to denote the horizontally diffused depth as

$$d_x = \delta_1\delta_2 d_r + (1-\delta_1\delta_2)\left(\delta_1 D_t^d(\boldsymbol{p}_1) + \delta_2 D_t^d(\boldsymbol{p}_2)\right), \tag{5.10}$$

where $\delta_1 := \delta\left(D_t^d(\boldsymbol{p}_1)\right)$ and $\delta_2 := \delta\left(D_t^d(\boldsymbol{p}_2)\right)$.

The vertically diffused value d_y is calculated similarly by using $\boldsymbol{p}_1 = (x, y-w)$ and $\boldsymbol{p}_2 = (x, y+w)$. We incorporate these two depths for each non-edge pixel $\boldsymbol{p}$ using

$$D_{t+1}^d(\boldsymbol{p}) = \frac{\delta(d_x)d_x + \delta(d_y)d_y}{\delta^\Sigma} \tag{5.11}$$

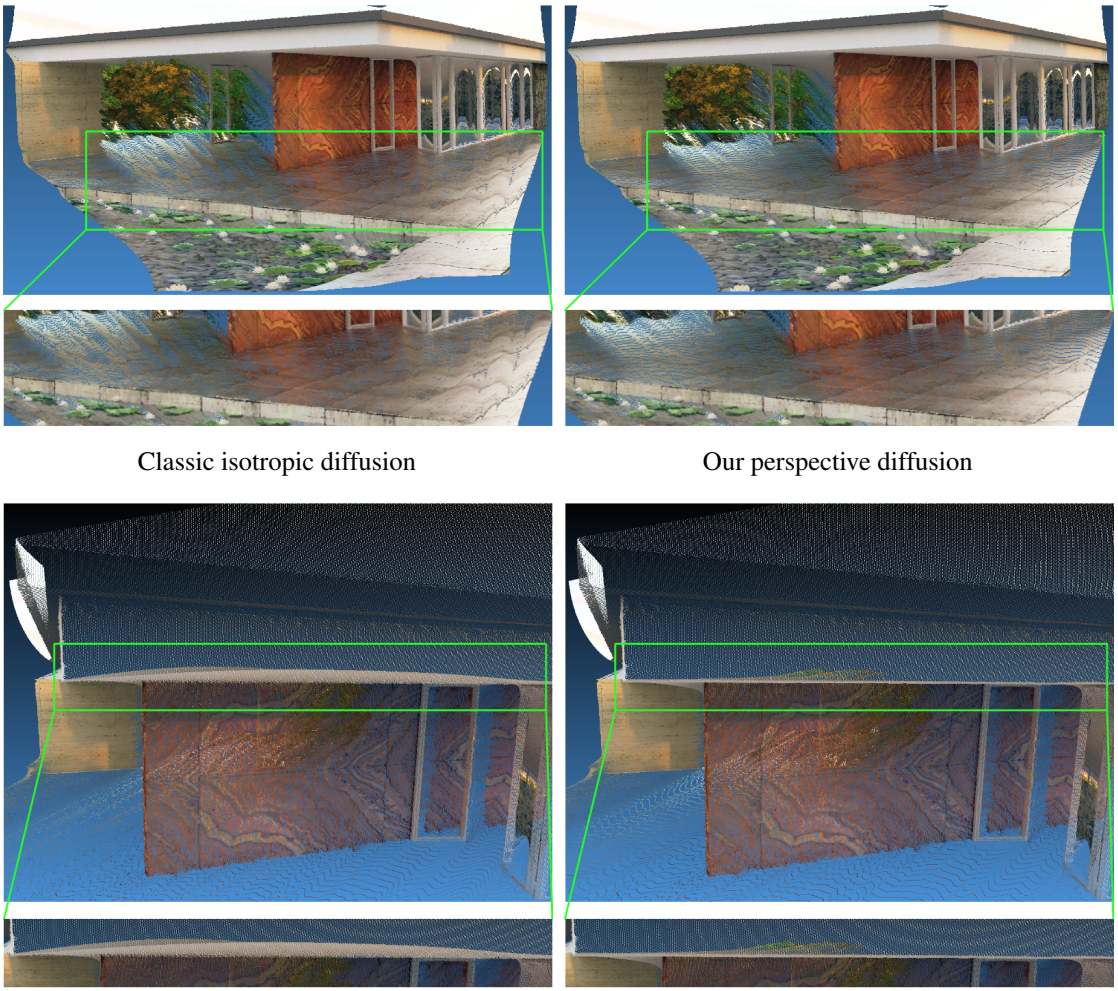
with

$$\delta^\Sigma = \delta(d_x) + \delta(d_y) \tag{5.12}$$

for normalization.

Thereby, a small stencil size w might slow down the depth transportation. We use the stencil-shrinking solver (Jeschke *et al.*, 2009) by initializing w for each interpolated pixel with the Euclidean distance from the nearest edge pixel.

Figure 5.8 compares the resulted 3D points using classic and our approaches. The distortion from isotropic depth diffusion produces uneven surfaces on the floor and an oscillation on the large-area homogeneous ceiling. In contrast, the perspective method generates flat surfaces in both cases.



Classic isotropic diffusion | Our perspective diffusion

Classic isotropic diffusion | Our perspective diffusion

Figure 5.8: Points reconstructed by classic and our diffusion strategies. The floor and ceiling are enlarged below for more clear comparison. The isotropic method creates distorted surfaces while the perspective interpolation produces smooth and flat results.

5.4.2 Visibility Conflict Invalidation

Depth discontinuities with homogeneous areas in the background exhibit the occlusion problem (see Section 2.4.2). As illustrated in Figures 5.2 and 5.9. Filling unreconstructed areas by diffusing the known edge depth (Section 5.4.1) leads to interpolating between fore- and background edges. This introduces incorrect interpolants which would occlude the real surfaces. We solve these visibility conflicts in a sparse-to-dense manner (see Figures 5.6 and 5.9). In each complete depth map, we first distinguish a set of definitely invalid depth values, and then make these sparse invalid pixels grow to larger areas.

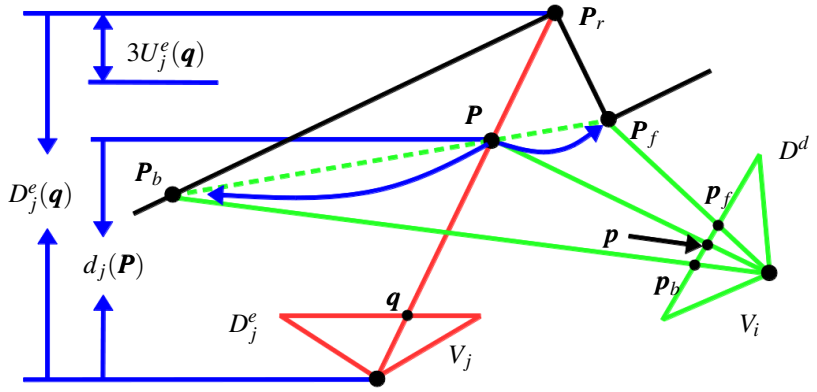


Figure 5.9: Occlusion problem and our solution. For the view V_i, diffusing the depth estimates of the foreground edge $\boldsymbol{P}_f$ and the background texture edge $\boldsymbol{P}_b$ produces the interpolants (dashed line) occluding the real surfaces. We use the detected edge $\boldsymbol{P}_r$ at pixel $\boldsymbol{q}$ in another view V_j to invalidate the interpolant $\boldsymbol{P}$ and then grow the invalid area towards $\boldsymbol{P}_b$ and $\boldsymbol{P}_f$.

Local Depth Invalidation

We find invalid interpolants in the diffused depth map D^d by checking for consistence with the edge depth values calculated in other views. This process generates a sparse error map E.

In particular, assume that D^d is created from the view V_i as shown in Figure 5.9. Let's consider an interpolated pixel $\boldsymbol{p}$ in D^d, whose point estimate is $\boldsymbol{P}$. We back-project $\boldsymbol{P}$ to the edge depth map D_j^e of another view V_j. Let the pixel projection be $\boldsymbol{q}$ and the projected depth be $d_j(\boldsymbol{P})$. If the depth estimate of $\boldsymbol{q}$ is available, it means that an edge ($\boldsymbol{P}_r$ in Figure 5.9) is reconstructed at $\boldsymbol{q}$. Since it is impossible to see an edge behind a surface in the same view, $\boldsymbol{P}$ is theoretically invalid in case $d_j(\boldsymbol{P}) < D_j^e(\boldsymbol{q})$. For more robustness to the imprecision of $D_j^e(\boldsymbol{q})$, we respect the depth uncertainty of $\boldsymbol{q}$ in the corresponding edge uncertainty map $U_j^e \in \boldsymbol{U}^e$ (see Figure 5.6) by comparing with a shortened depth (see Figure 5.9):

$$d_j(\boldsymbol{P}) < D_j^e(\boldsymbol{q}) - \beta U_j^e(\boldsymbol{q}). \tag{5.13}$$

β represents an uncertainty tolerance.[3] If we find enough edges in other views that disagree with $\boldsymbol{P}$, we define $\boldsymbol{P}$ as a bad interpolant, i.e., $D^d(\boldsymbol{p})$ is wrong, and store the average depth difference as its invalidation error:

$$E(\boldsymbol{p}) = \frac{1}{n}\sum_j \left(D_j^e(\boldsymbol{q}) - d_j(\boldsymbol{P})\right). \tag{5.14}$$

In the equation, n is the number of views where Equation (5.13) holds for.[3]

Invalid Area Growing

Because the edge depths are sparse, the above process only removes a few isolated depth areas (see Figure 5.6, E). These information has to be spread to all remaining pixels that most likely have bad interpolants as well (e.g., in Figure 5.9, $\boldsymbol{p}$ should grow towards pixels $\boldsymbol{p}_b$ and $\boldsymbol{p}_f$ to invalidate the whole surface between points $\boldsymbol{P}_b$ and $\boldsymbol{P}_f$). Our thought is that, for each pixel with uncertain validity in the diffused depth map D^d, we can calculate two uncertainty values:

- **Expected uncertainty** This value is only based on the measured U_i^e including uncertainty values at edges.
- **Approximated uncertainty** This value is based on the calculated E including invalidation errors at a few interpolated pixels.

If the latter uncertainty value is larger, the interpolant is indicated invalid.

Hereby, as shown in Figure 5.6, we first calculate the expected uncertainty map U^d by diffusing U_i^e. On the other hand, a combined map C is calculated by summing up the values in U_i^e and E, which is subsequently also diffused to get the approximated uncertainty map C^d. If $C^d(\boldsymbol{p}) > 1.2U^d(\boldsymbol{p})$, we label $D^d(\boldsymbol{p})$ incorrect by setting $M(\boldsymbol{p}) = 0$, M being a validity mask initialized by $M(\boldsymbol{p}) \leftarrow 1$. We use M to remove those invalid interpolants, producing the corresponding valid depth map, D_i^v in Figure 5.6.

Figures 5.2 and 5.10 show our reconstruction results before and after conflict invalidation. A failure would occur when some low-contrast foreground edges fail to be reconstructed in one view (see Figure 5.10). This case would generate sunken surfaces *behind* the real surfaces. These wrong areas cannot be invalidated because they are deeper than the corresponding edges correctly recovered in other views, i.e., Equation (5.13) fails. This problem, together with the holes reliably created for all occlusion problems, will be solved by the information from other views as introduced in Section 5.4.3.

5.4.3 Cross-View Propagation

Object edges probably have view-dependent image contrast (i.e., thus an unreconstructed edge in one view might be recovered in another image) and different views reconstruct the depth discontinuities at various 2D places in depth maps, so both depth values and holes in the valid depth maps are not consistent. Because all interpolants occluding the real surfaces have been invalidated and the final depth maps should contain the frontmost surfaces, we improve each depth map by taking the *closest* depth projected from other views. Trivial propagation might also project background surfaces into the holes. So we only fill holes with the data which agree with the hole boundary by thresholding the normalized color differences along the edge of propagated area. Figure 5.10 shows that the propagated estimates effectively overwrite the sunken surfaces and increase the surface completeness.

