Model-based Optical Flow: Layers, Learning, and Geometry

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Dipl.-Ing. Jonas Wulff

aus Warburg (Westfalen)

Tübingen

2017

Tag der mündlichen Qualifikation:	13.4.2018
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Michael J. Black, PhD
2. Berichterstatter:	Prof. Dr.-Ing. Hendrik Lensch
3. Berichterstatter:	Prof. Dr. rer. nat. Daniel Cremers

For Lea

Abstract

The estimation of motion in video sequences establishes temporal correspondences between pixels and surfaces and allows reasoning about a scene using multiple frames. Despite being a focus of research for over three decades, computing motion, or optical flow, remains challenging due to a number of difficulties, including the treatment of motion discontinuities and occluded regions, and the integration of information from more than two frames. One reason for these issues is that most optical flow algorithms only reason about the motion of pixels on the image plane, while not taking the image formation pipeline or the 3D structure of the world into account. One approach to address this uses layered models, which represent the occlusion structure of a scene and provide an approximation to the geometry. The goal of this dissertation is to show ways to inject additional knowledge about the scene into layered methods, making them more robust, faster, and more accurate.

First, this thesis demonstrates the modeling power of layers using the example of motion blur in videos, which is caused by fast motion relative to the exposure time of the camera. Layers segment the scene into regions that move coherently while preserving their occlusion relationships. The motion of each layer therefore directly determines its motion blur. At the same time, the layered model captures complex blur overlap effects at motion discontinuities. Using layers, we can thus formulate a generative model for blurred video sequences, and use this model to simultaneously deblur a video and compute accurate optical flow for highly dynamic scenes containing motion blur.

Next, we consider the representation of the motion within layers. Since, in a layered model, important motion discontinuities are captured by the segmentation into layers, the flow within each layer varies smoothly and can be approximated using a low dimensional subspace. We show how this subspace can be learned from training data using principal component analysis (PCA), and that flow estimation using this subspace is computationally efficient. The combination of the layered model and the low-dimensional subspace gives the best of both worlds, sharp motion discontinuities from the layers and computational efficiency from the subspace.

Lastly, we show how layered methods can be dramatically improved using simple semantics. Instead of treating all layers equally, a semantic segmentation divides the scene into its static parts and moving objects. Static parts of the scene constitute a large majority of what is shown in typical video sequences; yet, in such regions optical flow is fully constrained by the depth structure of the scene and the camera motion. After segmenting out moving objects, we consider only static regions, and

explicitly reason about the structure of the scene and the camera motion, yielding much better optical flow estimates. Furthermore, computing the structure of the scene allows to better combine information from multiple frames, resulting in high accuracies even in occluded regions. For moving regions, we compute the flow using a generic optical flow method, and combine it with the flow computed for the static regions to obtain a full optical flow field.

By combining layered models of the scene with reasoning about the dynamic behavior of the real, three-dimensional world, the methods presented herein push the envelope of optical flow computation in terms of robustness, speed, and accuracy, giving state-of-the-art results on benchmarks and pointing to important future research directions for the estimation of motion in natural scenes.

Kurzfassung

Bewegungsschätzung in Videos etabliert zeitliche Korrespondenzen zwischen Pixeln und Segmenten und ermöglicht es, mehrere Bilder zur Interpretation der Szene zu verwenden. Berechnung der Bewegung bzw. des sogenannten Optischen Flusses ist seit drei Jahrzehnten ein Schwerpunkt der Forschung im Bereich Bildverstehen. Problematisch sind hierbei beispielsweise die Berechnung der Bewegung an Objektgrenzen und in verdeckten Regionen sowieso die Kombination von Informationen aus mehr als zwei Frames. Bestehende Algorithmen zur Bewegungsschätzung betrachten nur die Bewegung der Pixel und vernachlässigen optische Eigenschaften des Kamerasystems und die dreidimensionale Struktur der Welt. Ein Ansatz ist, die Szene als eine Sammlung von Bildebenen zu betrachten. Dies stellt eine Approximation der 3D Geometrie dar und modelliert implizit Verdeckungseffekte. Die vorliegende Arbeit zeigt auf, wie weitere statistische Eigenschaften der Szene in solche auf Bildebenen basierenden Szenenmodelle integriert werden können, und wie dies die Robustheit, Effizienz und Genauigkeit der Algorithmen erhöht.

In dieser Arbeit wird zunächst gezeigt, wie Bildebenen helfen, Videos mit Bewegungsunschärfe zu analysieren. Bewegungsunschärfe entsteht bei schnellen Bewegungen während der Öffnung der Kamerablende. Bildebenen unterteilen ein Bild in Regionen mit unterschiedlichen Bewegungen; die Bewegungsunschärfe in jedem Segment kann daher über seine Bewegung ausgedrückt werden. Gleichzeitig erhalten Bildebenen Informationen über Verdeckungen und können daher Unschärfeeffekte an Objektkanten modellieren. Mittels Bildebenen kann daher ein generatives Modell für Videos mit Bewegungsunschärfe formuliert werden. Dieses Modell kann wiederum verwendet werden, um Bewegungsunschärfe zu entfernen und korrekten optischen Fluss zu berechnen.

Anschließend behandelt diese Arbeit die Bewegungsrepräsentation innerhalb der Bildebenen. Bei Verwendung von Bildebenen werden Diskontinuitäten bereits durch die Segmentierung modelliert. Der optische Fluss innerhalb einer Bildebene variiert daher nur langsam und kann als in einem niederdimensionalen Unterraum liegend approximiert werden. Dieser Unterraum kann mittels Hauptkomponentenanalyse gelernt werden und erlaubt eine effiziente Berechnung des Flusses, während die Segmentierung in Bildebenen für scharfe Diskontinuitäten sorgt.

Abschließend wird gezeigt, wie Bildebenen mit einfachen Semantiken kombiniert werden können. Anstatt alle Bildebenen gleich zu behandeln, unterteilt eine semantische Segmentierung die Szene in statische Regionen und Objekte in Bewegung. Statische Regionen stellen einen Großteil des Bildes in normalen Videosequencen

dar. Gleichzeitig ist der Fluss in diesen Regionen beschränkt und nur bestimmt durch die 3D-Struktur der Szene und die Bewegung der Kamera. Nach der Segmentierung der sich bewegenden Objekte können die Geometrie der Szene und die Kamerabewegung explizit berechnet werden. Dies führt zu Fluss mit deutlich höherer Genauigkeit. Darüberhinaus ermöglicht die explizite Repräsentation der Struktur der Szene, Informationen aus mehreren Frames zu kombinieren, was zu besseren Bewegungsschätzungen insbesondere in verdeckten Regionen der Szene führt. Ein vollständiges Bewegungsfeld wird dadurch erreicht, dass der so berechnete Fluss für statische Regionen mit dem Fluss für Objekte in Bewegung kombiniert wird, berechnet mit einem allgemeinen Algorithmus zur Bewegungsschätzung.

Die in dieser Arbeit beschriebenen Algorithmen kombinieren auf Bildebenen basierende Methoden mit Annahmen über und Eigenschaften der dreidimensionalen Welt. Sie verbessern den aktuellen Stand der Forschung im Hinblick auf Stabilität, Geschwindigkeit, und Genauigkeit, liefern gute Ergebnisse auf Vergleichsdatensätzen, und zeigen wichtige zukünftige Forschungsansätze zur Bewegungsschätzung in alltäglichen Szenen auf.

Acknowledgements

First of all, I would like to thank my advisor, Michael J. Black. Despite the monumental task of building up a new institute, he took time to take me on as a student. He taught and guided me through this experience with incredible knowledge, dedication, and never diminishing optimism in all aspects of research and life. A PhD experience is usually not called enjoyable and fun, but thanks to Michael, mine was. I also thank Prof. Daniel Cremers for providing an additional review of this thesis and the members of my committee, Prof. Hendrik Lensch, Prof. Matthias Bethge, and Prof. Andreas Geiger especially Hendrik for his willingness to take on one MPI student after another.

Melanie Feldhofer cannot be thanked enough for everything she has done (too much to list). There would be no, or certainly no functioning, PS without her. Rocko, who keeps proving himself more than capable of being an emotional support dog during each conference deadline.

Over the years, many people came to the institute, and many people left. They were colleagues, they are friends, and I am grateful for them all. Laura Sevilla-Lara, for being a great and free-wheeling mind, and always being up for the next craziness, be it in travels or in research. Ali Osman Ulusoy, for always being willing to discuss Geometry and everything else in and out of the lab. Siyu Tang, for support during the final months of thesis writing, when the air got somewhat thin. Eric Rachlin, Jessica Purmont and Javier Romero for starting the social life when the department was still young, Gerard Pons-Moll for engaging in discussion about Physics despite total absence of knowledge on all sides, and Sergey Prokudin and Angjoo Kanazawa for many great book recommendations that I am still trying to catch up on. Martin Kiefel for his German humor and neverending help with programming questions. Naureen Mahmood for a fun race to the top of coffee mountain and Jon Anning, who kept everything running as long as possible, and provided an incredible IT infrastructure for all of us.

I thank Deqing Sun for paving the way in layered optical flow and teaching me matrix calculus, Matt Loper for the fantastic Chumpy library, and Søren Hauberg for his robust PCA code. Christoph Lassner, Thomas Nestmeyer, Anurag Ranjan and Varun Jampani for patiently answering questions about CNNs, Andreas Geiger for commiserating about the joys and pains of dataset generation and maintenance, and Joel Janai and Fatma Güney for making flow slow.

Between 2012 and 2014, I spent several months as a guest at the Massachusetts Institute of Technology, and was always fortunate to be tolerated there. For this

I thank Prof. Pawan Sinha, Prof. Edward Adelson, and Prof. Antonio Torralba. Yuri Ostrovsky, Bei Xiao, and Lavanya Sharan were my points of contacts and helped me settle and connect; they and Philip Isola, Carl Vondrick, and Amy Kalia were fantastic office mates and friends.

In 2014, I spent a wonderful summer as a research intern at Adobe Research in Seattle. For this opportunity I am grateful to Eli Shechtman, Hailin Jin, and Scott Cohen.

I cannot sufficiently express my gratitude to my parents, Claudia Wulff and Michael Schimanski-Wulff. Their breadth of knowledge and interests formed me, and their unwavering support and encouragement helped me in countless situations. I can only hope to be a similar parent to my own daughter as you were to me.

Finally, my wonderful wife Sandra. You were with me from the start, and always questioned, showed me different perspectives, and made sure I never got lost in the strange world of research. I could not have done this without you.

Funding for this thesis was generously provided through a PhD fellowship from the Max Planck Society, as well as a fellowship from the Max-Planck / ETH Zürich Center for Learning Systems.

Contents

Chapter 1

Introduction

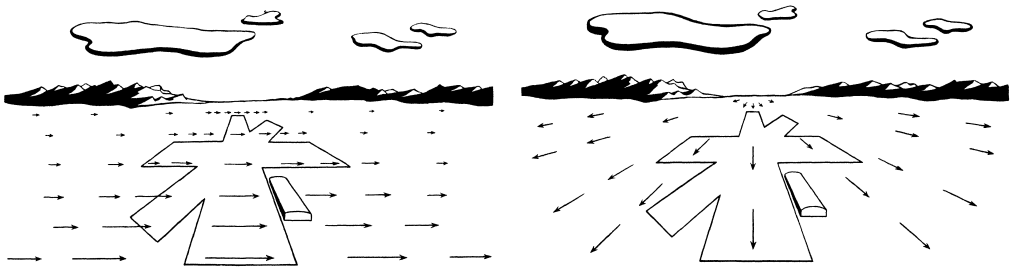
From the moment we first open our eyes as a newborn, we see a dynamic world, full of motion and change. We learn to move and act in this world, while constantly shifting our viewpoint. Objects and entities independent from us move by themselves or through the application of physical forces, and the environment itself causes changes in visual appearance, such as shifts of illumination and environmental effects like fog and clouds. All these effects cause the light patterns that hit the photoreceptors in our retina to vary throughout the day, exposing us to a wide range of different visual inputs.

Yet, despite this constantly changing input to our visual system, we perceive the world as stable and coherent in time. Objects are still or move as a whole, the environment is usually seen as static and non-moving, and when a surface becomes occluded behind something else, we do not perceive it as suddenly ceasing to exist. Obviously, we perceive change and dynamic properties of the world. Yet, the rate of change is very limited, leading to our perception of the world as temporally smooth and continuous[1].

A key mechanism for this temporally consistent perception of the world is establishing correspondences over time, that is to determine which part of an object at a point in time corresponds to which part of the same object at a different time. Given a temporally varying stimulus (such as a video), temporal correspondences allow us to perceive objects as coherent despite their changing appearance. Consider, for example, a car on a racetrack. The appearance of this car can change dramatically, as the car as seen from the front looks very different from the same car as seen from the side. Yet, since temporal correspondences tell us which parts of the car in the front view correspond to which parts in the side view, we are able to bind the car together and perceive it as a whole, solid object.

Establishing these links is a fundamental skill in the world; without them, every change in appearance would make an object seem new and unrelated to the same object seen at an earlier point in time. Whenever we moved our heads, we would be surprised about all the new objects that just appeared on our desks.

[1]Patients suffering from Akinetopsia do in fact perceive the world in stroboscopic instances, leading to great difficulties in movement, performing daily tasks, or following conversations.

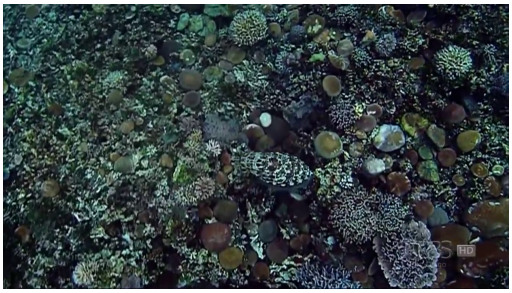


(a) Sideways motion (b) Expanding motion

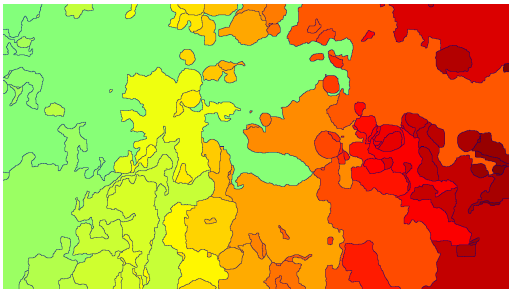
Figure 1.1: The pattern of the motion field, indicated by arrows, provides a strong cue to the motion of the observer. Images taken from [95].

However, purely "stabilizing" the world that we perceive is not the sole purpose of the computation of temporal correspondences. If this were so, motion could just be treated as noise, and the ultimate goal would be to remove it. This is not the case. To the contrary, motion (*i.e.* the established correspondences) contains a tremendous amount of information which can be used for a multitude of purposes. One such example is the computation of egomotion, that is, the computation of the motion of the observer in the world (whether she turns, moves forward, etc.). Egomotion causes distinct "motion patterns" in the visual input (see Fig. 1.1), which are used to detect heading direction in humans [95] as well as control heading in lower animals such as flies [77, 98]. Another example is so-called parallax, which refers to the difference in motion of two surfaces due to a difference in depth. Everybody who has ever looked out of a train window is aware of this effect, where the horizon appears stationary, while trees close to the train fly by, making them almost impossible to track visually. In this case, faster objects are perceived to be closer. Hence, motion can provide useful information not just about the motion of the observer, but also about the three-dimensional structure of the scene.

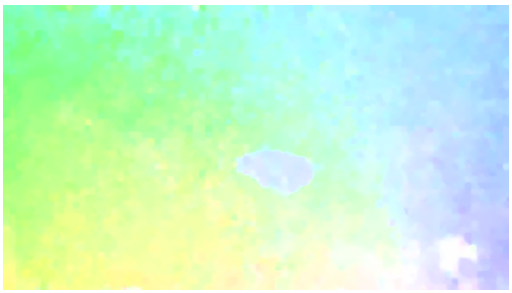
Motion also provides information about the segmentation of the scene into individual objects. In Gestalt psychology, this is often referred to as the principle of *common fate*, and can be simply described as "What moves together, belongs together". The principle of common fate is especially interesting when considering a scene containing heavily textured objects. Due to complex intensity patterns and strong edges within textures, such objects are often hard to segment from each other and the rest of the scene, which can lead to oversegmentation (objects are split into multiple parts) or undersegmentation (objects are not properly separated from the background). Figure 1.2 shows such an example. A camouflaged Cephalopod takes on a pattern that resembles the pattern of the background, making it very challenging for a human observer to detect in a static image (Fig. 1.2a). Sim-



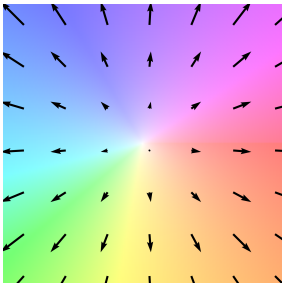
(a) First of two frames.



(b) Image segmentation.



(c) Computed motion field



(d) Motion visualization

Figure 1.2: Motion provides a strong cue to object segmentation. (a) When considering just an image, an object (in this case, a camouflaged Cephalopod) can be hard to detect. (b) Automatic image segmentation algorithms similarly fail in this case. (c) Motion provides a strong segmentation cue, since in the motion field the object is clearly visible. (d) For visualization purposes, the motion is coded with the hue denoting the direction and the saturation the magnitude of the motion.

ilarly to our perception, even state-of-the-art image segmentation algorithms[2] fail miserably (Fig. 1.2b). When computing the motion, however, the animal is clearly visible, and the camouflage becomes useless[3] (Fig. 1.2c). In fact, in human vision, some studies with subjects gaining visual sight for the first time have indicated that motion is one of the first cues to allow us to parse the world into objects, and further segmentation cues (such as boundary smoothness or color similarity) might be learned from the segmentation provided by motion [187, 190].

As these examples show, motion provides a crucial cue to the configuration of the surfaces of the world around us, about the presence and spatial extent of moving objects, and about the observer's location and motion within this world[4]. This is

[2]Here, we use globalPb [13] with scale parameter $s = 1.5$.

[3]Because of this, animals in full camouflage are usually extremely still [69].

[4]Note that the applications for motion estimations go even further and also include domains related to signal processing, such as denoising and temporal interpolation, and higher level applications such as action classification.

essential information for biological and artificial systems that move and act within the world and interact with other entities. In the quest of building perceiving machines, accurate motion estimation is hence a critical component, and the focus of this thesis.

The approach we take here is based on an important observation. Above, we argued that motion contains important information about the spatio-temporal configuration of the world. While this argument is intuitive from the point of view of an artificial system that has to reason about the world, it obstructs the actual direction of causality. The perceived motion contains the information about the configuration of the world only *because it is directly determined by this configuration.* This leads us to an important tenet: *Motion cannot be arbitrary.* Instead, there exist strong links between the world, its physics and the behavior of the observer on the one side, and the temporal stimulus perceived by a system on the other side. This strong connection makes it possible, as described above, to infer the properties of the world from a temporal stimulus, but at the same time allows us to impose constraints on the motion. Specifically, motion should be caused by a possible and likely spatio-temporal configuration of the world.