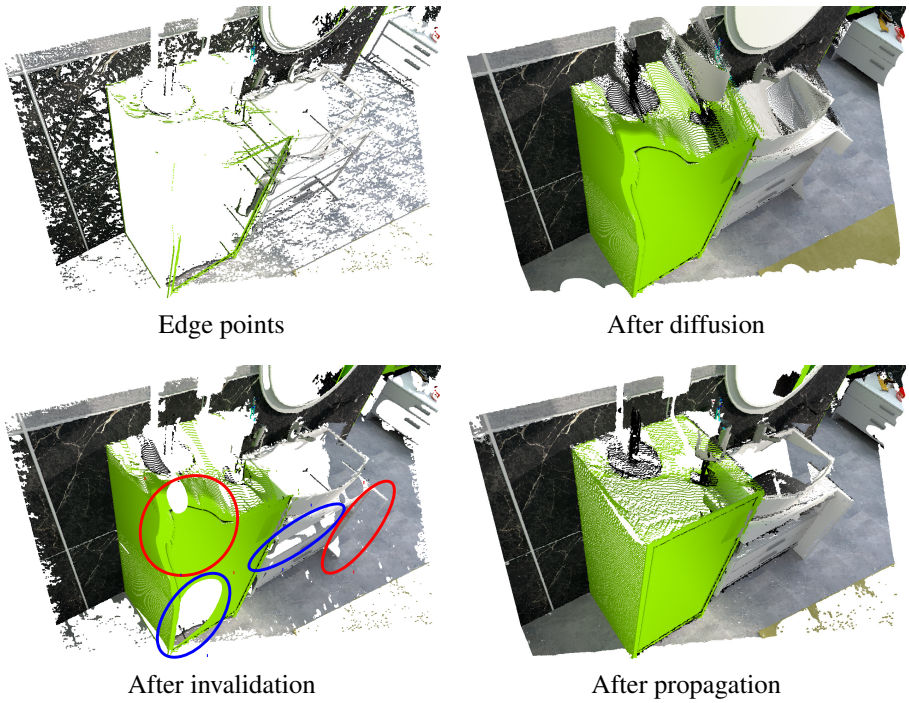


Figure 5.10: Points produced for one view. The depth propagation corrects the sunken surfaces (red) due to lost edge points and fills the holes (blue) arising from invalidation.

5.5 Results

We have tested our approach on four video sequences, each capturing a scene with large homogeneous surfaces along arbitrary camera trajectory (see Figure 5.11). Bathroom and Pabellon were synthetically generated from 3D computer models with ground truth, and the others were acquired from the real world. Each type includes both indoor and outdoor scenes. As our algorithm relies on accurate reconstruction of object edges for surface interpolation and visibility conflict invalidation, we used the full HD resolution (1920×1080) for all images.

We compare our reconstruction with the results of other depth-map-based methods: (Bailer *et al.*, 2012) (BAI), (Kim *et al.*, 2013) (KIM), (Engel *et al.*, 2014) (ENG), and our photos-oriented algorithm (PDMC, proposed in Chapter 4). KIM only probed 256 hypotheses in the depth sweeping. For fair evaluation, we used the new results with 1024 depth values from the authors. We obtained bad camera tracking on Pabellon and Boxes using the SLAM system ENG, so it was only tested on Bathroom and Building. Additionally, because the dimensions of the images input to its source code must be multiples of 16, we cropped the images into 1920×1072 and thus cannot compare its results to the ground truth. Our experiments were run on NVIDIA GeForce GTX 680 (KIM) and Titan (others) GPUs. In the remainder of this thesis, we mention our approach for video-based depth map calculation as VDMC.



Bathroom Pabellon Boxes Building

Figure 5.11: Sample images of the unstructured videos used for testing our approach.

5.5.1 Parameter Settings

Since we sparsely sample the secondary images for about 90° movement around the scene (see Section 5.3.3), experiments demonstrated that using 100 secondary images together with two-scale view sampling provides just enough angular resolution on the surface points for reconstruction.

256 depth values are sufficient for visualizing depth maps as in KIM. But we need more robust estimates for visibility conflict invalidation. So we used 1024 hypotheses to avoid skipping the correct estimates in the depth sweeping process. We found that this setting best compromises on quality and speed.

In Equation (5.1), we set $\sigma_c = 8$ for synthetic scenes and 15 for real-world, i.e., a larger value leads to better noise robustness. In case 2) of our score aggregation (Section 5.3.2), we utilized a stricter score threshold s_1 for dense image samples to greatly refine the initial depth range and a looser threshold s_2 at sparse scale. Specifically, we used $s_1 = 0.8$ and $s_2 = 0.4$ for synthetic scenes but smaller values $s_1 = 0.4$ and $s_2 = 0.2$ for real-world to have higher noise tolerance. In Equation (5.3), we set $\alpha = 0.05$. With more noisy score profiles due to image noise, increasing it can improve the depth completeness.

For the conflict detection, we set the uncertainty tolerance $\beta = 3$ in Equation (5.13) and defined a depth interpolant as invalid only if it conflicts with the edges from at least three views, i.e., $n \geq 3$ in Equation (5.14).

5.5.2 Evaluation

Synthetic scenes. Figure 5.12 presents the depth map comparisons using the ground truth, and the completeness curves for varying relative error thresholds. Table 5.1 lists the quantitative evaluation in terms of completeness and mean relative error. It can be seen that, BAI and PDMC both eliminate most of homogeneous surfaces by removing all inconsistent outliers, although PDMC improves the depth consistency. KIM reconstructs complete scenes but generating discrete artifacts and wrong geometries in textureless areas. In contrary, we only aim to reconstruct the data which is captured and do not guess where the surfaces are most likely. The depth propagation introduces both correct (e.g., the refrigerator corner in Bathroom) and wrong (e.g., the roof in Pabellon) surfaces, but it significantly improves the completeness and we still yield the highest accuracy (see Table 5.1). Figure 5.13 shows how bad the KIM's points are which we do not recover,

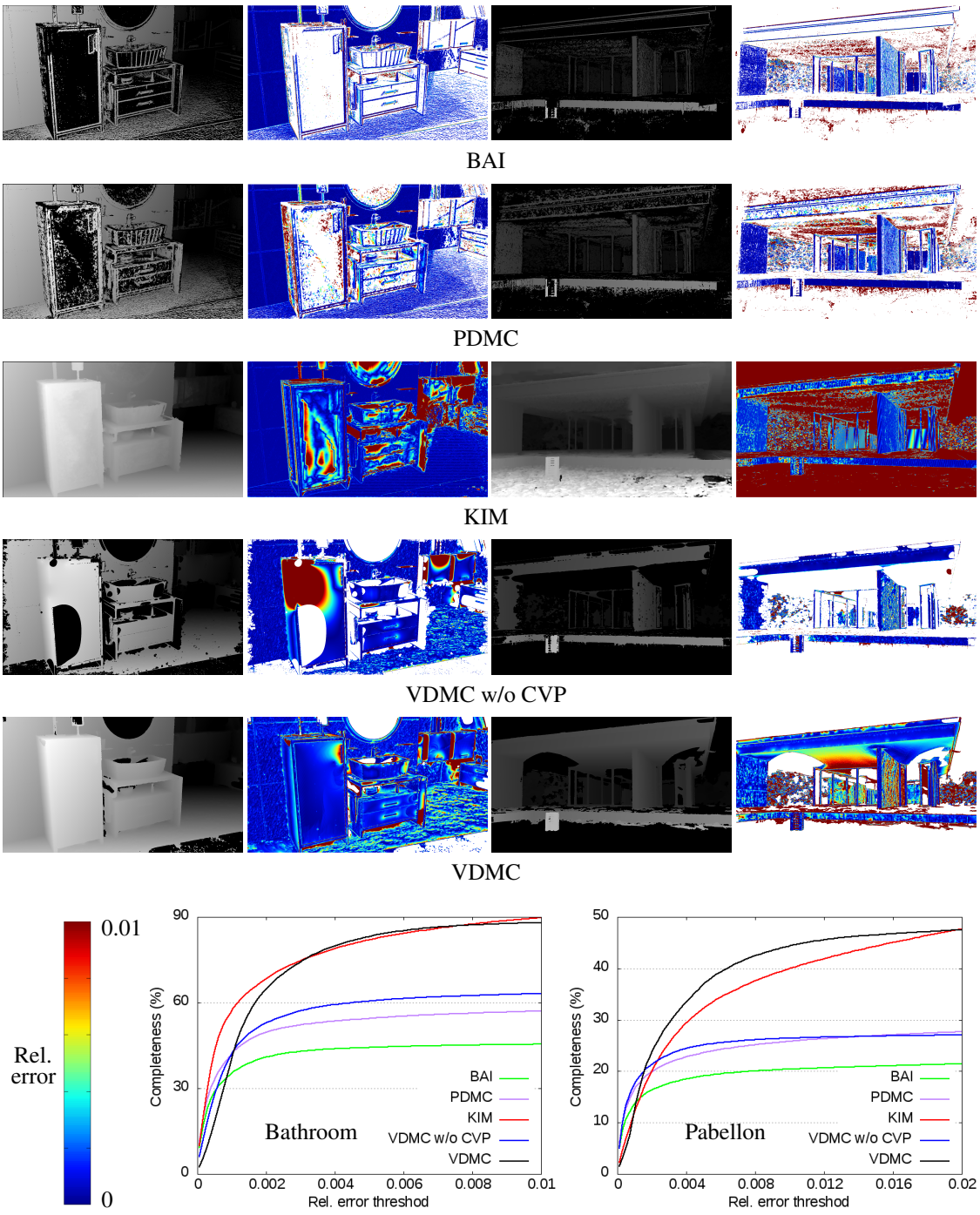


Figure 5.12: Comparisons against the ground truth. For each scene, we present the depth maps (odd columns), the relative error maps (even columns, the white pixels having no estimate or no ground-truth data), and the curves (bottom) measuring how much of the depth estimates has a smaller relative error than a given threshold. w/o CVP is short for without cross-view propagation (Section 5.4.3). Our approach only recovers the reliable surfaces, and invalidates the noisy or inconsistent reconstructions (see Figure 5.13).

Measurement (Bathroom)	BAI	PDMC	KIM	VDMC w/o CVP	VDMC
Completeness (%) ↑	47.499	60.368	100.000	70.221	92.503
Mean Rel. Error ($\times 10^{-3}$) ↓	4.467	4.150	9.867	3.695	3.275
Measurement (Pabellon)	BAI	PDMC	KIM	VDMC w/o CVP	VDMC
Completeness (%) ↑	26.886	35.697	100.000	28.221	56.255
Mean Rel. Error ($\times 10^{-3}$) ↓	69.616	91.890	216.955	7.248	34.065

Table 5.1: Statistical comparisons of the results presented in Figure 5.12. The arrows indicate preferred directions.

Method	Bathroom	Pabellon	Boxes	Building
BAI	68.12	63.28	71.29	75.92
PDMC	34.02	35.14	32.84	35.30
KIM (256 / 1024)	33.72 / 121.55	33.27 / 117.12	31.74 / 119.76	35.77 / 125.39
VDMC	10.21	12.53	11.67	11.07

Table 5.2: Runtime (sec.) for calculating one depth map. The time of KIM testing 256 and 1024 depths are both listed.

and that ENG only reconstructs noisy edges.

Real-world scenes. We compare the real-world reconstructions in Figure 5.14. Although the captured images suffer from noise and compression artifact, we can still recover dense and smooth surfaces without generating wrong interpolants (e.g., the surface of the table recovered by KIM occludes the bottom of the background boxes) or noisy points (e.g., due to the reflection on the windows of the building).

Runtime. Table 5.2 provides the GPU runtimes for reconstructing one view. Our approach achieves the fastest reconstruction even compared with KIM when testing 256 depth hypotheses. The reason is that, we only sweep depth values on edges and our robust score calculation avoids the mean-shift iterations employed by KIM, although we iteratively test 1024 depths. Moreover, our diffusion-based technique is more parallelizable than the region growing strategy of BAI and PDMC.

5.6 Conclusion

We proposed an efficient approach for dense recovery of textureless surfaces from videos. We have discovered that the depth estimates at object edges are sufficient to infer dense reasonable surfaces for the enclosed areas which are assumed to be smooth. Our edge depth calculation is investigated in a novel depth-view space and the incorporated techniques are robust to unstructured camera trajectories. Homogeneous surfaces are reconstructed by diffusing the scattered depth while occlusions are solved effectively. The results demonstrate the superior performance of our algorithm in terms of completeness, accuracy, and consistency.

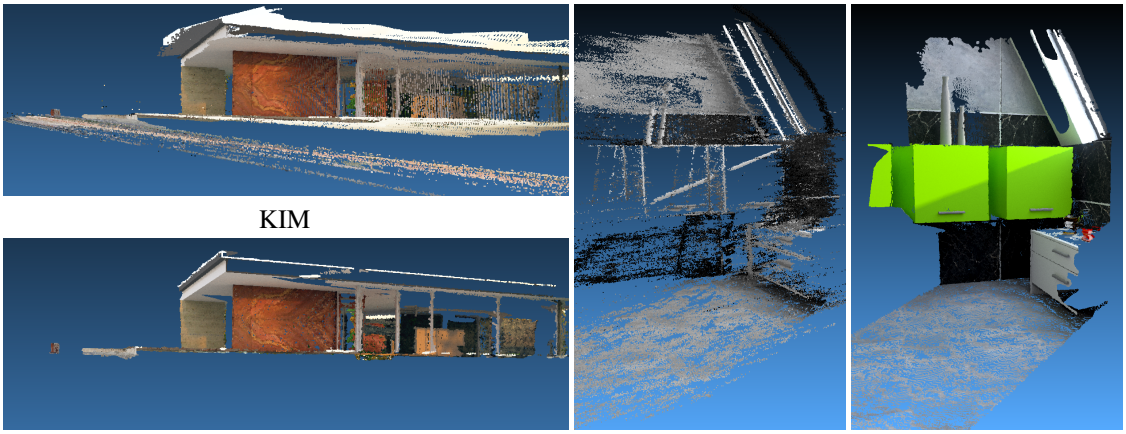


Figure 5.13: Comparisons of the reconstructed points on synthetic scenes. The points of (Engel *et al.*, 2014) exhibit no color due to grayscale input of the source code.