To follow this route of constrained motion estimation, in this thesis we employ the framework of layered methods for optical flow. By modeling a scene as a set of overlapping "cutouts", or layers, these methods provide a first approximation to the three-dimensional geometry of the world. Beyond the layered model itself, however, they still employ heuristics, for example in the permissible motion of each layer, or in the mechanisms used to segment the scene into layers in the first place. This thesis investigates ways to upgrade layered scene models for optical flow computation with more physically grounded mechanisms, thereby constraining the space of possible optical flow fields and increasing the accuracy of the estimation, while at the same time allowing extraction of richer information about the scene than a simple cutout-based representation.

1.1 Representing and computing motion

In artificial applications, motion is typically represented as the two-dimensional motion field, or *optical flow*[5]. It can be represented as a 2D function $\mathbf{u} : \mathbb{R}^2 \mapsto \mathbb{R}^2$, mapping a continuous location $\mathbf{x}$ to a two-dimensional displacement value $\mathbf{u}(\mathbf{x}) \in \mathbb{R}^2$. In practice, the optical flow field $\mathbf{u}$ is represented at a 2D vector at each pixel

[5]Here, we need to make an important distinction between two modalities of motion. On the one hand, *motion* can refer to the projection of the three-dimensional motion of an object onto the image plane ("projected scene motion"), which is independent from illumination effects. On the other hand, it can also refer to the motion of image appearance ("image motion"), which for example tracks the motion of a specular highlight across the surface of a non-moving object. In practice, we are almost always care about the first, and hence use the term optical flow to refer to the projected scene motion.

location $\mathbf{x}$, from which the values at fractional locations are interpolated [229]. For integer pixel locations, $\mathbf{u}(\mathbf{x})$ can hence be intuitively interpreted as "the motion that the pixel at $\mathbf{x}$ undergoes". The task of computing the motion is usually understood as computing the optical flow $\mathbf{u}$ between two temporally successive frames I_t and I_{t+1}.

Local motion computation

The most intuitive approach to compute $\mathbf{u}$ is to simply search, at each location $\mathbf{x}$, for the motion $\mathbf{u}(\mathbf{x})$ so that $I_{t+1}(\mathbf{x}+\mathbf{u}(\mathbf{x}))$ is most similar (as measured over a neighborhood) to $I_t(\mathbf{x})$. Formally,

$$\hat{\mathbf{u}}(\mathbf{x}) = \arg\min_{\mathbf{u}} \texttt{err}\left(N(\mathbf{x}, I_t), N(\mathbf{x}+\mathbf{u}(\mathbf{x}), I_{t+1})\right), \tag{1.1}$$

where $N(\mathbf{x}, I)$ denotes a patch around $\mathbf{x}$ extracted from I,

$$N(\mathbf{x}, I) = \{I(\mathbf{y}) | \mathbf{y} \in \mathcal{N}(\mathbf{x}) \cup \{\mathbf{x}\}\}, \tag{1.2}$$

and $\mathcal{N}(\mathbf{x})$ the set of neighborhood locations around $\mathbf{x}$. The error function `err` measures the dissimilarity between two of those patches and can take multiple forms, for example the L^1 or L^2 norm, a robust metric [39], the normalized cross-correlation, or an explicit feature transformation and matching (using, for example, a convolutional neural network [101]). For a small neighborhood size, the solution of Eq. (1.1) at one location is independent from the solution at another location in the same image, and can be computed locally. Methods solving Eq. (1.1) are therefore referred to as *local methods*. In salient image regions such as corners or finely textured surfaces, local methods yield good results. Furthermore, since they usually match a whole image patch, they are fairly robust to camera noise [53]. However, local methods suffer from the so-called *aperture problem* [12], illustrated in Fig. 1.3. To successfully compute matching candidates for a patch, the image information within this patch has to vary in two spatial directions, such as in Fig. 1.3, green circle. Patches that do not contain enough variation in the image values are hard or impossible to match. If for example a patch is situated on a straight edge, it matches all patches along the edge (Fig. 1.3, blue circle). Even worse, if a patch is situated within an unstructured region, it matches all patches with the same background color (Fig. 1.3, red circles)[6]. Local methods therefore compute accurate flow only at a few limited locations in the image, corresponding

[6]The mathematical argument is that each pixel within a patch only provides information about the motion along the gradient direction at this pixel. If a patch contains only an edge, all the nonzero gradients point to the same direction, and hence only the motion orthogonal to the edge can be computed. Therefore, to locate a patch in the 2D pixel grid, non-parallel image gradients need to be present within the patch.

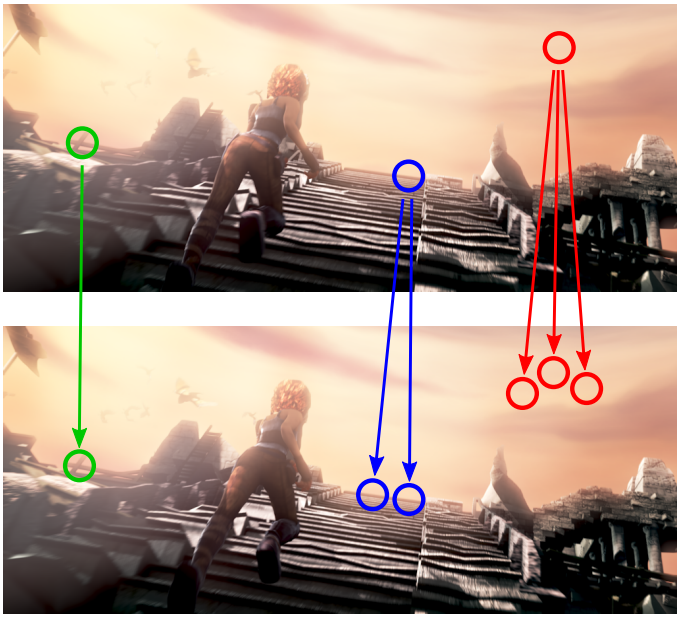


Figure 1.3: Illustration of the aperture problem. Patches with sufficient structure (green) can be matched locally, while matching of patches with structure in a single direction (blue) or without any structure (red) is ambiguous.

to highly structured image patches, and not across the full image plane.

Global motion computation

To obtain a *dense* optical flow field (*i.e.*, a 2D motion vector at each pixel), one can add a regularization or smoothness term. The flow can then be computed as the solution to a constrained minimization problem,

$$\hat{\mathbf{u}} = \arg\min_{\mathbf{u}} E_{data}\left(\mathbf{u}, I_t, I_{t+1}\right) + E_{reg}\left(\mathbf{u}, I_t\right), \tag{1.3}$$

where E_{data} is a matching energy, indicating how well $\mathbf{u}$ maps I_{t+1} back onto I_t, and E_{reg} is a regularization energy, imposing some form of (potentially image-modulated) spatial smoothness on $\mathbf{u}$. A classical example are simple quadratic terms [116],

$$E_{data}\left(\mathbf{u}, I_t, I_{t+1}\right) = \sum_{\mathbf{x}} \left\| N(\mathbf{x}, I_t) - N(\mathbf{x} + \mathbf{u}(\mathbf{x}), I_{t+1})) \right\|^2 \tag{1.4}$$

$$E_{reg}\left(\mathbf{u}\right) = \lambda \sum_{\mathbf{x}} \left\| \nabla u_1(\mathbf{x}) \right\|^2 + \left\| \nabla u_2(\mathbf{x}) \right\|^2. \tag{1.5}$$

This encourages the computed optical flow field to vary smoothly. Motion information is hence propagated into regions for which the image structure itself is not rich enough for matching. Note that in classical global methods, the size of the patch N is often set to 1×1 pixel, *i.e.* the pixel values are compared directly.

In the presence of motion boundaries, however, the enforced smoothness presents a problem. If two adjacent pixels belong to different objects (for example, one pixel to a moving car and an adjacent pixel to the static background), the true motion vectors of these pixels are uncorrelated and can differ by a large amount. Computing the flow while enforcing a smoothness constraint thus results in a flow field that does not accurately represent motion boundaries. Instead, motion boundaries in the flow field will be blurry, yielding wrong results around objects.

Approaches such as robust error metrics [40] and image-guided regularization [229, 196] can alleviate this problem by allowing flow discontinuities, either without constraints, or only when flow discontinuities coincide with edges in the image. While such extensions can dramatically improve optical flow computation, they still fail in challenging scenarios, for example in the presence of motion blur, or if two neighboring but disconnected surfaces have a similar appearance. Ultimately, these methods are heuristic: they reason about the structure of the optical flow on the level of pixels, ignoring the fact that optical flow is the projection of motion in the three-dimensional, structured world.

An additional problem of both local and global methods is that of unmatched regions. Since both methods measure the image similarity under a given flow estimation, they can at best make an educated guess about the motion in areas that are visible in I_t but not in I_{t+1}. Such areas usually correspond to surfaces that either become occluded by other surfaces or that leave the visible frame. Due to the smoothness constraint, global methods can interpolate into these regions; however, the interpolated value are purely based on the closest visible pixels, and not based on any actual image content at the corresponding locations.

Layers

To overcome these problems, several works have proposed to model the scene as a composition of layers [228, 245, 260]; see Fig. 1.4 for an illustration. In a layer-based representation, the scene is approximated as a set of overlapping segments ordered in depth. If the segmentation into layers is known, the regularization term can be adjusted so as to not cross layer boundaries, thereby preventing blurring. Additionally, if more than two frames are available, occluded regions can be assigned to the correct layer, avoiding wrong flow interpolation[7] into these regions [232]. Lastly, layers can be used for methods beyond flow computation, for example for the treatment of motion blur across object boundaries (see Chapter 3).

[7] This restricts the source of interpolation to the correct layer; however, in common layered methods, the flow in occlusions is still only interpolated and not measured.

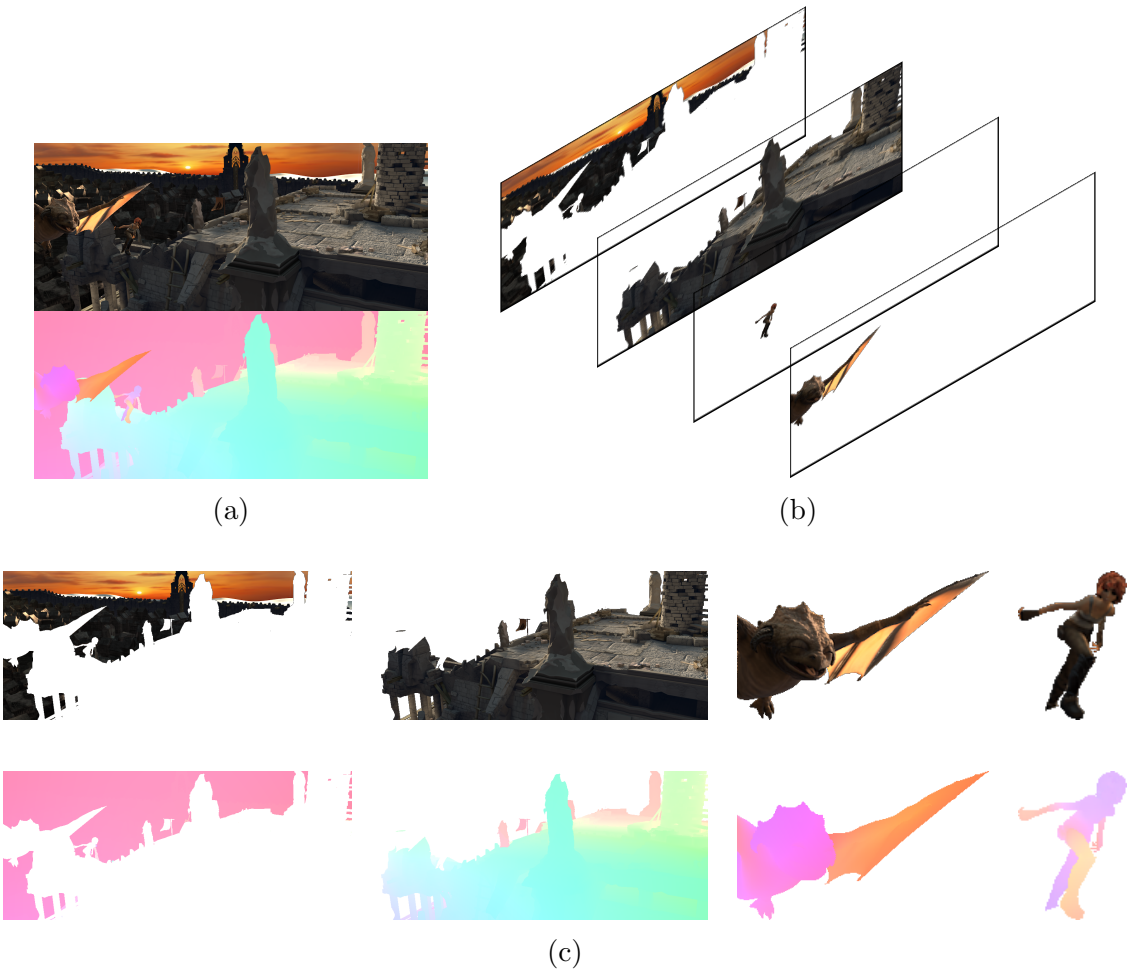


Figure 1.4: A layered representation approximates a scene (a) through a set of overlapping segments (b). Within each layer, the flow varies mostly smoothly, since most motion discontinuities are captured by the layer boundaries (c).

Early layered methods assumed only parametric motion of each layer, that is, each layer was undergoing a very simple motion such as a translation, a purely affine motion, or a planar perspective motion modelled as a homography. While such models can already significantly improve motion accuracy at object boundaries, the problem is that each layer is effectively seen as a flat, "cardboard"-like object. Typical scenes, however, contain a large number of planar surfaces. Treating such scenes in a parametric layer-based framework thus requires a very large number of layers, which quickly becomes computationally prohibitive. Additionally, non-planar surfaces are hard to model and have to be approximated. Lastly, all layered methods impose a strict global depth ordering on the layers. However,

in realistic, complex scenes, such an ordering might not hold (for example due to self-occlusions), or might only be valid locally [228].

An alternative to assigning a layer to each planar surfaces is to allow each layer to deviate from the parametric motion, which is often called "smoothness in layers" [228, 268]. These methods divide the scene into a relatively low number of layers, and within each layer use a standard robust regularization, often only regularizing the deviation from a parametric motion. These approaches can thus be seen as global methods within each layer. While these methods can capture motion boundaries and accomodate non-planar structures even within a layer, they still suffer from similar problems as global methods, namely that the regularizations are based on heuristics, and that they only interpolate the motion into occluded regions. Furthermore, such methods effectively solve one full optical flow problem for each layer, and in addition have to compute the layer assignment (which pixel belongs to which layer) and the depth ordering between the layers. This makes layered methods computationally inefficient, to the point that some methods take up to 22 hours to compute flow for a sequence of 5 frames [228].

What would be an appropriate way to address these issues? For this, let us recapitulate the basic assumption of layers: They correspond to meaningful segments in the world. Such segments, however, obey certain structural constraints, such as temporal consistency: they do not suddenly appear or disappear, approximately maintain their volume in 3D, and move according to the laws of physics such as the conservation of energy and momentum; in short, the motion of a layered scene is, again, limited to possible spatio-temporal configurations of the world. Layers provide an approximation of the layout of the scene; yet, the scene is still bound by what physics allows. This brings us to the central hypothesis of this work.

1.2 Main hypothesis

As we pointed out above, layered optical flow methods provide a good model for the overall structure and composition of the scene. Yet, many aspects of layered methods, for example the methods used to regularize the flow *within* a layer, are still fairly simplistic and based on heuristics, and ignore the fact that optical flow is the projection of a process happening in the real world. The main hypothesis of this thesis is that introducing principled assumptions based on the image formation process and geometrical considerations into the estimation of layered optical flow can benefit the estimation process and increase accuracy and computational efficiency by bringing layered flow estimation closer to modeling the spatio-temporal properties of the three-dimensional world.

1.3 Approach

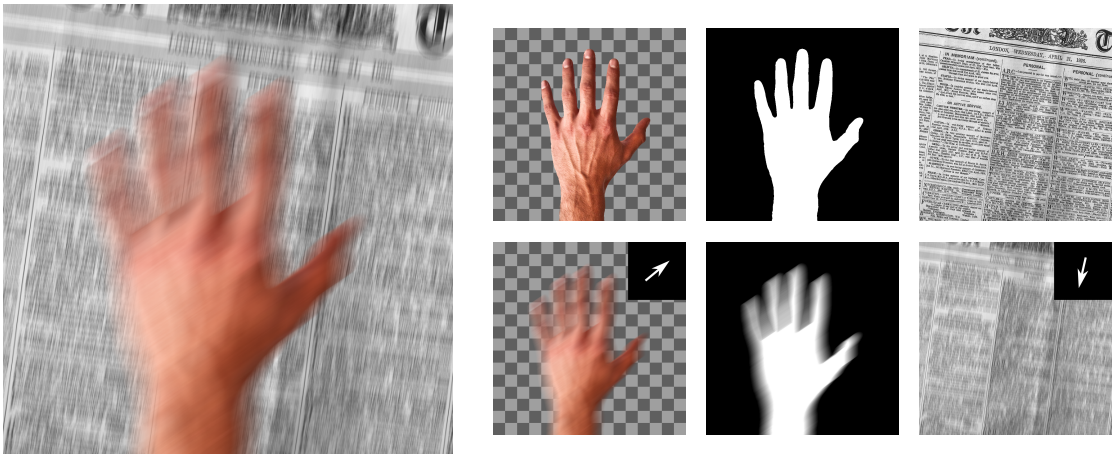
To investigate our hypothesis, we extend layered models in three, physically plausible ways. First, we model the presence of complex motion blur as a by-product of image formation in dynamic scenes containing moving objects. Second, we note that a layer usually corresponds to a single coherent object. The flow within a layer can thus be assumed to be fairly low-dimensional; we use this insight to learn and restrict the flow statistics within layers from a large body of data. Third, we note that realistic, dynamic scenes consist of large regions of static background and few moving objects. By explicitly segmenting the scene into these types of regions, we can use much stronger constraints than previously possible in large amounts of the frame.

1.3.1 Motion blur and layers

The first question is whether layers are a useful representation for the computation of optical flow. Despite its intuitive appeal, most methods dominating current optical flow benchmarks do not employ a layered model of the scene, but instead rely on advanced feature matching and image-aware regularization. We therefore begin by demonstrating the usefulness of a layered model of the scene in a specific scenario, namely, in videos that contain motion blur. Processing such videos poses two main difficulties. First, it is hard to remove the motion blur, since the combination of blurred moving objects with background blur caused by camera motion creates blur patterns that are spatially varying and, near motion boundaries, the results of an interplay of background and foreground blur (see Fig. 1.5). As we will show, conventional deblurring methods fail in these cases. Second, the computation of optical flow itself is hard. As shown above, optical flow algorithms rely on a measure of photometric consistency. In the presence of motion blur, the appearance of a point can change dramatically from one frame to the next, especially at motion boundaries, where the appearance of a pixel is a mixture of the background and foreground appearances.

Using a layered framework helps us to overcome these problems, since it allows us to formulate a generative model of a multi-layered scene containing motion blur. The key insight here is that the blur is fully determined by the motion of the layers. Hence, the free parameters in our model are the motion of the layers, the unblurred layer appearance, and the layer segmentation. We can compute initializations for all of these parameters, and then refine them using our generative model of the scene. This yields both optical flow as well as the deblurred layer appearance; at the time of publication, the results in both modalities surpassed specialized algorithms for each of the two.