Figure 5.14: Points reconstructed by different methods. The proposed pipeline recovers dense and smooth surfaces in large homogeneous areas. We invalidate the unreliable surfaces by cross-view constraints, and merge the consistent subpixel-accuracy results from multiple views. Since the bottom edge on the back of the foreground box is invisible in almost all images, we fail to get correct interpolants for the table's surface behind thus getting a hole. Wrong surfaces are produced by KIM for most unmeasurable areas.

Chapter 6

Dense and Scalable MVS from Unstructured Videos with Occlusions

Exhaustively estimating depth maps is unfeasible for dense reconstruction of large scenes from videos. Therefore, this chapter is responsible for improving the scalability of the approach proposed in Chapter 5 while maintaining the high surface coverage and strong occlusion robustness. Our scalable method is computationally cheaper and has less memory requirements.

6.1 Introduction

Scalability is a major bottleneck in MVS, particularly for the video-based reconstruction algorithms when much more frames are needed to capture large scenes. As reviewed in Section 3.4, most algorithms only target at modeling single objects or small scenes, process sparse view samples for partial reconstruction, or handle the scale extension by decreasing one or both of computational efficiency and surface quality.

Especially in the presence of large homogeneous areas, the depth-map-based methods typically generate dense estimates per view using smoothness assumption or plane segmentation (see Section 3.1), thus probably producing wrong surfaces which occlude the geometry created from other views. Hence as solved in Section 5.4.2, propagation of information across views is usually required to eliminate the visibility conflicts. In this case, the reconstruction would scale poorly on long image sequences, since redundant computations have to be done on a lot of overlapping views and very large memory space is inevitably occupied to maintain the near-duplicate depth maps.

Instead of processing densely sampled views, this chapter proposes a more scalable approach. The goal is to produce a concise point cloud from far fewer views, while retaining the superior completeness and visibility consistency of the recovered surfaces.

We select the views in a content-adaptive manner by addressing the NBV problem (see Section 3.4 for existing solutions). Our approach incrementally improves the geometric consistency using the newly created points and do not get to another part of the scene until all visibility conflicts are invalidated locally.

As the surface points accumulated, the computational cost of the conflict invalidation step goes up and the memory usage increases linearly. Therefore, we encapsulate the views of each set of locally consistent points into a *cluster* (see Figure 6.1). In this way we can perform the model updating only for the current view cluster, while storing the *merged* points for previous view clusters. The finally integrated point cloud is obtained by removing the inconsistency and redundancy among the points of all view clusters. Our framework is equivalent to an active point clustering process.

For the task of point merging, we exploit the continuous implicit function of (Fuhrmann and Goesele, 2014) (see Section 6.6.1). Besides the point scale, the uncertainty is also taken into account. We use the data redundancy to optimize the points with high surface resolution along visual rays so that the surfaces can be simplified but never be coarsened.

In summary, our main contributions are three-fold:

1. Content-adaptive NBV selection for incremental improvement of visibility consistency and completeness.

2. More efficient reconstruction framework via automatic view clustering, each cluster producing a simplified, optimized, and locally consistent point set.

3. Ray-wise point merging that respects point's scale and uncertainty, and preserves the high surface resolution.

Most parts in our approach are point-wise, thus can be easily parallelized on GPU and multithreaded on CPU. We begin presenting our system by first giving an overview in Section 6.2.

6.2 System Overview

Our scalable algorithm takes as input a (long) video sequence and executes the same preprocessing described in Section 5.2.1. As mentioned in Section 6.1, we construct view clusters to improve reconstruction efficiency (see Figure 6.1). Therefore, our approach can be separated into two levels: partial reconstruction per cluster and multi-cluster integration. The reconstruction begins by picking any of the candidate views as the first reference view of the first cluster.

The *per-cluster* level divides the scene into multiple parts. In each part, the surface recovery is incremental to let model updating and NBV determination benefit from each other (see Figure 6.2). For economization of memory space, once the partial reconstruction finishes, we merge the per-view estimates to obtain a locally consistent and concise point cloud. More specifically, this level performs the following steps:

1. For the current reference view, a set of points are first created from the dense depth map which is calculated using the sparse-to-dense scheme introduced in Sections 5.3 and 5.4.1. The associated information of each point is also maintained (see Section 6.3).

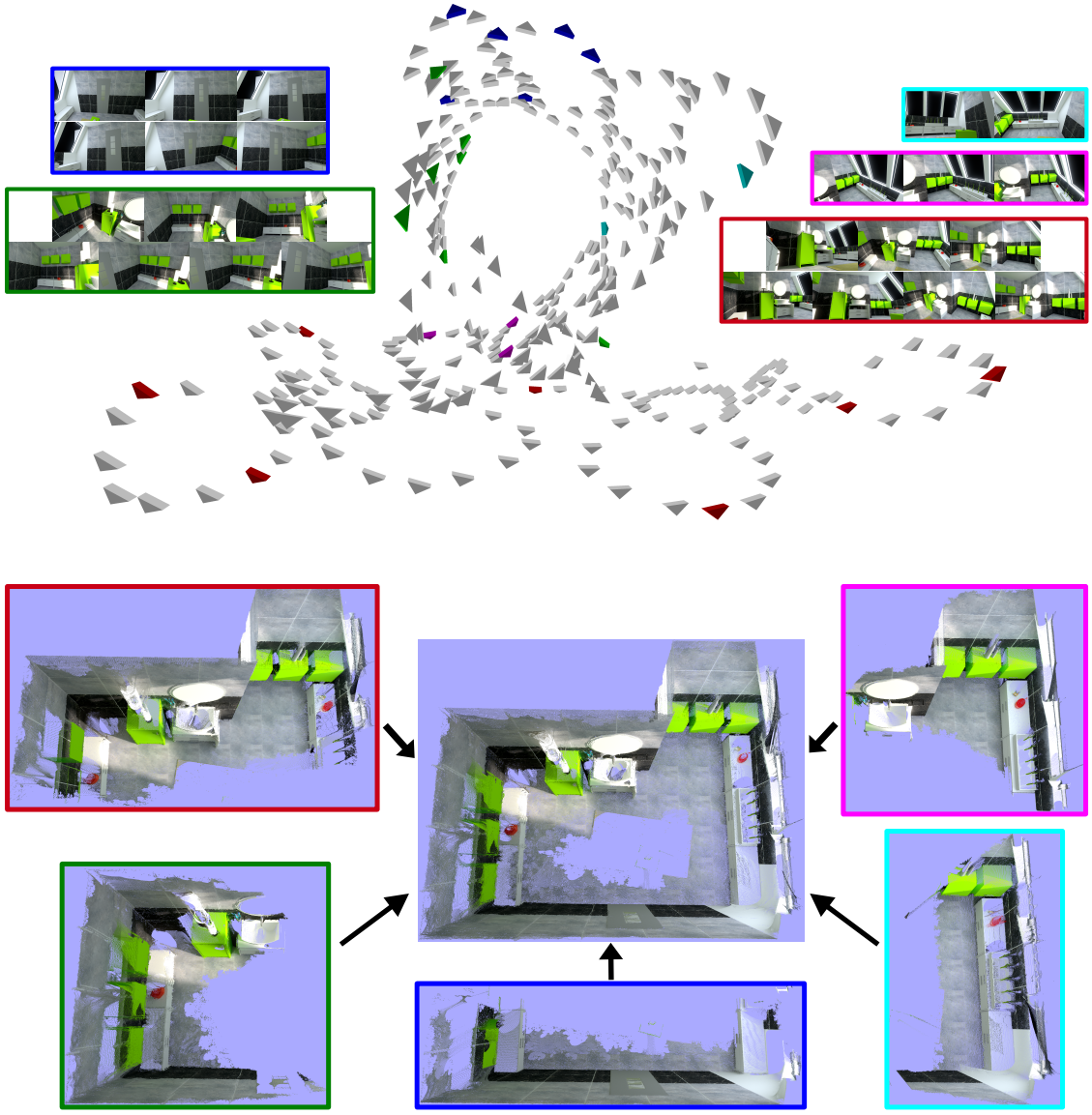


Figure 6.1: A complete indoor scene recovered using our scalable approach. Top: The sampled camera trajectory and the clusters of views selected in our reconstruction. Bottom: Partial reconstructions and the final point cloud. The cameras, images, and point set for each view cluster are marked with the same color. Redundancy and visibility conflicts are removed at both per- and multi- cluster levels.

2. Then the detectable visibility conflicts are invalidated from both the new and existing points by checking inconsistency between (see Section 6.4).

3. Afterwards we select the NBV as the next reference view. Our geometry-aware method aims at seeing as many wrongly occluded surfaces as possible. When the consistency improvement reaches convergence, it automatically creates a new cluster by selecting a view that captures novel scene content (see Section 6.5).

4. If the reconstruction for the current view cluster finishes, the accumulated points are simplified and optimized (see Section 6.6).

5. The above steps are iterated until all parts of the scene are traversed.

At the *multi-cluster* level, the conflict invalidation (Section 6.4) and point merging (Section 6.6) are implemented in order over all clusters to remove global inconsistency and redundancy.

6.3 Point Representation

Our scalable approach parameterizes each point estimate by

$$\boldsymbol{P} := \{v, c, b, u, s, \boldsymbol{x}, \boldsymbol{n}, \boldsymbol{c}, \boldsymbol{N}\}. \tag{6.1}$$

v and c are the indexes of view and cluster, respectively. That means, the point is created from the vth processed view and maintained in the cth cluster. $b \in \{0,1\}$ indicates whether the point is created from an edge pixel. u and s separately denote the point's uncertainty and scale. The former measures the imprecision when calculating depth and the latter conveys the surface area the point covers. As drawn in Figure 6.3, we use the depth's standard deviation defined in Equation (5.6) as the point uncertainty but measured along the visual ray, and the footprint (Fuhrmann and Goesele, 2011) of the depth map pixel as the point scale, i.e., twice the 3D distance to the nearest neighbor. $\boldsymbol{x}$, $\boldsymbol{n}$, and $\boldsymbol{c}$ are the 3D position, surface normal, and RGB color. Due to the smoothness of diffused depth values, the normal is simply computed using the central difference of point positions. $\boldsymbol{N}$ represents the index set of n neighboring points around the point (n might change during the reconstruction), and is used in our conflict invalidation (see Section 6.4). We initialize $\boldsymbol{N}$ using the 4-connected neighborhood and renew it once the point cloud is updated.

6.4 Visibility Conflict Invalidation

To incrementally remove the visibility conflicts, we perform the region-growing-based scheme proposed in Section 5.4.2. As described in Section 6.2, our algorithm filters out these occlusion errors iteratively, therefore it would lead to expensive computational

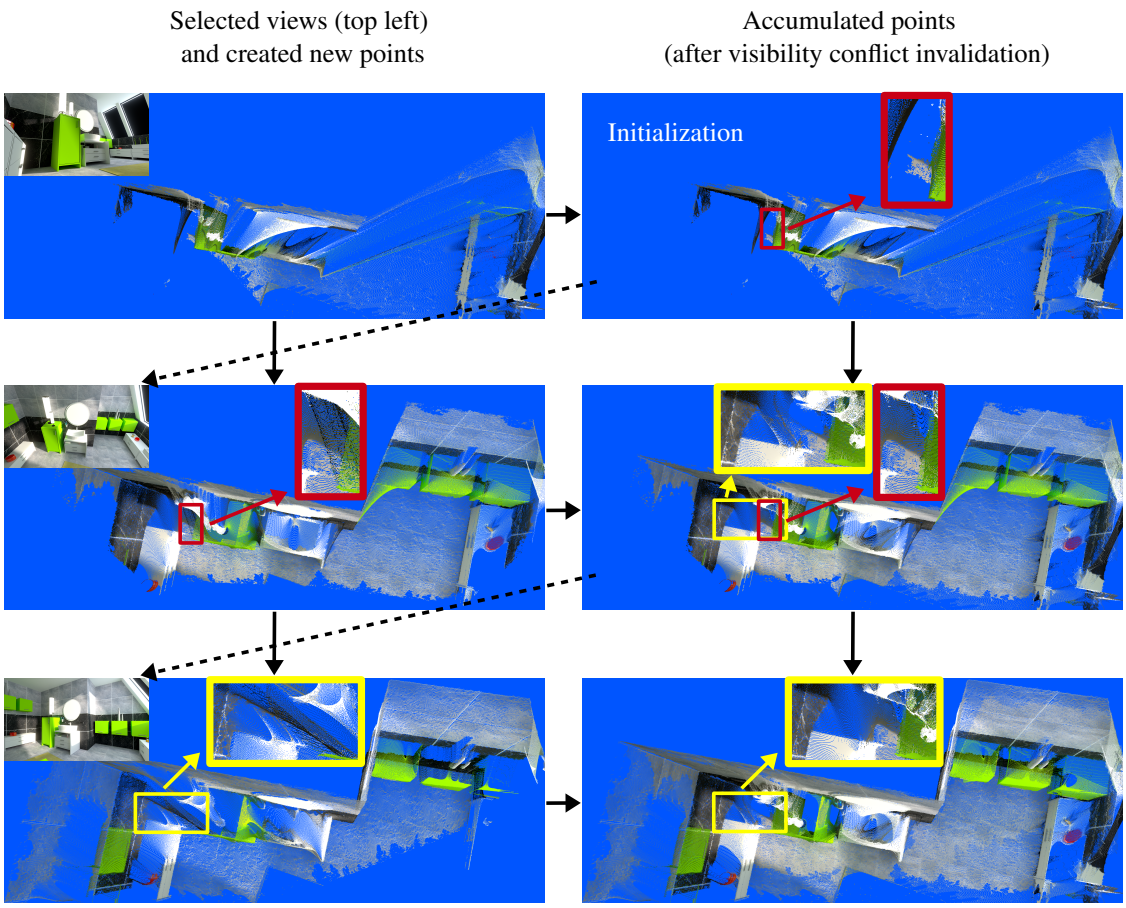


Figure 6.2: Automatically selected views and immediate results of our incremental reconstruction. Due to space limitation, only the first three processed views are presented. See Figure 6.1 for all selected views. Note how the occlusion errors in the accumulated points are gradually addressed using the new points and how the errors generated from the new views (the enlarged regions) are left out of the model.