This example shows the main reason for the usefulness of layers: They can serve as an abstraction and approximation of the scene. This allows them to capture



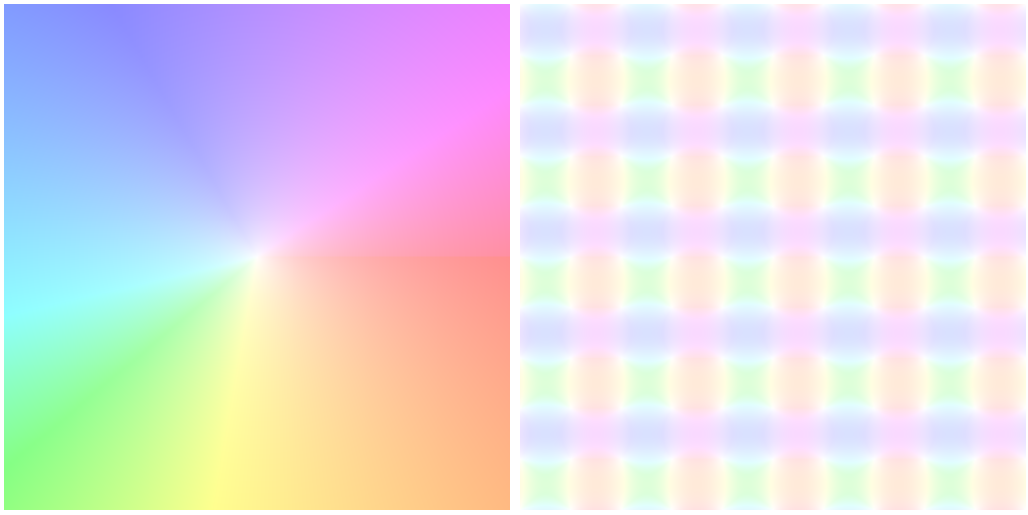
(a) Blurred image (b) Foreground (c) Segmentation (d) Background

Figure 1.5: Using a layered model, a blurred image (a) can be expressed as a composite of a blurred foreground (b) over a blurred background (d). The segmentation into foreground and background is shown in (c). For (b-d), the top row shows the sharp images, and the bottom row shows the images after being blurred due to motion blur.

important properties about the image formation process such as the behavior of motion blur at occluded regions and near depth discontinuities. Modeling this behavior directly improves results on the tasks at hand.

1.3.2 Learning the statistics of flow within layers

Due to the nature of layers, their modeling power is most visible at object boundaries and occlusions. In contrast, the optical flow *within* the extent of a layer rarely benefits from adopting a layered method. Instead, within a layer, a local, pairwise regularization is usually used, similar to what is used in non-layered optical flow. Such a local regularizer only reasons about the structure of the optical flow field within a small spatial extent, for example by encouraging the flow at neighboring pixels to be similar. Due to the limited spatial extent that it considers, such a regularizer often fails to capture the global structure of an optical flow field. A simple example is given in Figure 1.6. Figure 1.6(a) shows an optical flow field as it would typically occur if the camera moves towards a planar surface parallel to the image plane, which is a very common and realistic scenario, for example for a robot moving towards a wall. Figure 1.6(b), in contrast, shows an artificially constructed optical flow field containing sinusoidal motion patterns in both horizontal and vertical directions. Encountering such a motion pattern in reality is highly unlikely. However, when computing the regularization energy simply based on the smoothness within a local neighborhood (as it is done in both layered and



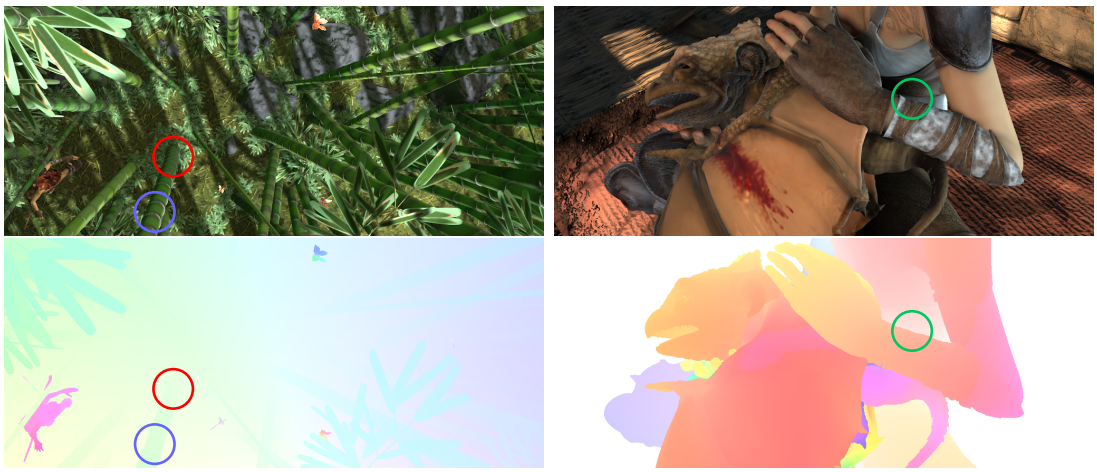
(a) Realistic motion. $E_{reg} = 18608$. (b) Artificial motion. $E_{reg} = 15477$.

Figure 1.6: A purely local regularization does not always capture the physical plausibility of an optical flow field. Instead, the regularization energy of a plausible flow field (a) can be higher than that of a purely artificial and implausible flow field (b).

traditional optical flow methods), the energy of (a) is higher (18608) than that of (b) (15477). In the absence of further information, a purely local regularizer would hence prefer the unrealistic motion pattern (b) over the physically plausible motion pattern (a).

To adress this issue, we learn a *global* regularizer for optical flow. As a training dataset, we use optical flow fields computed offline from four live-action feature films spanning different scenarios, scenes, and genres. We then perform Principal Component Analysis (PCA) on these optical flow fields, which yields a low-dimensional linear subspace that captures as much information about the global structure of optical flow as possible. Each principal direction in this subspace corresponds to a "basis flow field", and any optical flow field can be approximated as a point in this subspace, *i.e.* as a weighted combination of a limited set of basis flow fields. The computation of a new optical flow field is therefore simplified to computing the weights of the basis flow fields, that is, the coordinates in the flow subspace. This greatly reduces the required computational cost to compute optical flow.

Since we limit the dimenisonality of the subspace, however, the resulting flow fields only approximate the true flow fields. More specifically, they lack high frequency information and hence appear blurry. While the resulting oversmooth optical flow fields might be sufficient for some applications such as egomotion estimation or coarse scene classification, sharp motion boundaries are an important piece of information. To remedy the lack of high-frequency information, we turn back to



(a) Contact surfaces (b) Self-occlusions

Figure 1.7: Examples for challenging layered flow estimation. (a) Depth discontinuities can cause motion discontinuities (blue circle); when the depth difference becomes zero at the contact point (red circle), the motion discontinuities disappear. (b) Self-occlusions create motion discontinuities (green circle) between two surfaces that semantically belong to the same object.

a layer-based framework. The intuitive reasoning here is that the flow within a layer usually varies smoothly, while strong motion discontinuities correspond to object boundaries. Hence, we segment the scene into a set of layers, and use our PCA-based regularizer to only regularize the flow *within* each layer. This prevents oversmoothing across object boundaries and at the same time keeps the required computation time low.

1.3.3 Static and moving regions

So far, we have seen that the main benefits of layers are the treatment of occlusions and motion boundaries, as well as an implicit segmentation of the scene into object-like surfaces. To achieve these benefits, layered scene models approximate the scene as a set of overlapping cutouts. Taking one step back, we notice that such a representation makes two implicit assumptions. First, since the primary reason for a layered model is to capture motion discontinuities, these discontinuities are expected to be stronger between than within layers. Second, all layers are qualitatively equal, in that the local structure that is imposed on the motion of each layer is the same for all layers. Upon closer investigation, both assumptions fail to hold. First, motion discontinuities often do not obey the "cutout" model, as illustrated in Fig. 1.7. For example, at the top of a cylinder standing on a flat surface there

exists a significant depth discontinuity with the background which, in the case of egomotion, creates a motion discontinuity. At the contact point of the cylinder, however, no depth discontinuity and hence no motion discontinuity exists, and it is not obvious what a good segmentation would be in this case (Fig. 1.7a). Another example where the cutout model fails are self-occlusions, such as a person's arm in front of her body. Here, a significant motion discontinuity exists, but semantics and intuition dictate that the arm and the body should be grouped together (Fig. 1.7b).

The second assumption is that of qualitatively equal layers. This ignores the everyday experience that most natural scenes consist of a few dynamic objects acting within a large static scene. This suggests a qualitative difference between at least two types of layers: a large background on the one hand and small objects on the other hand.

Building on these ideas, we therefore adjust the concept of layers. Instead of segmenting the scene into a set of layers that are treated equally, we now divide the scene into two main parts: regions that are static, and regions that move independently[8]. Moving segments can correspond to rigidly moving objects, such as cars, but also to deforming objects such as humans or animals. Ideally, one would use different models for the motion of such objects, based on the semantic class; however, this goes beyond the scope of this thesis. Instead, within the moving regions, we use a standard optical flow algorithm employing an image-guided regularizer [196]. In contrast to moving regions, the motion of the static background is highly constrained: within the static scene, the observed motion at each pixel is only caused by the motion of the camera, and the magnitude and direction of the observed motion only depend on the motion of the camera and the distance of the corresponding 3D point to the camera. One advantage of this is that, when estimating the flow, the total number of variables is greatly reduced: instead of estimating a two-dimensional motion vector at each pixel (a problem with $2 \times H \times W$ unknowns for an image of size $H \times W$), we now require the computation of only $6 + H \times W$ variables, 6 for the camera rotation and translation, and $H \times W$ for a single depth value per pixel.

Additionally, this changed concept of layers makes the segmentation problem easier. In the common layered framework, segmentation amounts to a chicken-and-egg problem. Motion and segmentation depend on each other, and hence usually have to be computed iteratively. In our framework, however, moving regions usually correspond to objects in the world. Such objects, cars, people, planes, animals, etc., all belong to well-defined semantic categories. Recent advances in algorithms for semantic segmentation of images allow us to compute segmentations of images into such objects from the images alone before reasoning about any motion, removing

[8]Note that this is again merely an approximation of the real world, since on an atomic level, everything moves. However, for all practical purposes, this approximation can be considered sufficient.

the need for an iterative approach. When computing the motion, it becomes now possible to use appropriate constraints on the optical flow in both types of regions, static and moving.

A third advantage of this formulation is that information can be easily accumulated over time. Recall that the motion at each point within the static parts of the scene only depends on the camera motion and the depth (*i.e.* the distance to the observer) of this point. Hence, both can be treated as latent variables, and computed explicitly. The motion of the camera is time-dependent, *i.e.* it changes from frame to frame. On the other hand, the depth at a pixel that belongs to the static scene is indepenent of time within a given reference frame. Consider, for example, the flow from frame I_t to I_{t+1} and from I_t to I_{t-1} induced by a moving camera in a static scene. Each of the two flow fields can be expressed as the combination of a camera motion and a depth map. While the camera motions are different, in theory the depth maps at reference time t have to be equal, since the surfaces do not move by themselves. These different estimates can be combined to obtain a better estimate of the depth, and hence the flow, filling in occluded regions and pixels that leave the visible frame.

To summarize, this thesis extends classical layer-based approaches to optical flow in three ways. First, it shows how to employ a layered scene model to improve motion and appearance estimation in the presence of motion blur. Second, it proposes a low-dimensional, global regularizer for the flow of each layer, exploiting the fact that the optical flow belonging to a single object usually varies smoothly. Lastly, it shows how semantic information can be used to improve a layered world model by splitting the image into parts that are static and parts that move, and demonstrates how to incorporate strong, geometry-based constraints into the estimation of the motion of the static part of the scene. These strong constraints are hence used wherever possible, leading to a regularization that adapts to the motion characteristics of each layer.

1.4 Organization

The remainder of this thesis is organized as follows. In **Chapter 2**, we review previous work on optical flow. First, a general overview of past and current approaches and trends is given. This is followed by a description of different regularizers of optical flow, a description of layered and segmentation-based optical flow methods, and an overview of attempts to include the depth structure of the scene into motion computation.

Chapter 3 shows how physical knowledge can be included in a layered scene model. Using a parametric two-layer model, we show how even such a simple model can be useful in reasoning about the physical processes involved in the emergence of complex motion blur. The resulting combination of layers with a model of the image

formation process allows estimation of both motion and deblurred appearance in complex scenes containing moving objects and occlusions.

Classical layered models mainly improve the results in areas near depth discontinuities, which are modelled as layer boundaries, and largely ignore the motion within layers. In **Chapter 4**, we address this issue by learning the global structure of optical flow fields from data. This yields a global, low-dimensional representation of optical flow, which can be used to efficiently compute an approximate, smooth optical flow field. Since this representation cannot capture sharp motion boundaries, we show how such a low-dimensional representation can be embedded in a layered framework. This takes the best from both worlds: Layers are used to model the sharp motion boundaries between objects, while the fast, approximate flow is used to globally regularize the flow within each layer.

One drawback of this method, however, is that layers themselves are only an approximation, and often do not faithfully capture important aspects of the geometry of the world. **Chapter 5** therefore proposes a modification of the standard layered flow estimation. By allowing qualitative differences in the motion of different types of segments, the scene can be divided into moving objects and the static background, and strong constraints can be used in the later case, allowing for better modeling of intra-layer motion discontinuities and occlusions, and temporal integration of motion information across multiple frames. This modification yields a segmentation of the scene into moving segments and the static background, the depth structure of the static background, and accurate optical flow.

Finally, **Chapter 6** summarizes and concludes the thesis, and points out directions of future work.

Chapter 2

Background and related work

Motion has been recognized as a fundamental mode of perception for a long time, with its study going back at least to Helmholtz [258] and Wertheimer and the Gestalt theorists establishing the connections between motion and perceptual grouping [272, 273]. Even the problem of motion *computation*, that is, the extraction of motion from a given image sequence, has been studied for almost four decades [251].

As expected, the literature is therefore vast, and providing a comprehensive review of the field of optical flow and its computation goes beyond the scope of even a thesis. Interested readers are referred to a number of comprehensive surveys [27, 23, 84]. In particular, the investigation of different optimization algorithms will be omitted here.

Instead, I will provide a taxonomy of methods, oriented on a model for the recording of a video sequence as outlined in Fig. 2.1. The source of all perceptual phenomena is the world, that is, the scene, objects, and their actions and interactions within the scene and on a continuous timescale. This world world is recorded and discretely sampled, producing a set of frames, *i.e.* two-dimensional projections of the world. The temporal evolution of these frames is modeled by the optical flow field. An optical flow method can incorporate assumptions about each of these three large pieces, the world, the recording process, and the structure of the two-dimensional optical flow field. The taxonomy I propose here classifies optical flow methods by the location of their assumptions. We classify optical flow methods as either *constraining the motion field*, *constraining the observation process*, or *constraining the geometry*. The vast majority of optical flow methods belong to the first category. Within each class, we furthermore subdivide methods by the complexity of their assumptions. In methods imposing constraints on the motion field, for example, we see that there exists a natural ordering of complexity, starting from simple parametric motion models (such as a global translation) and increasing complexity up to methods reasoning about the motion of multiple overlapping objects and their temporal characteristics.

It should be noted that in many cases the constraints described in the following are *soft constraints*. Often, they do not impose strict boundaries on which flow fields are possible; rather, they encourage an algorithm to prefer certain flow fields

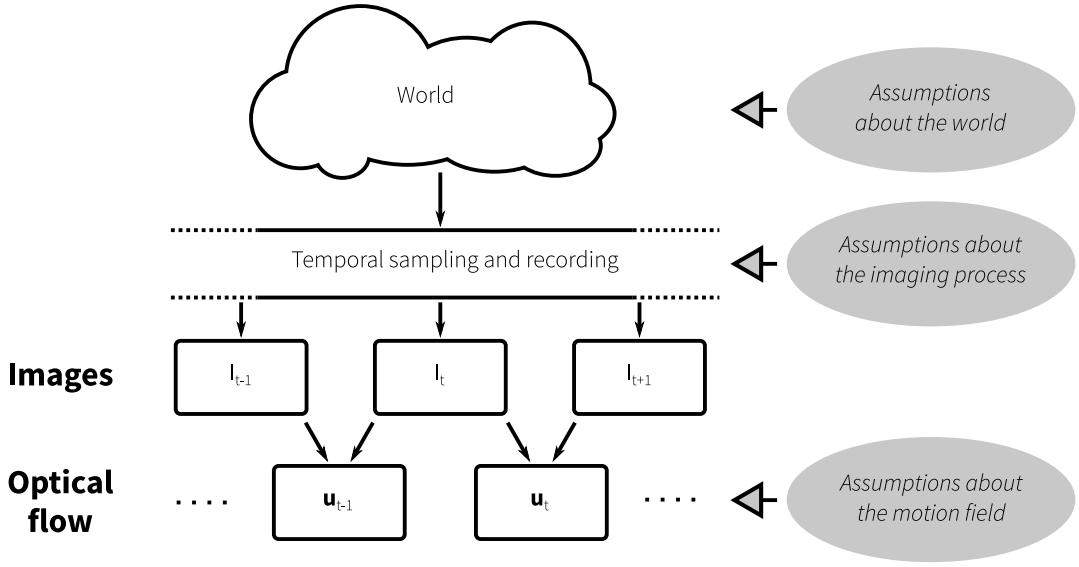


Figure 2.1: Overview of the imaging process. The continuous, three-dimensional world is temporally sampled and recorded using an imaging process, producing the recorded frames. The optical flow is a representation of the motion between these frames. Assumptions can be made at any point in the process, such as assumptions about the structure of the world, about the imaging process, or about the two-dimensional optical flow field.

over others. Under a penalty, however, flow fields violating the constraints are still possible. From a probabilistic perspective, they function as a prior on the structure of the optical flow field.

2.1 Constraints on the motion field

The majority of works in the field of optical flow explicitly reason about the structure of the motion field, that is, the pixel representation of the projected motion of the three-dimensional scene. In this case, the optical flow uses an image-like representation: It is defined on the same domain as the image and is sampled at the same sites. One can therefore think of the flow as a two-channel image, with the channels denoting the motion in horizontal and vertical direction, respectively. Thinking about the flow in this way has a number of advantages. The most important is that it is very intuitive. In this framework, the optical flow at a given pixel can be simply interpreted as "where does this pixel move". Therefore, no reasoning in three dimensions is necessary, and all the computation takes place on the image plane. A second advantage is that such a representation is convenient to

handle, since existing data structures and algorithms for handling images can often be re-used to handle optical flow. After all, each optical flow value can be simply thought of as a pixel. For these reasons, treating the optical flow as an image has long been the dominant paradigm of computation.