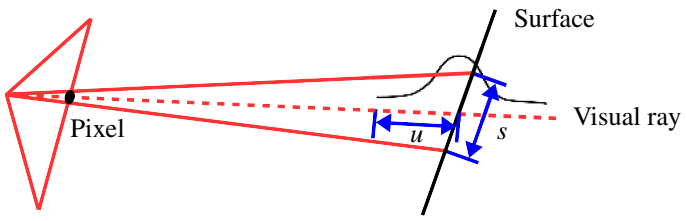


Figure 6.3: Uncertainty and scale of a point estimate. The uncertainty u is measured along the visual ray, and the scale s is calculated on the reconstructed surface.

costs if we implement such image-space invalidation for all (processed and new) views whenever a set of new points are created. We address the issue by extending this method to support growing invalid areas over the *accumulated* surfaces rather than in single depth maps. This is achieved by exploiting the maintained point neighborhood connections, i.e., $\boldsymbol{N}$ in Equation (6.1).

Since the local area invalidation (the first step of Section 5.4.2) relies on back-projecting the interpolated points to the edge depth maps of different views and respecting the depth uncertainties, our reconstruction system needs to additionally maintain the edge's depth and uncertainty maps of all processed views. However, because our view selection enables us to process a minimum subset of views, the memory requirements of our approach are still modest.

In particular, let $\mathcal{P}$ be the point cloud reconstructed so far. The two sets $\boldsymbol{D}^e$ and $\boldsymbol{U}^e$ denote the edge's depth and uncertainty maps of the views producing $\mathcal{P}$, respectively. Assume that a point set $\mathcal{P}_n$ is created from a new view with its edge's depth and uncertainty maps being D_n^e and U_n^e. Our approach eliminates the inconsistency between $\mathcal{P}$ and $\mathcal{P}_n$ by refining $\mathcal{P}$ using D_n^e and U_n^e followed by conversely refining $\mathcal{P}_n$ using $\boldsymbol{D}^e$ and $\boldsymbol{U}^e$.

To refine $\mathcal{P}$, we first judge the invalidities of its non-edge points using the local invalidation method of Section 5.4.2.[4] The sparse invalidation errors calculated by Equation (5.14) are stored in a set $\mathcal{E}$ (with the same size of $\mathcal{P}$). To grow the detected invalid areas, we utilize the same strategy of Section 5.4.2 which compares two uncertainty values for each point. To this end, we store the uncertainties of all points in $\mathcal{P}$ in a dense set $\mathcal{U}$ as their *expected* uncertainties. We diffuse $(\mathcal{E}+\mathcal{U})$ using the point neighborhood information obtaining another dense uncertainty set $\mathcal{C}$ as the *approximated* uncertainties. If the value of a non-edge point in $\mathcal{C}$ is 1.2 times larger than the corresponding value in $\mathcal{U}$, this point is invalid thus removed.

The procedure for refining $\mathcal{P}_n$ is similar. The difference is that, the definitely invalid areas (or the invalidation errors in $\mathcal{E}$) are obtained by back-projecting $\mathcal{P}_n$ to all the views of $\boldsymbol{D}^e$. As introduced in Section 6.2, the conflict invalidation is also performed at multi-cluster level, i.e., among multiple point sets each obtained from one view cluster (see Section 6.6).

Figure 6.2 presents the accumulated points after removing the conflicts between the new and already reconstructed points. It shows that the new points increase the surface completeness while the accumulated points are visibility-consistent with respect to the currently selected views.

6.5 Content-Aware View Selection and Clustering

We propose a content-adaptive method for determining which view to process next, the so-called NBV, and where to encapsulate the views into a cluster. In our approach, these

[4] See Section 6.7.1 for our parameter settings.

two tasks depend on the visibility of the currently reconstructed part of the scene and the view's abilities to handle potential occlusions. Completeness of the entire scene is increased by constructing more view clusters.

Section 6.5.1 first describes our view evaluation to select the NBV for one view cluster. The criterion is extended in Section 6.5.2 to enable active view clustering.

6.5.1 NBV Determination

As described in Section 6.4, our conflict invalidation needs to reconstruct the edges occluded by wrong surfaces (see also Section 5.4.2). For this reason, we select the NBV by giving higher priority to the views seeing the recovered surfaces in a *novel* and *front* perspective. Below, we explain our view evaluation using a point set $\mathcal{P}$ which is created from a view set $\boldsymbol{V}$.

To solve the occlusion errors of $\mathcal{P}$, we define a view score for each NBV candidate V_j at a point $\boldsymbol{P}_i \in \mathcal{P}$ by

$$s(\boldsymbol{P}_i, V_j) = \left(\max(0, \boldsymbol{n} \cdot (-\boldsymbol{v}))\right)^2, \tag{6.2}$$

where $\boldsymbol{n}$ denotes the point normal and $\boldsymbol{v}$ the viewing direction in which V_j observes $\boldsymbol{P}_i$. The squaring operation is used for more distinguishable score peak over all view candidates. We truncate negative values of the dot product to zero. Therefore, V_j is assigned a higher score at $\boldsymbol{P}_i$ if it is closer to the fronto-parallel view of the point's surface patch, and a zero score if seeing the surface from the opposite side.

The NBV can be simply determined by calculating all point-wise scores for each V_j and picking the view with the maximum of the score averages

$$s_{\text{mean}}(\mathcal{P}, V_j) = \frac{1}{m} \sum_{\boldsymbol{P}_i \in \mathcal{P}} s(\boldsymbol{P}_i, V_j), \tag{6.3}$$

where m is the number of points possibly visible in V_j, i.e., the dot product in Equation (6.2) is above zero. For enough overlap of captured content, we incorporate a parameter[4] to only consider the view candidates with

$$m > \alpha |\mathcal{P}|. \tag{6.4}$$

The drawback of Equation (6.3) is that, it would suggest the view next to the previously selected view due to their similar viewing perspectives. Since our scalable system processes *sparse* view samples, we require the NBV to have a clear perspective distinction from the already processed views. Towards this end, we favor the views causing the most significant increase of the view scores.

Specifically, we incorporate the maximum score at $\boldsymbol{P}_i$ achieved by the previously selected views $\boldsymbol{V}$:

$$s_{\max}(\boldsymbol{P}_i, \boldsymbol{V}) = \max_{V_k \in \boldsymbol{V}} s(\boldsymbol{P}_i, V_k). \tag{6.5}$$

Then the change of $s_{\max}$ when selecting V_j as the NBV is

$$s^{\Delta}_{\max}(\boldsymbol{P}_i, \boldsymbol{V}, V_j) = s_{\max}(\boldsymbol{P}_i, \boldsymbol{V} \cup V_j) - s_{\max}(\boldsymbol{P}_i, \boldsymbol{V}). \tag{6.6}$$

Hereby we determine the NBV by finding the view maximizing the average of $s^{\Delta}_{\max}$:

$$s^{\Delta}_{\text{mean}}(\mathcal{P}, \boldsymbol{V}, V_j) = \frac{1}{n} \sum_{\boldsymbol{P}_i \in \mathcal{P}} s^{\Delta}_{\max}(\boldsymbol{P}_i, \boldsymbol{V}, V_j). \tag{6.7}$$

If the s^{Δ}_{mean} value of the NBV is smaller than a threshold τ, we say that the consistency refinement of $\mathcal{P}$ is converged and thus terminate the reconstruction.

As shown in Figures 6.1 and 6.2, our method is able to select a small set of views that are still sufficient to capture all scene structures and remove all occlusion errors.

6.5.2 Active View Clustering

The visibility constraint (Equation (6.4)) makes the view selection scheme of Section 6.5.1 merely apply to small scenes. Hence, we design an *active* view clustering strategy for large-scale reconstruction. Our main idea is that, once the surfaces of the current view cluster are locally consistent, we lower the request in visibilities to reselect the NBV, which creates a new view cluster so that the surface coverage can be gradually extended. Furthermore, clustering views also results in a great saving in computational and memory efficiency as we can perform the conflict invalidation (Section 6.4) only for the current view cluster while merging the points (see Section 6.6) for each previous cluster.

Concretely, let $\boldsymbol{V}$ contain all views in the existing clusters and $\mathcal{P}$ be the points produced from $\boldsymbol{V}$. Let $\boldsymbol{V}_c$ represent the current view cluster and $\mathcal{P}_c$ denote the points accumulated from $\boldsymbol{V}_c$, i.e., $\mathcal{P}_c \subset \mathcal{P}$ and $\boldsymbol{V}_c \subset \boldsymbol{V}$. Our active view clustering works as follows:

1. We first select the NBV, denoted as V_n, by maximizing $s^{\Delta}_{\text{mean}}(\mathcal{P}, \boldsymbol{V}, V_j)$[5], where V_j is a view candidate. If $s^{\Delta}_{\text{mean}}(\mathcal{P}_c, \boldsymbol{V}_c, V_n) > \tau_l$[4], it means that $\mathcal{P}_c$ is inconsistent. So we continue refining $\mathcal{P}_c$ and add V_n into $\boldsymbol{V}_c$.

2. Otherwise, we select V_n again by halving the visibility threshold α (in Equation (6.4)).[6] Still, if $s^{\Delta}_{\text{mean}}(\mathcal{P}_c, \boldsymbol{V}_c, V_n) > \tau_l$, it denotes that V_n probably sees the occluded geometry behind some small surfaces of $\mathcal{P}_c$ but more novel parts of the scene are visible in it. Hence, we set up a new cluster with V_n and leave the invalidation of potentially remaining visibility conflicts to the multi-cluster level.

3. Otherwise, V_n is reselected by neglecting the visibility constraint (Equation (6.4)).[6] If $s^{\Delta}_{\text{mean}}(\mathcal{P}, \boldsymbol{V}, V_n) < \tau_g$[4], it means that the scene geometry visible in V_n has already been recovered from $\boldsymbol{V}$, and consequently the whole reconstruction finishes.

[5] We use $\mathcal{P}$ and $\boldsymbol{V}$ to avoid picking the NBV which is similar to a view in the previous clusters.

[6] The views satisfying the visibility constraint of previous steps are not considered in the reselection.

4. Otherwise, a novel part of the scene is to be recovered by generating a new view cluster with V_n.

Figure 6.1 shows that we can deal with large scenes by clustering the selected views. The overlap of scene content between neighboring clusters allows conflict removal among the clusters and the redundant points are merged finally.

6.6 Ray-Wise Point Merging

The accumulated points contain many redundant and potentially noisy estimates leading to expensive memory consuming and poor reconstruction. Therefore, we merge them to get a *concise* and *optimal* point cloud. Similarly to the conflict invalidation (Section 6.4), our point merging is implemented at both per- and multi-cluster levels. The difference is at the per-cluster level, i.e., the point cloud is invalidated incrementally but merged only after all conflicts are removed.

We exploit the continuous, signed implicit function of (Fuhrmann and Goesele, 2014) (see Section 6.6.1). Our and their approaches mainly differ in three ways:

1. They *discretely* sample the implicit function based on a scale-aware octree and reconstruct an isosurface *mesh* corresponding to the zero-level set. Conversely, we obtain optimized *points* by seeking the *exact* intersections between the isosurface and visual rays of the input points.

2. Although the point scale is respected in their method, the small-scale points might still slightly decrease the final resolution of the merged surfaces. In contrast, our goal is to preserve the surface resolution while removing the redundancy at the same time. To do this, the points with low resolution, i.e., large scale, are not optimized but used for optimizing other points and then are eliminated. This reduces the computational effort and also avoids degrading the high-resolution surfaces.

3. The point uncertainty is also incorporated in our approach to limit the search range.

Section 6.6.2 details the proposed optimization and simplification of points. In Section 6.6.3, we present how the point's associated information is updated. For fast search operations in 3D space, we insert all input points into an octree regarding their scales. Please see (Fuhrmann and Goesele, 2014) for the octree building.