Two notes should be made here. First, two main representations of the flow on the image plane exist, either as a discrete set of pixels, or as a functional (in so-called variational methods). The difference is mostly theoretical, however. Variational methods treat the optical flow as a continuous function that maps a two-dimensional input $\mathbf{x}$ to a two-dimensional output $\mathbf{u}(\mathbf{x})$. This function can be found using calculus of variations. This mainly helps with the derivation of the optimization procedure; when actually optimizing, all operators as well as the final results are discretized (since, as mentioned above, the flow is represented as the motion vectors at a fixed, grid-structured set of sites). The alternative method is to consider the optical flow as discrete from the start, and derive the necessary optimization steps using matrix calculus. In practice both approaches often lead to very similar algorithms [23].

The second consideration is that the notion of "reasoning on the image plane" is a slight oversimplification. The optical flow on the image plane is a projection of the three-dimensional motion of the world relative to the camera. Hence, all constraints on this projected optical flow field *implicitly* impose some constraints on the three-dimensional motion of the world. Explicitly establishing links between constraints on the projected motion field and the world motion, however, can be difficult and is rarely done. In the taxonomy we use here, we consider *explicitly formulated* constraints, and the stage of the projection process (see Fig. 2.1) that is subject to those constraints.

Also, the reader should keep in mind that the transitions between different types of constraints and assumptions are not always clear-cut. To give two examples, Ju *et al.* [135] model the optical flow field as a set of moving patches (as described in Sec. 2.1.1), but allow small pixel-wise deviations from the patch model (Sec. 2.1.2). Nir *et al.* [182] use a pairwise, variational framework, *i.e.* they only consider the structure of the flow of neighboring pixels. The motion models that they assume for each pixel, however, are full homographies, making stronger assumptions about the motion of the scene (Sec. 2.1.3). Here, we attempt to categorize previous works by what the general assumption *about the world* is, and in the upcoming sections place them accordingly.

2.1.1 Parametric motion

The simplest way to represent motion is to express it by a few values or parameters. These could for example include global shifts in horizontal and vertical direction in case of a purely translational motion or a rotation around the line of sight. Similarly, the motion can be parameterized in the three dimensional space, for

example through a 3D translational motion and the roll, pitch, and yaw angles for a rotation. Hence, the motion of a large portion of the frame is expressed using relatively few numbers.

Mathematically, one can define a *transformation function* $\mathbf{w} : \mathbb{R}^2 \mapsto \mathbb{R}^2$ controlled by a set of parameters $\boldsymbol{\theta}$. For each point $\mathbf{x}$, the location in the next frame is then given as

$$\mathbf{x}' = \mathbf{w}(\mathbf{x}, \boldsymbol{\theta}) \tag{2.1}$$

and the flow is subsequently given as

$$\mathbf{u}(\mathbf{x}) = \mathbf{x}' - \mathbf{x} = \mathbf{w}(\mathbf{x}, \boldsymbol{\theta}) - \mathbf{x}. \tag{2.2}$$

Simple examples for such transformations corresponding to geometric transformations on the image plane include

Translation: $$\mathbf{w}(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{x} + \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \tag{2.3}$$

Affine: $$\mathbf{w}(\mathbf{x}, \boldsymbol{\theta}) = \begin{pmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & \theta_5 & \theta_6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} \tag{2.4}$$

Planar-projective: $$\mathbf{w}(\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{\theta_7 x_1 + \theta_8 x_2 + \theta_9} \begin{pmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & \theta_5 & \theta_6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} \tag{2.5}$$

After defining a proper parameterization, the question then becomes how to actually compute the parameters given two frames. We want to find a set of parameters $\boldsymbol{\theta}$ so that

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{\mathbf{x}} \rho\left(I_1(\mathbf{x}) - I_2(\mathbf{w}(\mathbf{x}, \boldsymbol{\theta}))\right). \tag{2.6}$$

As before, $\rho(z)$ denotes a robust error function.

The simplest way is to use an exhaustive search across a discretized set of values for $\boldsymbol{\theta}$, and use the $\boldsymbol{\theta}$ for which the sum in Eq. (2.6) is minimized. However, this is only feasible if (a) the image resolution is small or the maximal range of the parameters severely restricted, and (b) the dimensionality of $\boldsymbol{\theta}$ is low. This excludes motion that is more complicated than a simple translation; for example, exhaustively searching in the 8-dimensional space of the parameters of a homography is infeasible. Additionally, the possible values that $\boldsymbol{\theta}$ can take on is limited to the discrete search space defined a-priori, limiting the accuracy. For these reasons, dense search is not common.

An alternative, originally proposed by Lucas and Kanade [160] is to assume that both images are already in a somewhat close alignment. Their algorithms

incrementally solve for a series of parameter increments $\Delta\boldsymbol{\theta}^{(k)}$, where k denotes the iteration number, and update the estimated parameter $\boldsymbol{\theta}^{(k)}$ after each step:

$$\Delta\boldsymbol{\theta}^{(k+1)} = \arg\min_{\Delta\boldsymbol{\theta}} \sum_{\mathbf{x}} \rho\left(I_1\left(\mathbf{x}\right) - I_2\left(\mathbf{w}\left(\mathbf{x}, \boldsymbol{\theta}^{(k)} + \Delta\boldsymbol{\theta}\right)\right)\right) \tag{2.7}$$

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \Delta\boldsymbol{\theta}^{(k+1)}. \tag{2.8}$$

In [160], a quadratic error norm $\rho(z) = z^2$ is used, and, after linearizing I_2 around $\boldsymbol{\theta}^{(k)}$, the solution to Eq. (2.7) can be computed in closed form. Intuitively, this approach warps I_2 at each iteration, computes the parameter increment $\Delta\boldsymbol{\theta}$, and updates the warping parameters. Since their linearization is only valid for small motions, they propose to use a multi-resolution framework, in which the motion of a blurry (and/or downsampled) version of the images is computed first, and used to initialize the motion estimation at finer scales.

Black [35] shows how the quadratic error can be replaced by a robust error function $\rho(z)$ to account for occlusions, and Hager and Belhumeur [103] extend the parameters to lighting variations and hand-crafted object-specific deformations, and introduce the idea of computing linearization on I_1 instead of the warped I_2, resulting in significant computational speed-ups. Shum and Szeliski [222] propose a theoretically slightly different approach that updates the parameters not through addition, but through composition. In compositional approaches, the warped pixel coordinates in iteration (k), $\mathbf{x}'^{(k)}$ are given as $\mathbf{x}'^{(k)} = \mathbf{w}\left(\mathbf{w}\left(\mathbf{x}, \Delta\boldsymbol{\theta}^{(k)}\right), \boldsymbol{\theta}^{(k)}\right)$ instead of $\mathbf{x}'^{(k)} = \mathbf{w}\left(\mathbf{x}, \boldsymbol{\theta}^{(k)} + \Delta\boldsymbol{\theta}^{(k)}\right)$ as in the case for additive approaches like [160]. Since they allow more pre-computation, compositional approaches are computationally much more efficient; in terms of the computed results, they are almost equivalent [22].

One common use case for global parametric motion estimation are graphics applications, where it can be used to stabilize the camera or stitch together multiple images to create panoramas [222, 237]. In such applications, the main focus is on analyzing or removing the camera motion, which is assumed to primarily consist of a rotational component. Since parallax is only an effect of translational camera motion, discontinuities in the flow field are expected to be minimal, and a simple global parametric model captures the information necessary for the tasks at hand.

When computing an optical flow field, however, one is usually interested in extracting more fine-grained information about the scene such as motion boundaries, the three-dimensional layout of the scene, or object motion. A motion representation must be expressive enough to capture this type of information; hence, a global full-frame parametric motion estimation might not be particularly useful.

One possibility is to use parametric models in smaller regions of the frame, for example in patches or around objects [16, 103, 136, 160]. Such methods improve upon global parametric estimation, in that they are capable to model motion coming from different sources, such as object motion or different motion magnitudes due

to parallax. However, they come with two main issues. First, being local optical flow methods (see Chapter 1.1), they suffer from the aperture effect and can only compute accurate optical flow in regions with sufficient non-parallel structure. This makes the computation of optical flow in large regions with low visual structure (such as a white wall) difficult and often unreliable. Second, they pool motion information across an image region of non-trivial size. Thus the motion of small structures, for example a vertical pole making up only a fraction of the pooling region cannot be detected. Furthermore, if an object moves and deforms at the same time (such as walking human), a simple a-priori motion parameterization is often not enough [30, 127].

Learned motion models

It is therefore often advantageous to not define the motion model by hand but to learn it. Such motion models keep the good computational properties of parametric models while being able to capture as much or as little relevant motion detail as determined in the learning phase. Typically, a learned motion parameterization takes on form of a set of learned basis flow fields, which span a linear subspace of possible motions. Let $\{\mathbf{b}_1 \dots \mathbf{b}_B\}$ be a set of B pre-computed optical flow fields serving as the basis of the space of permissible motions. The final motion field can then be computed as

$$\mathbf{u} = \sum_{i=1}^{B} \theta_i \mathbf{b}_i \tag{2.9}$$

and the parameters $\boldsymbol{\theta} = \{\theta_1, \dots, \theta_B\}$ correspond to the weights of each basis flow field. The basis flow fields $\mathbf{b}_i$ are usually computed using a Principal Component Analysis (PCA).

Typical examples for such custom flow spaces are the analysis and computation of motion of deformable objects such as faces and full humans. They fit this paradigm well: On one hand, they undergo severe deformations, making their motion hard to capture. On the other hand, due to the underlying skeletal structure, the deformation is structured and inherently low-dimensional. Black *et al.* [43] uses custom, but pre-defined parameterized models of the motion of faces and connect the parameters with facial expressions, making it possible to detect sentiments and moods from video. Fleet *et al.* [83] extend this by learning a motion subspace for facial motion, instead of manually defining it.