6.6.1 Implicit Function

The implicit function has positive values in front of the surface and negative behind. Its value at the 3D position $\boldsymbol{x}$ is calculated using a weighted average of basis functions:

$$F(\boldsymbol{x}) = \frac{\sum_{\boldsymbol{P}_i} w(\boldsymbol{x}_i) f(\boldsymbol{x}_i)}{\sum_{\boldsymbol{P}_i} w(\boldsymbol{x}_i)}, \tag{6.8}$$

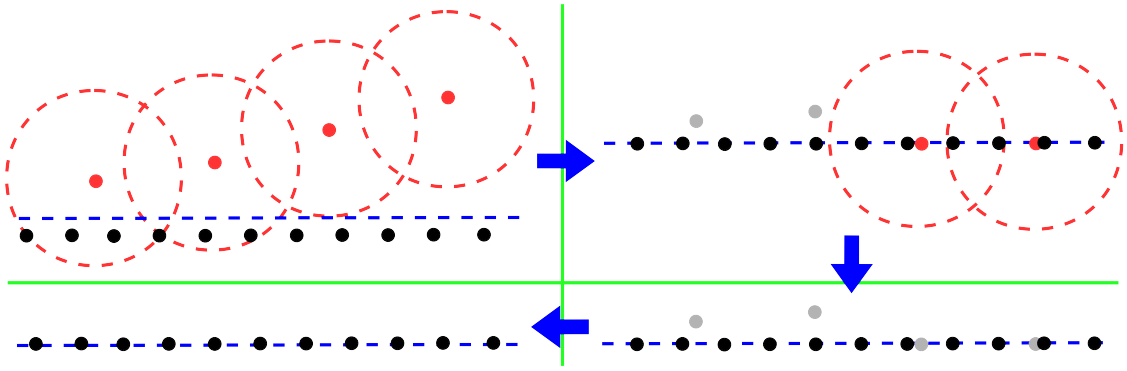


Figure 6.4: Merging two point sets at different resolutions. The low-resolution points (left two red points) whose neighborhood (dashed circles) covers high-resolution points (black points) are not optimized. Some farther low-resolution points (right two red points) may be optimized onto the isosurface (blue dashed line) of high-resolution points. They are post-detected via second neighborhood checking. All low-resolution points (gray points) are eliminated subsequently.

$$\boldsymbol{P}_i \in \mathcal{P}_f(\boldsymbol{x}) : ||\boldsymbol{x} - \boldsymbol{x}_i|| < 3s_i. \tag{6.9}$$

$\mathcal{P}_f$ is a subset of input points that influence $\boldsymbol{x}$. For each point in $\mathcal{P}_f$, the corresponding basis function f is rotation invariant around the point normal and contributes to F with the same volume but different distribution depending on the point scale. See (Fuhrmann and Goesele, 2014) for the formulation of f and the weighting function w.

6.6.2 Point Optimization and Simplification

We optimize a point by testing position hypotheses along its visual ray and calculating the implicit function for each hypothesis until the zero crossing is found.

Because only the highest-resolution points survive in our approach, it makes no sense to optimize the points which will be removed later anyway. Therefore, before optimization we first roughly judge whether a point has a relatively low surface resolution and ignore it if yes. For this purpose, we seek a set of smaller-scale points $\mathcal{P}_r$ within a spherical neighborhood centered on each point $\boldsymbol{P}$ as shown in Figure 6.4, such that

$$\boldsymbol{P}_j \in \mathcal{P}_r(\boldsymbol{x}) : ||\boldsymbol{x} - \boldsymbol{x}_j|| < \frac{1.5s}{2},\ s > s_j, \text{ and } v \neq v_j. \tag{6.10}$$

$v \neq v_j$ denotes that $\mathcal{P}_r$ excludes the points from the same view of $\boldsymbol{P}$. If $\mathcal{P}_r \neq \emptyset$, it demonstrates that $\boldsymbol{P}$ has a low resolution, and then we *label* this point but do not remove it for the moment.

Next we optimize the *unlabeled* points using *all* input information. Let $\boldsymbol{v}$ denote the visual ray direction of $\boldsymbol{P}$. We optimize its position $\boldsymbol{x}$ as follows:

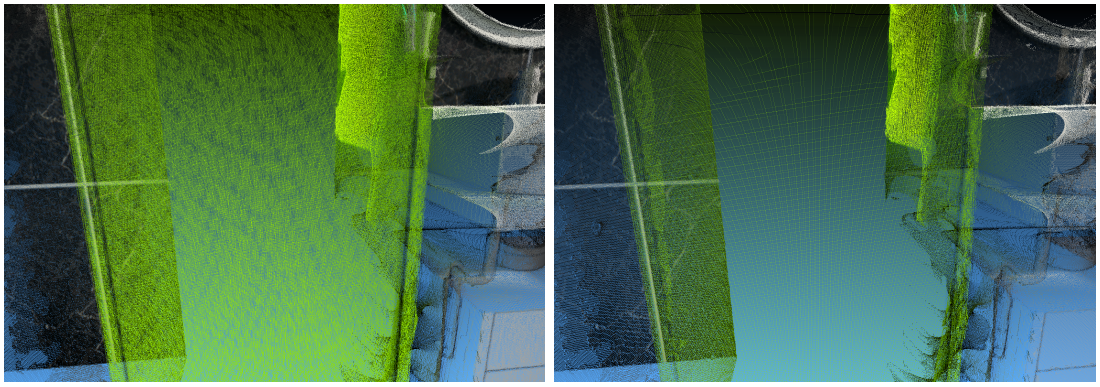


Figure 6.5: Points before (left, 32434 points) and after (right, 10251 points) our point merging procedure. Points are simplified but the surface resolution is preserved.

1. The zero crossing of the implicit function F is searched along the line segment $\boldsymbol{l}$ connecting $(\boldsymbol{x} - \boldsymbol{v}u)$ and $(\boldsymbol{x} + \boldsymbol{v}u)$ by considering the point uncertainty u.

2. Instead of redundantly finding $\mathcal{P}_f$ (see Equation (6.9)), we first calculate a larger point subset $\mathcal{P}_f(\boldsymbol{l})$, within which each tested position $\boldsymbol{x}'$ can yield its $\mathcal{P}_f(\boldsymbol{x}')$ easily.

3. We adopt a progressive search starting from $\boldsymbol{x}$ and the search direction determined by the sign of calculated F after each testing. The shift along $\boldsymbol{v}$ is in inverse depth to allow for the distance to the camera. The absolute value of the shift is initialized as 0.001 and halved if the F values of the two previous hypotheses cross zero. The optimal position $\hat{\boldsymbol{x}}$ is determined until the change of F is below a certain threshold.

As illustrated in Figure 6.4, some relatively distant low-resolution points might survive the neighborhood checking, i.e., Equation (6.10), but then optimized onto the same surface of high-resolution points. We remove these redundant estimates by performing the checking once more after the optimization. To avoid rebuilding the octree and redoing the radius search using the new point positions, we obtained almost the same results by finding the point subset $\mathcal{P}_r(\hat{\boldsymbol{x}})$ within $\mathcal{P}_f(\boldsymbol{l})$. The newly detected low-resolution points together with the pre-labeled points are eliminated from the merged point cloud.

Figure 6.5 compares the points before and after merging. Our method can remove the redundant estimates while maintaining the high surface resolution. See also Figure 6.8.

6.6.3 Associated Information Updating

After a point gets an optimal position, other associated attributes are updated accordingly. As in (Fuhrmann and Goesele, 2014), the normal and color are calculated using a second, simpler implicit function (see the paper for details). The neighborhood information ($\boldsymbol{N}$ in Equation (6.1)) is updated by merging the connection information of surviving neighboring points. The point scale is recalculated on the optimized surface.

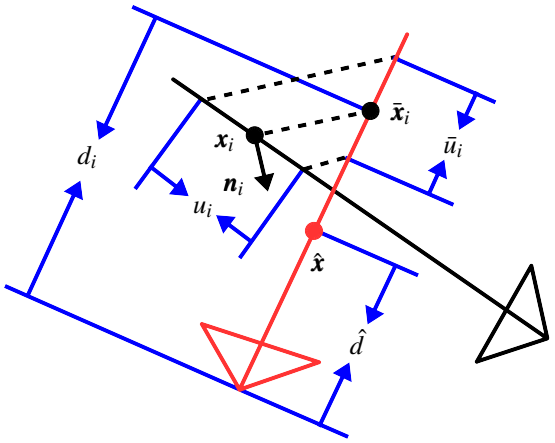


Figure 6.6: Explanation of our uncertainty merging. We project the position $\boldsymbol{x}_i$ and uncertainty u_i of each point that influences the new position $\hat{\boldsymbol{x}}$ onto the visual ray (red) of the optimized point. Our method considers both the projected uncertainty $\bar{u}_i$ and the depth difference $|d_i - \hat{d}|$ along the ray.

To calculate the point uncertainty, we utilize the algorithm of Crouse *et al.* (2011) which is devoted to merging multiple Gaussian components. Since the uncertainty is defined along visual rays, we make a modification to support cross-ray merging. More concretely, for the point $\boldsymbol{P}$ at $\hat{\boldsymbol{x}}$, we project the position and uncertainty of each point $\boldsymbol{P}_i \in \mathcal{P}_f(\hat{\boldsymbol{x}})$ along the tangent plane onto the ray of $\boldsymbol{P}$, as depicted in Figure 6.6. Let the projected position and uncertainty are $\bar{\boldsymbol{x}}_i$ and $\bar{u}_i$, respectively. By denoting the depths of $\bar{\boldsymbol{x}}_i$ and $\hat{\boldsymbol{x}}$ along the ray as d_i and $\hat{d}$, the merged uncertainty is defined by

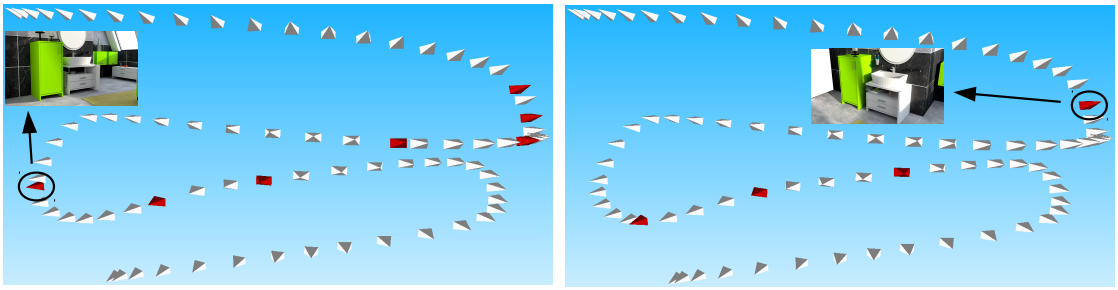
$$\hat{u} = \frac{\sum_{\boldsymbol{P}_i} w(\boldsymbol{x}_i)(\bar{u}_i + |d_i - \hat{d}|)}{\sum_{\boldsymbol{P}_i} w(\boldsymbol{x}_i)}. \tag{6.11}$$

Hereby, the new value combines the input point uncertainties and respects the surface distances as well. After updating all points, we diffuse the point uncertainties over the surface for smoothness.

6.7 Results

We tested four different scenes to assess the performance of our scalable algorithm, two synthetic and two real-world. The synthetic scenes include the Bathroom used in Chapter 5, and a larger dataset covering the entire bathroom, denoted as BathroomL. The real-world scenes are both in outdoor environment with a large scale, separately named BuildingL and ChurchL. As explained in Section 5.5, we used the image resolution of 1920×1080 for accurate reconstruction of the object edges.

In Section 5.5, we have proved the superiority of our video-based depth map calculation method (VDMC, proposed in Chapter 5) over (Bailer *et al.*, 2012), (Kim *et al.*, 2013), (Engel *et al.*, 2014), and our MVS approach for unorganized photos (proposed in Chapter 4). Therefore, this section shows the scalability improvement by comparing our scalable point cloud reconstruction algorithm, denoted as SPCR, to VDMC. The



View subset 1 View subset 2

Figure 6.7: Selected views of our method (red) for Bathroom by beginning from different views (see the images). For clear visualization, the trajectories of unselected views are sampled (every 10).

traditional cluster-based MVS approach CMVS (Furukawa *et al.*, 2010) was also implemented. Unlike our incremental reconstruction, CMVS achieves scalability by *independently* processing multiple view clusters and the selected views in each cluster. Since CMVS's view clustering and selection rely on the SfM features and associated visibilities while we exploited the computer-generated camera extrinsics for synthetic scenes, CMVS was only used for reconstructing real-world scenes in our experiments. All tests were run on NVIDIA GeForce Titan GPUs and multithreaded CPUs (OpenMP).

6.7.1 Parameter Settings

The performance of our approach is sensitive to four parameters. First, for the conflict detection (Section 6.4), we used a larger uncertainty tolerance by setting $\beta = 12$ in Equation (5.13) to avoid over-invalidation by inaccurate edge estimates, and defined a point as invalid if it conflicts with the edges from more than one views. Next, $\alpha = 0.8$ was used in Equation (6.4) to let 80% of the model be visible in the NBV. A smaller value would produce fewer and larger view clusters. As the convergence criteria for NBV selection (see Section 6.5.2), we used $\tau_l = 0.007$ for the points of the current cluster and $\tau_g = 0.009$ for the whole point cloud. We let $\tau_g > \tau_l$ to avoid small, unnecessary clusters at the end of the reconstruction which are built by the views capturing the duplicate scene content but still leading to a slight increase of the s^{Δ}_{mean} value. Increasing these two parameters probably makes some visibility conflicts survive the occlusion invalidation procedure, and decreasing them would make us process redundant views.

6.7.2 Evaluation

Synthetic scenes. Ground-truth depth maps are available for the synthetic scenes. As our approach outputs a point cloud, we measure its completeness by back-projecting the points to the ground truth and then calculating the average percentage (related to

Measurement	Bathroom							BathroomL		
	VDMC-Full	View subset 1			View subset 2					
		SPCR w/o PM	SPCR	VDMC-Sub	SPCR w/o PM	SPCR	VDMC-Sub	SPCR w/o PM	SPCR	VDMC-Sub
Selected views (clusters) ↓	100	7 (1)	7 (1)	7	4 (1)	4 (1)	4	25 (5)	25 (5)	25
Point number ($\times 10^3$)	151210	10921	3920	10922	6313	3426	6276	31436	11648	31477
Completeness (%) ↑	86.774	85.452	84.769	84.767	82.261	81.596	82.183	84.831	81.798	84.577
Mean error ($\times 10^{-3}$) ↓	3.670	3.857	6.760	3.990	4.498	6.237	4.373	2.949	5.094	3.020
Runtime (sec.) ↓	3267	219	1152	140	115	583	78	746	5691	481

Table 6.1: Statistics for synthetic scenes obtained by VDMC and our method. Bathroom and BathroomL have 100 and 800 candidate reference views, respectively. We also list our results without (w/o) the point merging (PM) step. VDMC-Full and -Sub separately denote VDMC processing all candidate views and only the views selected by ours. We failed to implement VDMC-Full on BathroomL due to the huge memory requirements. See Figure 6.7 for our selected view subsets when reconstructing Bathroom. The arrows indicate preferred directions.

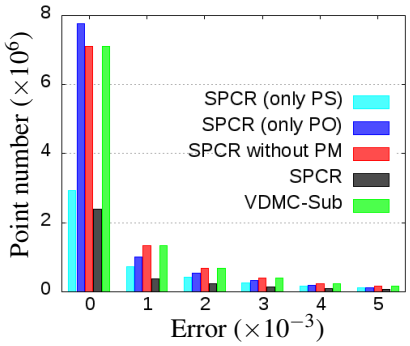


Figure 6.8: Error distributions for Bathroom obtained by the methods and view subset 1 in Table 6.1. Point simplification (PS) reduces the point number and optimization (PO) increases accurate points. Our scalable approach obtains almost the same result compared with VDMC when using only our selected views

the image size) of the projections whose 3D-space errors are smaller than 0.1. The inaccuracy of each point is computed as the smallest error between its position and the ground-truth positions of its per-view projections. High density and high resolution of the reconstructed points enable us to fill up the gaps between the projections using one-pass morphological closing operation. The same measuring method was applied to the points of VDMC which were generated from all estimated depth maps.

Table 6.1 and Figure 6.8 present the quantitative evaluation by turning on/off point merging (PM) and implementing VDMC on all or only our selected views. To test the flexibility of our view selection, we reconstructed the smaller Bathroom scene by separately beginning from two distinct perspectives (see Figure 6.7). From the results, it is evident that:

1. Comparing SPCR w/o PM and VDMC-Sub, our scalable method without merging points obtains comparable surface quality (similar point numbers, completeness, and mean errors) to manually selecting the same views for VDMC. The increase of runtime is the result of our iterative conflict invalidations.

2. Comparing SPCR with and without PM, the points are significantly simplified by the merging step. The increase of mean error is due to the reason that most redundant points have high precision while the wrong estimates are left because of their low density. However, the merging stage does not lead to a large decrease of the surface completeness. The runtime is much longer when performing the merging because we need to perform radius search in the octree for finding $\mathcal{P}_f(\boldsymbol{l})$ in Step 2 of our point optimization.

3. Comparing VDMC-Full and SPCR, our geometry-aware approach still produces more concise points in substantially shorter time compared with the exhausted depth map estimation while maintaining the high surface completeness.

4. Comparing the statistics of the two view subsets on Bathroom, the setting of the first reference view affects the process of our incremental reconstruction but we can still get similar final results.

The same conclusions can be obtained from the point cloud comparison in Figure 6.9.

Real-world scenes. Figures 6.10 and 6.11 show the results of the two real-world scenes. CMVS fails to recover the homogeneous geometry and produces noisy points. It processes more views because the immediate results are not considered in its view selection and clustering. Instead, we generate dense points and our geometry-aware method processes much less views.

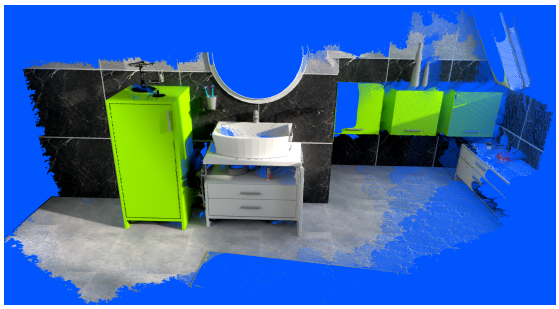
6.8 Conclusion

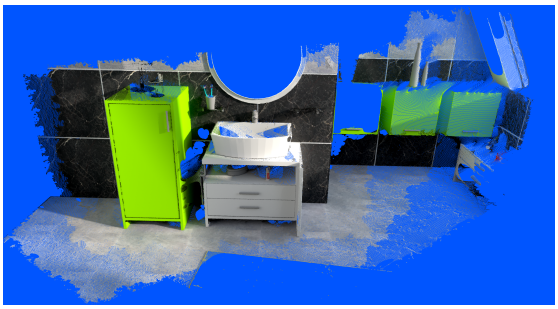
We presented a video-oriented 3D reconstruction framework that improves the scalability of a recently proposed method while maintaining the high surface completeness and strong occlusion robustness. This is achieved by replacing creation of near-duplicate depth maps with incremental occlusion reasoning for existing 3D points. The content-aware view sampling enables us to actively divide the scene model into multiple overlapping point clusters. The clustering allows our approach to work more efficiently at per- and multi-cluster levels. Additionally, our point merging respects per-point characters (scale and uncertainty) and preserves the high surface resolution. The above superiorities make handling large-scale datasets more feasible.

VDMC-Full

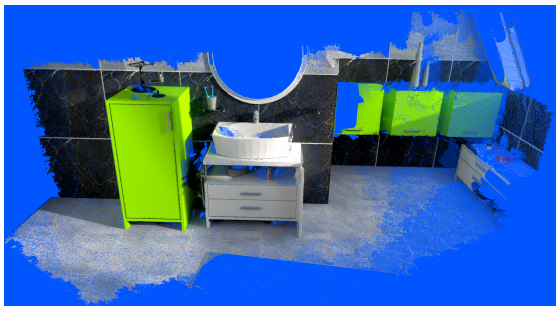


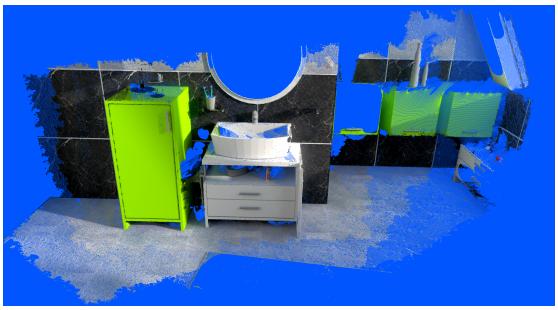
View subset 1 View subset 2





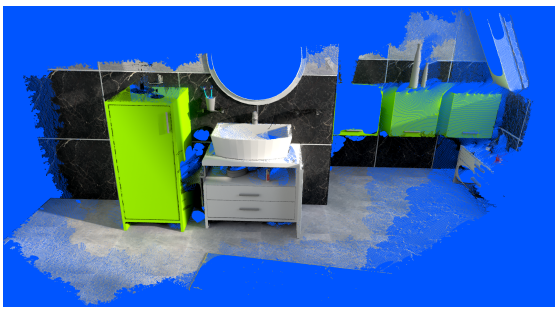
SPCR without point merging





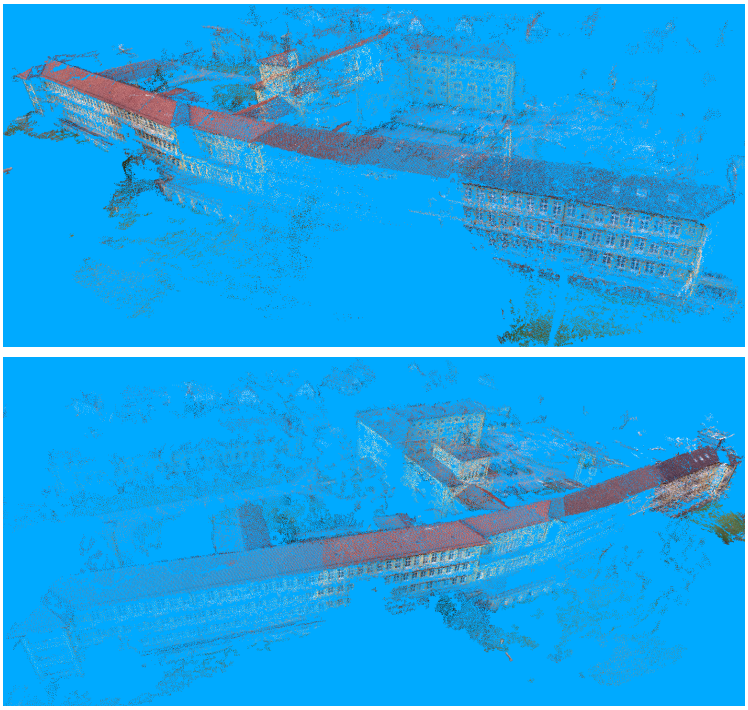
SPCR



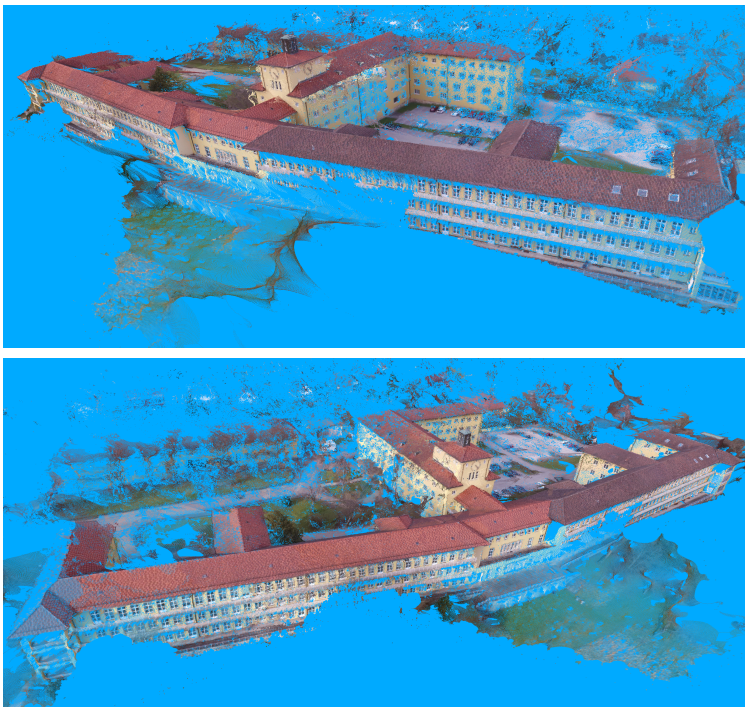


VDMC-Sub

Figure 6.9: Point clouds of Bathroom reconstructed by SPCR without and with the point merging, as well as VDMC on all views (VDMC-Full) and only the views selected by SPCR (VDMC-Sub). The columns compare the results produced from the two selected view subsets (see Figure 6.7).



CMVS



SPCR

Figure 6.10: Point clouds of BuildingL reconstructed by CMVS and our incremental approach (with point merging). CMVS uses 4 clusters of 49 views. Our method constructs 3 clusters including only 13 views.



CMVS SPCR

Figure 6.11: Point clouds of ChurchL reconstructed by CMVS and our incremental approach (with point merging). CMVS uses 4 clusters of 47 views. Our method constructs 3 clusters including only 13 views.

Chapter 7

Conclusion

This chapter concludes this thesis with a summary of our contributions (Section 7.1) and some discussions on the directions of future work (Section 7.2).

7.1 Contributions

The goal of this thesis is to develop new depth-map-based MVS approaches that can be applied to different input data and achieve high performance in terms of accuracy, global consistency, surface completeness, as well as reconstruction scalability. Our main contributions can be divided into three parts:

- **Consistent depth map estimation for photo collections**

  In Chapter 4, we advanced the traditional PatchMatch workflow for faster depth propagation and, more importantly, higher global consistency. By utilizing shorter scanlines and a coarse-to-fine framework for depth propagation, the optimal estimates can be more efficiently spread into larger areas. Our main contribution is that, we proposed a solution to balance between the photo consistency and the cross-view depth coherence such that more good estimates and less outliers are merged into the final surfaces. The evaluation on the Middlebury benchmark shows that our technique is among the most efficient MVS approaches with great precision of the reconstructed models.

- **Dense and occlusion-robust depth map estimation for videos**

  Chapter 5 presented an algorithm for creating dense depth maps from unstructured videos, where we focused our attention on recovering homogeneous surfaces. We first introduced a novel way of visualizing and analyzing the appearance of a point hypothesis in the secondary images. This helps us to more robustly calculate the edge depth with the arbitrary camera movements taken into account. For the major task, our depth diffusion is free of perspective distortions, and our region-growing-based method for visibility conflict invalidation merely relies on the edges reconstructed in other views. The experimental results demonstrate that accurate object

edges are sufficient to recover dense, reasonable surfaces for the enclosed areas using the smoothness assumption.

- **Scalable point cloud reconstruction from videos**

  In Chapter 6, we designed a scalable method for reconstructing dense, concise point cloud instead of producing redundant depth map sequence as in Chapter 5. Our geometry-aware view selection enables us to only process the most valuable views while maintaining the surface completeness and occlusion robustness, and also supports active construction of view clusters each generating a locally consistent point set. Our ray-wise point merging scheme can obtain optimized and simultaneously simplified points with the highest surface resolution preserved. By testing on both small- and large-scale datasets, the superior computational and memory efficiency of our system has been proved.

In summary, the techniques proposed in the thesis provide some new insights into the MVS problem, which are useful for developing novel solutions and also can be further improved for better surface quality or higher reconstruction efficiency.

7.2 Future Work

The results we have obtained suggest multiple directions for future work including theoretical and technical improvements, as well as potential applications.

First, some parameters used in our approaches have to be set by hand in our current implementation (see Sections 4.6.1, 5.5.1, and 6.7.1). Hence future work should enhance the generality and automaticity of the proposed reconstruction systems.

As demonstrated in Section 6.7.2, the most time-consuming task in our scalable MVS algorithm is point merging, where the radius search significantly slows down the whole reconstruction process. The Point Cloud Library (PCL) provides a GPU function for radius search but we failed to get the correct neighbors when using different radii for individual query points. Thus a more effective GPU-based scheme for this problem should be found to make our approach more practical.

In this thesis, the point uncertainty is defined along the visual ray (see Figure 6.3). It may be helpful to formulate it using a covariance matrix, e.g., (Mendez *et al.*, 2016), so that we can measure the uncertainty in any direction.

One limitation of our current work is that we assume Lambertian and smooth surface. Therefore, it is worth investigating how to solve the unreliable photo-consistency constraint induced by rolling shutter, different exposures, varying illumination, glossy or specular surfaces, or semi-transparent materials, e.g., glass or water. Moreover, reconstruction of curved or more complicated surfaces should be further investigated.

An interesting direction for future work is plugging our video-oriented depth estimation methods for edges and homogeneous areas into other MVS techniques. For example,

we could perform the perspective depth interpolation and the edge-based occlusion invalidation for the textureless surface recovery from photographs, if the object edges can be precisely reconstructed.

One possible solution to the depth interpolation from inaccurate object edges is to exploit the semantic segmentation. Furthermore, we can try to integrate the tasks of edge depth calculation, image segmentation, and depth interpolation to design a joint solution.

Another possibly promising direction may be to apply the pixel-level edge reconstruction and occlusion-robust homogeneous-area interpolation to calculate the optical flow (Lang *et al.*, 2012) or intrinsic video (Ye *et al.*, 2014).

After the dense reconstruction of object shapes from videos, the next frontier is the recovery of reflectance and illumination, which relies on the depth information to find correspondences across views.

Texturization of the reconstructed surfaces (Arikan *et al.*, 2016) is also of special interest for future research, since each point is assigned only one color but it might be captured multiple times with varying appearances.

Symbols

$\boldsymbol{p}$	Image pixel or projection of 3D point
x, y	X-, Y-coordinate of image pixel
d	Depth value
v	Depth variance
$\boldsymbol{n}$	2D normal vector used in patch matching
$\boldsymbol{P}$	3D point
$\boldsymbol{O}$	Optical center of camera
$\boldsymbol{v}$	Viewing or visual ray direction
V	Camera view
W	Neighborhood window around a pixel
I	Color image
D	Depth map
N	Normal map
E	Matching or invalidation error map
U	Uncertainty map
C	Uncertainty map combining expected uncertainties and invalidation errors
M	Validity mask, 1 for valid and 0 for invalid
F	Implicit function
$\boldsymbol{V}$	Set of camera views
$\boldsymbol{I}$	Set of images
$\boldsymbol{D}$	Set of depth maps
$\boldsymbol{U}$	Set of uncertainty maps
$\mathcal{P}$	Point cloud
$\mathcal{U}$	Set of point uncertainties
$\mathcal{E}$	Set of invalidation errors
$\mathcal{C}$	Set of point uncertainties combining expected uncertainties and invalidation errors
$\boldsymbol{F}(\cdot)$	Set of feature points in a view
$\boldsymbol{S}(\cdot)$	Set of secondary images (views) for a reference image (view)
$\delta(\cdot)$	Indicator function, 1 for true and 0 for false
$d(\cdot,\cdot)$	Distance between two 3D points
$\lvert\cdot\rvert$	Number of elements in a set
$\lvert\lvert\cdot\rvert\rvert$	Length of a line segment

In Chapter 6, a 3D point $\boldsymbol{P}$ is parameterized as follows:

v	View index
c	Index of view cluster
b	Boolean indicating whether $\boldsymbol{P}$ is created from an edge pixel, 1 for yes and 0 for not
u	Point uncertainty
s	Point scale
$\boldsymbol{x}$	3D position
$\boldsymbol{n}$	3D surface normal
$\boldsymbol{c}$	RGB Color
$\boldsymbol{N}$	Index set of neighboring points

Abbreviations

2D, 3D	Two-, Three-dimensional
HD	High-definition
MVS	Multi-view stereo
SfM	Structure from motion
SLAM	Simultaneous localization and mapping
VRIP	Volumetric range image processing
NCC	Normalized cross correlation
EPI	Epipolar-plane image
GPU	Graphics processing unit
CPU	Central processing unit
OpenMP	Open multi-processing
LP	Local propagation
HF	Hierarchical framework
CVF	Cross-view filtering
CVP	Cross-view propagation
RC	Random checking
MLS	Minimum least square
PSR	Poisson surface reconstruction
NBV	Next best view
CMVS	Clustering views for multi-view stereo
PCL	Point Cloud Library
PM	Point merging
PS	Point simplification
PO	Point optimization

In the evaluations, we mention our new MVS approaches as follows:

PDMC	Photos-oriented depth map calculation (proposed in Chapter 4)
VDMC	Video-based depth map calculation (proposed in Chapter 5)
SPCR	Scalable point cloud reconstruction (proposed in Chapter 6)

Bibliography

Arikan, M., Preiner, R., and Wimmer, M. (2016). Multi-depth-map raytracing for efficient large-scene reconstruction. *IEEE Trans. Vis. Comput. Graph.*, **22**(2), 1127–1137.

Bailer, C., Finckh, M., and Lensch, H. P. A. (2012). Scale robust multi view stereo. In *ECCV*.

Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. (2009). Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, **28**(3).

Besse, F., Rother, C., Fitzgibbon, A., and Kautz, J. (2012). Pmbp: Patchmatch belief propagation for correspondence field estimation. In *BMVC*.

Bleyer, M., Rhemann, C., and Rother, C. (2011). Patchmatch stereo-stereo matching with slanted support windows. In *BMVC*.

Bódis-Szomorú, A., Riemenschneider, H., and Van Gool, L. (2015). Superpixel meshes for fast edge-preserving surface reconstruction. In *CVPR*.

Boubekeur, T., Heidrich, W., Granier, X., and Schlick, C. (2006). Volumesurface trees. *Computer Graphics Forum*, **25**(3), 399–406.

Bourmaud, G. and Mgret, R. (2015). Robust large scale monocular visual slam. In *CVPR*.

Bradley, D., Boubekeur, T., and Heidrich, W. (2008). Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *CVPR*.

Brandao, M., Ferreira, R., Hashimoto, K., Takanishi, A., and Santos-Victor, J. (2016). On stereo confidence measures for global methods: evaluation, new model and integration into occupancy grids. *IEEE Trans. Pattern Anal. Mach. Intell*, **38**(1), 116–128.

Calakli, F. and Taubin, G. (2011). Ssd: Smooth signed distance surface reconstruction. *Computer Graphics Forum*, **30**(7), 1993–2002.

Campbell, N., Vogiatzis, G., Hernandez, C., and Cipolla, R. (2008). Using multiple hypotheses to improve depth-maps for multi-view stereo. In *ECCV*.

Chekhlov, D., Gee, A. P., Calway, A., and Mayol-Cuevas, W. (2007). Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual slam. In *ISMAR*.

Chen, X., Xu, L., Wang, Y., Wang, H., Wang, F., Zeng, X., Wang, Q., and Egger, J. (2015). Development of a surgical navigation system based on augmented reality using an optical see-through head-mounted display. *Journal of biomedical informatics*, **55**, 124–131.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell*, **24**(5), 603–619.

Crouse, D. F., Willett, P., Pattipati, K., and Svensson, L. (2011). A look at gaussian mixture reduction algorithms. In *International Conference on Information Fusion*.

Cuccuru, G., Gobbetti, E., Marton, F., Pajarola, R., and Pintus, R. (2009). Fast low-memory streaming mls reconstruction of point-sampled surfaces. In *GI*.

Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. *ACM Trans. Graph.*, **30**, 303–312.

Ducke, B., Score, D., and Reeves, J. (2011). Multiview 3d reconstruction of the archaeological site at weymouth from image series. *Computers & Graphics*, **35**(2), 375–382.

Dunn, E. and Frahm, J. M. (2009). Next best view planning for active model improvement. In *BMVC*.

Engel, J., Schops, T., and Cremers, D. (2014). Lsd-slam: Large-scale direct monocular slam. In *ECCV*.

Fioraio, N., Taylor, J., Fitzgibbon, A., Di Stefano, L., and Izadi, S. (2015). Large-scale and drift-free surface reconstruction using online subvolume registration. In *CVPR*.

Fuhrmann, S. and Goesele, M. (2011). Fusion of depth maps with multiple scales. *ACM Trans. Graph. (Proc. ACM SIGGRAPH Asia)*, **30**.

Fuhrmann, S. and Goesele, M. (2014). Floating scale surface reconstruction. *ACM Trans. Graph.*, **33**(4), 46.

Furukawa, Y. and Ponce, J. (2010). Accurate, dense, and robust multi-view stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell*, **32**(8), 1362–1376.

Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2009). Manhattan-world stereo. In *CVPR*.

Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2010). Towards internet-scale multi-view stereo. In *CVPR*.

Gallup, D., Frahm, J. M., and Pollefeys, M. (2010). Piecewise planar and non-planar stereo for urban scene reconstruction. In *CVPR*.

Gastal, E. S. L. and Oliveira, M. M. (2011). Domain transform for edge-aware image and video processing. *ACM Trans. Graph.*, **30**(4), 69:1–69:12.

Gee, A. P., Chekhlov, D., Mayol-Cuevas, W., and Calway, A. (2007). Discovering planes and collapsing the state space in visual slam. In *BMVC*.

Goesele, M., Snavely, N., Curless, B., Hoppe, H., and Seitz, S. M. (2007). Multi-view stereo for community photo collections. In *ICCV*.

Habbecke, M. and Kobbelt, L. (2007). A surface-growing approach to multi-view stereo reconstruction. In *CVPR*.

Haner, S. and Heyden, A. (2012). Covariance propagation and next best view planning for 3d reconstruction. In *ECCV*.

Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

Hernandez, C. and Schmitt, F. (2004). Silhouette and stereo fusion for 3d object modeling. *Comput. Vis. Image Underst.*, **96**(3), 367–392.

Hoppe, C., Klopschitz, M., Rumpler, M., Wendel, A., Kluckner, S., Bischof, H., and Reitmayr, G. (2012). Online feedback for structure-from-motion image acquisition. In *BMVC*.

Hoppe, C., Klopschitz, M., Donoser, M., and Bischof, H. (2013). Incremental surface extraction from sparse structure-from-motion point clouds. In *BMVC*.

Hornung, A., Zeng, B., and Kobbelt, L. (2008). Image selection for improved multi-view stereo. In *CVPR*.

Hu, X. and Mordohai, P. (2012). Least commitment, viewpointbased, multi-view stereo. In *3DIMPVT*.

Hur, J. amd Roth, S. (2016). Joint optical flow and temporally consistent semantic segmentation. In *CVPR*.

Jadidi, H., Ravanshadnia, M., Hosseinalipour, M., and Rahmani, F. (2015). A step-by-step construction site photography procedure to enhance the efficiency of as-built data visualization: a case study. *Visualization in Engineering*, **3**(1), 1–12.

Jancosek, M. and Pajdla, T. (2011). Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR*.

Jancosek, M., Shekhovtsov, A., and Pajdla, T. (2009). Scalable multi-view stereo. In *ICCV*.

Jeschke, S., Cline, D., and Wonka, P. (2009). A gpu laplacian solver for diffusion curves and poisson image editing. *ACM Trans. Graph.*, **28**(5), 116:1–116:8.

Kang, Z. and Medioni, G. (2014). Fast dense 3d reconstruction using an adaptive multi-scale discrete-continuous variational method. In *WACV*.

Kang, Z. and Medioni, G. (2015). Progressive 3d model acquisition with a commodity hand-held camera. In *WACV*.

Kazhdan, M. and Hoppe, H. (2013). Screened poisson surface reconstruction. *ACM Trans. Graph.*, **32**(3), 29.

Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *SGP*.

Keller, M., Lefloch, D., Lambers, M., Izadi, S., Weyrich, T., and Kolb, A. (2013). Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *3DV*.

Khan, S. M., Yan, P., and Shah, M. (2007). A homographic framework for the fusion of multi-view silhouettes. In *ICCV*.

Kim, C., Zimmer, H., Pritch, Y., Sorkine-Hornung, A., and Gross, M. H. (2013). Scene reconstruction from high spatio-angular resolution light fields. *ACM Trans. Graph.*, **32**(4), 73:1–73:12.

Kolev, K., Tanskanen, P., Speciale, P., and Pollefeys, M. (2014). Turning mobile phones into 3d scanners. In *CVPR*.

Kolmogorov, V. and Zabih, R. (2002). Multi-camera scene reconstruction via graph cuts. In *ECCV*.

Kopf, J., Cohen, M., Lischinski, D., and Uyttendaele, M. (2007). Joint bilateral upsampling. *ACM Trans. Graph. (Proc. ACM SIGGRAPH)*, **26**(3).

Kuhn, A. and Mayer, H. (2015). Incremental division of very large point clouds for scalable 3d surface reconstruction. In *ICCV Workshops*.

Kuhn, A., Hirschmueller, H., and Mayer, H. (2013). Multi-resolution range data fusion for multi-view stereo reconstruction. In *GCPR*.

Kuhn, A., Hirschmller, H., Scharstein, D., and Mayer, H. (2016). A tv prior for high-quality scalable multi-view stereo reconstruction. *Int. J. Comput. Vision*.

Kundu, A., Li, Y., Dellaert, F., Li, F., and Rehg, J. M. (2014). Joint semantic segmentation and 3d reconstruction from monocular video. In *ECCV*.

Labatut, P., Pons, J., and Keriven, R. (2009). Robust and efficient surface reconstruction from range data. *Computer Graphics Forum*, **28**, 2275–2290.

Ladikos, A., Ilic, S., and Navab, N. (2009). Spectral camera clustering. In *ICCV Workshops*.

Lang, M., Wang, O., Aydin, T., Smolic, A., and Gross, M. (2012). Practical temporal consistency for image-based graphics applications. *ACM Trans. Graph.*, **31**(4), 34:1–34:8.

Laurentini, A. (1994). The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Anal. Mach. Intell*, **16**(2), 150–162.

Li, H. and Flierl, M. (2012). Sift-based multi-view cooperative tracking for soccer video. In *ICASSP*.

Li, J., Li, E., Chen, Y., and Xu, L. (2010). Bundled depth-map merging for multi-view stereo. In *CVPR*.

Liu, Y., Jang, Y., Woo, W., and Kim, T. K. (2014). Video-based object recognition using novel set-of-sets representations. In *CVPR workshops*.

Locher, A., Perdoch, M., and Van Gool, L. (2016). Progressive prioritized multi-view stereo. In *CVPR*.

Ma, Z., Jorge, R. N., Mascarenhas, T., and Tavares, J. M. R. (2013). A level set based algorithm to reconstruct the urinary bladder from multiple views. *Medical engineering & physics*, **35**(12), 1819–1824.

Martinez-Carranza, J. and Calway, A. (2010). Unifying planar and point mapping in monocular slam. In *BMVC*.

Mauro, M., Riemenschneider, H., Van Gool, L., Leonardi, R., and Brescia, I. (2013). Overlapping camera clustering through dominant sets for scalable 3d reconstruction. In *BMVC*.

Mauro, M., Riemenschneider, H., Signoroni, A., Leonardi, R., and Van Gool, L. (2014a). An integer linear programming model for view selection on overlapping camera clusters. In *3DV*.

Mauro, M., Riemenschneider, H., Signoroni, A., Leonardi, R., Van Gool, L., and Brescia, I. (2014b). A unified framework for content-aware view selection and planning through view importance. In *BMVC*.

McNames, J. (2001). A fast nearest-neighbor algorithm based on a principal axis search tree. *IEEE Trans. Pattern Anal. Mach. Intell*, **23**(9), 964–976.

Mendez, O., Hadfield, S., Pugeault, N., and Bowden, R. (2016). Next-best stereo: Extending next-best view optimisation for collaborative sensors. In *BMVC*.

Merrell, P., Akbarzadeh, A., Wang, L., Mordohai, P., Frahm, J. M., Yang, R., Nistér, D., and Pollefeys, M. (2007). Real-time visibility-based fusion of depth maps. In *ICCV*.

Mount, D. M. and Arya, S. (1998). Ann: library for approximate nearest neighbour searching. Technical report.

Múcke, P., Klowsky, R., and Goesele, M. (2011). Surface reconstruction from multi-resolution sample points. In *VMV*.

Mustafa, W., Pugeault, N., and Kruger, N. (2013). Multi-view object recognition using view-point invariant shape relations and appearance information. In *ICRA*.

Narayanan, P., Rander, P., and Kanade, T. (1998). Constructing virtual worlds using dense stereo. In *ICCV*.

Newcombe, R. A. and Davison, A. J. (2010). Live dense reconstruction with a single moving camera. In *CVPR*.

Nießner, M., Zollhöfer, M., Izadi, S., and Stamminger, M. (2013). Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. Graph.*, **32**(6), 169.

Ohtake, Y., Belyaev, A., Alexa, M., Turk, G., and Seidel, H. P. (2003). Multi-level partition of unity implicits. *ACM Trans. Graph. (Proc. ACM SIGGRAPH)*, **22**(3), 463–470.

Osher, S. and Sethian, J. A. (1988). Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of Computational physics*, **79**(1), 12–49.

Oxholm, G. and Nishino, K. (2014). Multiview shape and reflectance from natural illumination. In *CVPR*.

Pan, Q., Reitmayr, G., and Drummond, T. (2009). Proforma: Probabilistic feature-based on-line rapid model acquisition. In *BMVC*.

Pizzoli, M., Forster, C., and Scaramuzza, D. (2014). Remode: Probabilistic, monocular dense reconstruction in real time. In *ICRA*.

Pollefeys, M., Koch, R., Vergauwen, M., and Van Gool, L. (1998). Metric 3d surface reconstruction from uncalibrated image sequences. In *SMILE Workshop*.

Pollefeys, M., Van Gool, L., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., and Koch, R. (2004). Visual modeling with a hand-held camera. *Int. J. Comput. Vision*, **59**(3), 207–232.

Pollefeys, M., Nießner, D., Frahm, J. M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S. J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewnius, H., Yang, R., Welch, G., and Towles, H. (2008). Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, **78**(2), 143167.

Reichl, F., Weiss, J., and Westermann, R. (2015). Memoryefficient interactive online reconstruction from depth image streams. *Computer Graphics Forum*, **34**(2), 1–13.

Resch, B., Lensch, H. P. A., Wang, O., Pollefeys, M., and Solkine-Hornung, A. (2015). Scalable structure from motion for densely sampled videos. In *CVPR*.

Riemenschneider, H., Bódis-Szomorú, A., Weissenberg, J., and Van Gool, L. (2014). Learning where to classify in multi-view semantic segmentation. In *ECCV*.

Russell, B., Sivic, J., Ponce, J., and Dessales, H. (2011). Automatic alignment of paintings and photographs depicting a 3d scene. In *ICCV Workshops*.

Rzeszutek, R. and Androutsos, D. (2013). Efficient automatic depth estimation for video. In *DSP*.

Rzeszutek, R. and Androutsos, D. (2015). A framework for estimating relative depth in video. *Comput. Vis. Image Underst.*, **133**, 15–29.

Schoenberger, J. L., Zheng, E., Frahm, J. M., and Pollefeys, M. (2016). Pixelwise view selection for unstructured multi-view stereo. In *ECCV*.

Schroers, C., Zimmer, H., Valgaerts, L., Bruhn, A., Demetz, O., and Weickert, J. (2012). Anisotropic range image integration. In *DAGM*.

Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*.

Shen, C., O'Brien, J. F., and Shewchuk, J. R. (2005). Interpolating and approximating implicit surfaces from polygon soup. *ACM Trans. Graph. (Proc. ACM SIGGRAPH)*, pages 896–904.

Sinha, S. N., Steedly, D., and Szeliski, R. (2009). Piecewise planar stereo for image-based rendering. In *ICCV*.

Snavely, N., Seitz, S. M., and Szeliski, R. (2006). Photo tourism: Exploring photo collections in 3d. *ACM Trans. Graph.*, pages 835–846.

Snavely, N., Seitz, S. M., and Szeliski, R. (2008). Skeletal graphs for efficient structure from motion. In *CVPR*.

Stefanoski, N., Bal, C., Lang, M., Wang, O., and Smolic, A. (2013). Depth estimation and depth enhancement by diffusion of depth features. In *ICIP*.

Strecha, C., Hansen, W. v., Van Gool, L., and Thoennessen, U. (2008). On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *CVPR*.

Sugiura, T., Torii, A., and Okutomi, M. (2013). 3d surface extraction using incremental tetrahedra carving. In *ICCV Workshops*.

Tola, E., Lepetit, V., and Fua, P. (2010). Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Trans. Pattern Anal. Mach. Intell*, **32**(5), 815–830. `http://cvlab.epfl.ch/cms/site/cvlab2/lang/en/software/daisy`.

Uslu, B., Golparvar-Fard, M., and de la Garza, J. M. (2011). Image-based 3d reconstruction and recognition for enhanced highway condition assessment. In *ASCE Intl. Workshop on Computing in Civil Engineering*.

Verhoeven, G., Doneus, M., Briese, C., and Vermeulen, F. (2012). Mapping by matching: a computer vision-based approach to fast and accurate georeferencing of archaeological aerial photographs. *Journal of Archaeological Science*, **39**(7), 2060–2070.

Vogel, C., Roth, S., and Schindler, K. (2014). View-consistent 3d scene flow estimation over multiple frames. In *ECCV*.

Vogiatzis, G., Torr, P. H., and Cipolla, R. (2005). Multi-view stereo via volumetric graph-cuts. In *CVPR*.

Vogiatzis, G., Torr, P. H. S., Seitz, S. M., and Cipolla, R. (2008). Reconstructing relief surfaces. *Image and Vision Computing*, **26**(3), 397–404.

Vu, H. H., Labatut, P., Pons, J. P., and Keriven, R. (2012). High accuracy and visibility-consistent dense multiview stereo. *IEEE Trans. Pattern Anal. Mach. Intell*, **34**(5), 889–901.

Wei, J., Resch, B., and Lensch, H. P. A. (2014). Multi-view depth map estimation with cross-view consistency. In *BMVC*.

Wei, J., Resch, B., and Lensch, H. P. A. (2016). Dense and occlusion-robust multi-view stereo for unstructured videos. In *CRV*.

Wei, J., Resch, B., and Lensch, H. P. A. (2017). Dense and scalable reconstruction from unstructured videos with occlusions. In *CVPR*. Submitted.

Weise, T., Wismer, T., Leibe, B., and Van Gool, L. (2009). In-hand scanning with online loop closure. In *ICCV Workshops*.

Weise, T., Wismer, T., Leibe, B., and Van Gool, L. (2011). Online loop closure for real-time interactive 3d scanning. *Comput. Vis. Image Underst.*, **115**(5), 635–648.

Xiang, Y., Song, C., Mottaghi, R., and Savarese, S. (2014). Monocular multiview object tracking with 3d aspect parts. In *ECCV*.

Yang, M. D., Chao, C. F., Huang, K. S., Lu, L. Y., and Chen, Y. P. (2013). Image-based 3d scene reconstruction and exploration in augmented reality. *Automation in Construction*, **33**, 48–60.

Ye, G., Garces, E., Liu, Y., Dai, Q., and Gutierrez, D. (2014). Intrinsic video and applications. *ACM Trans. Graph.*, **33**(4), 80.

Yu, F. and Gallup, D. (2014). 3d reconstruction from accidental motion. In *CVPR*.

Yu, S. and Lhuillier, M. (2012). Incremental reconstruction of manifold surface from sparse visual mapping. In *3DIMPVT*.

Zach, C. (2008). Fast and high quality fusion of depth maps. In *3DPVT*.

Zaharescu, A., Cagniart, C., Ilic, S., Boyer, E., and Horaud, R. (2008). Camera-clustering for multi-resolution 3-d surface reconstruction. In *M2SFA2*.

Zhang, G., Jia, J., Wong, T. T., and Bao, H. (2009). Consistent depth maps recovery from a video sequence. *IEEE Trans. Pattern Anal. Mach. Intell*, **31**(6), 974–988.

Zheng, E., Dunn, E., Raguram, R., and Frahm, J. M. (2012). Efficient and scalable depthmap fusion. In *BMVC*.

Zhou, Z., Wu, Z., and Tan, P. (2013). Multi-view photometric stereo with spatially varying isotropic materials. In *CVPR*.