Several authors have explored PCA models of walking humans [80, 102]. Fleet *et al.* [83] use a learned motion model to detect walking humans in video and analyze their gate. Similarly, Guthier *et al.* [102] learn larger optical flow bases, but restrict themselves to walking humans. They use the resulting optical flow bases for analysis only, and not to estimate new optical flow fields.

Learned parametric motion models can also be used to learn other, inherently

low-dimensional motion patterns, such as ego-motion, which can theoretically be expressed using six degrees of freedom. Roberts *et al.* [198] learn an optical flow subspace for ego-motion using probabilistic PCA. Using this subspace, they estimate a dense flow field from sparse motion estimates. They restrict themselves to ego-motion, train and test on similar sequences captured from the same platform, and use only a two-dimensional subspace with a low resolution of 45×13 pixels. Recent work extends this, but focuses on semantic classification into obstacle classes from this motion subspace, rather than accurate motion estimation [197].

Lastly, when used in small patches, learned motion models can be used to detect and analyze local phenomena that are hard to describe analytically. Fleet *et al.* [83] learn a parametric motion model to detect motion boundaries and occlusions, which are hard to capture using conventional parametric models. Chessa *et al.* [63] use basis flow fields in local patches to account for affine motion plus deformation due to the geometric structure of the scene.

What all parametric methods have in common is that they parameterize motion using a low-dimensional model. Therefore, they inherently make strong assumptions about the content, structure, and motion of the scene. These assumptions are either made a-priori, or rely on an underlying low-dimensional structure of the motion of specific objects and limit the motion estimation to these objects. Capturing the richness and variability of natural scenes is therefore challenging for parametric methods.

2.1.2 First-order structure

A more general approach is to compute the full optical flow field, that is, estimate the horizontal and vertical velocities $\mathbf{u}(\mathbf{x}) = (u_1(\mathbf{x}), u_2(\mathbf{x}))$ at each pixel location $\mathbf{x}$. This corresponds to the global methods mentioned in 1.1. Compared to parametric motion models, this can be seen as the limit case of estimating a 2D translational motion at each patch with a patch size of 1×1 pixel. Even without further consideration it is obvious that this problem is ill-posed, since the task is to estimate $2 \times W \times H$ flow values based on $W \times H$ measurements. For such global methods, it is therefore necessary to impose additional constraints or priors on the optical flow fields; the form and expressiveness of these priors is the topic of the next few sections.

The simplest prior to impose on the optical flow field is that of first-order spatial smoothness. Under this prior, neighboring pixels are expected to have similar optical flow. To reproduce the example given in 1.1, the energy function that is minimized in this case becomes

$$\hat{\mathbf{u}} = \arg\min_{\mathbf{u}} E_{data} + \lambda E_{reg}, \tag{2.10}$$

with

$$E_{data} = \sum_{\mathbf{x}} \rho_{data} \left(I_t \left(\mathbf{x} \right) - I_{t+1} \left(\mathbf{x} + \mathbf{u}(\mathbf{x}) \right) \right) \tag{2.11}$$

$$E_{reg} = \lambda \sum_{\mathbf{x}} \rho_{reg} \left(\sqrt{\|\nabla u_1(\mathbf{x})\|^2 + \|\nabla u_2(\mathbf{x})\|^2} \right). \tag{2.12}$$

Here, ρ_{data} and ρ_{reg} are penalty functions. In the simplest and earliest formulation of the optical flow problem, they are assumed to be quadratic, *i.e.* $\rho_{data}(z) = \rho_{reg}(z) = z^2$ [116].

The common way to solve (2.10) is to employ a variational framework, in which the horizontal and vertical components of the flow u_1, u_2 are seen as continuous functions. The energy (2.10) is then seen as a functional to be minimized on the image domain Ω with respect to u_1 and u_2.

$$\begin{aligned} E_V \left[u_1, u_2 \right] &= \int_{\Omega} E \left(x_1, x_2, u_1, u_2, u_{1,1}, u_{1,2}, u_{2,1}, u_{2,2} \right) \, \mathrm{d}\Omega \\ &= \int_{\Omega} \rho_{data} \left(I_t(\mathbf{x}) - I_{t+1}(\mathbf{x} + \mathbf{u}(\mathbf{x})) \right) \\ &\quad + \lambda \rho_{reg} \left(\sqrt{u_{1,1}(\mathbf{x})^2 + u_{1,2}(\mathbf{x})^2 + u_{2,1}(\mathbf{x})^2 + u_{2,2}(\mathbf{x})^2} \right) \mathrm{d}\Omega, \end{aligned} \tag{2.13}$$

where $u_{1,1} = \frac{\partial u_1}{\partial x_1}$, $u_{1,2} = \frac{\partial u_1}{\partial x_2}$ and so on. A solution can be found by solving the corresponding Euler-Lagrange equations

$$\begin{aligned} \frac{\partial E}{\partial u_1} - \frac{\partial}{\partial x_1} \left(\frac{\partial E}{\partial u_{1,1}} \right) - \frac{\partial}{\partial x_2} \left(\frac{\partial E}{\partial u_{1,2}} \right) = 0 \\ \frac{\partial E}{\partial u_2} - \frac{\partial}{\partial x_1} \left(\frac{\partial E}{\partial u_{2,1}} \right) - \frac{\partial}{\partial x_2} \left(\frac{\partial E}{\partial u_{2,2}} \right) = 0. \end{aligned} \tag{2.14}$$

To solve Eq. (2.14), $I_{t+1} \left(\mathbf{x} + \mathbf{u}(\mathbf{x}) \right)$ is usually linearized via a Taylor expansion, yielding the approximation [116]

$$I_{t+1} \left(\mathbf{x} + \mathbf{u}(\mathbf{x}) \right) \approx I_{t+1} \left(\mathbf{x} \right) + \frac{\partial I_{t+1}(\mathbf{x})}{\partial x_1} u_1(\mathbf{x}) + \frac{\partial I_{t+1}(\mathbf{x})}{\partial x_2} u_2(\mathbf{x}). \tag{2.15}$$

The data term then becomes

$$E_{data} \approx \sum_{\mathbf{x}} \rho_{data} \left(I_t \left(\mathbf{x} \right) - I_{t+1} \left(\mathbf{x} \right) - \frac{\partial I_{t+1}(\mathbf{x})}{\partial x_1} u_1(\mathbf{x}) - \frac{\partial I_{t+1}(\mathbf{x})}{\partial x_2} u_2(\mathbf{x}). \right), \tag{2.16}$$

yielding the well-known *Brightness Constancy Constraint (BCC)*

$$\frac{\partial I}{\partial t} + \frac{\partial I}{\partial x_1} u_1 + \frac{\partial I}{\partial x_2} u_2 = 0. \tag{2.17}$$

If the penalty functions ρ_{data} and ρ_{reg} are not quadratic, an additional linearization step is required for those [50]. Since the linearizations only hold if the flow to be computed is small, the common procedure is to iteratively solve for flow increments $\Delta u_1, \Delta u_2$, and update the current flow estimates accordingly, similar to the parametric Lucas-Kanade approach described above.

Due to the long history of optical flow computation, a large body of work exists on the optimization of Eq. (2.10). However, since detailed optimization procedures are not the focus of the thesis, the reader shall therefore be referred to [23, 84]. However, even if they use different optimization algorithms, many optical flow methods share a number of problems, since they are a direct result of modeling and reasoning about the flow on the image plane. These issues deserve a more detailed description.

Large displacements

A core issue of optical flow computation is that I_{t+1} is generally[1] not only non-linear, but non-convex in $\mathbf{u}$. As described above, the first problem is usually addressed through linearization and iterative computation. If the ground truth flow $\mathbf{u}$ is large, however, a large number of iterations is required, since each increment has to be small for the linearization to hold. This makes such an approach computationally inefficient. Furthermore, the larger $\mathbf{u}$ is, the less likely it is that the initial estimate $\mathbf{u}^{(0)}$ is in the basin of attraction of the correct solution. In such a case an iterative approach would not lead to the correct solution, but to a local minimum, which may or may not be close to the correct solution.

The classical way to compute optical flow even in the presence of large displacements is to perform the computation at multiple scales [12]. Such a scheme starts by computing the flow on the images resized to a low resolution. In such a low resolution, the small spatial extent for which the linearization is valid spans a large extent in the original image resolution; hence, fewer iterations are required to get close to the true solution. Additionally, the error surface is effectively blurred [175], helping to address the problem of small local minima. After the flow is computed, it is then scaled up to a slightly larger resolution, and used as an initialization to compute the flow at the next iteration. This process is repeated until the real resolution of the image is reached; the structure containing the images with gradually increasing resolutions is called an *image pyramid* [4]. Typical scale factors between

[1] Here we omit pathological cases such as images containing just a gradient, and only consider natural images.

the pyramid levels are either 0.5 [229] or values much closer to 1, for example 0.95 [50]. In the later case, the initialization of each scale is closer to the optimum, and hence fewer iterations are required at each scale; however, the required number of pyramid scales is obviously higher. Similarly, scale-space methods [10, 175] compute the flow on multiple scales, but treat the scale as being a continuous variable instead of separate, discrete pyramid layers.

The problem of schemes incorporating a pyramid is that, at a given pyramid level (*i.e.* at a certain downsampling factor) small image structures are blurred out. Hence, if the displacement of a small structure such as a pole is larger than its own spatial extent, the flow within such a structure usually cannot be reliably reconstructed. Sevilla *et al.* [217] decompose the image into a large number of channels, each of which is blurred separately. This retains the benefits of multi-scale computation while still allowing to reconstruct small structures; however, it comes with considerable computational cost.

Another approach is to solve for the flow not in an iterative manner, but instead incorporate a form of exhaustive search for the best displacement. Steinbrücker *et al.* [225] propose to re-cast the minimization of Eq. (2.10) as an alternating optimization problem. They introduce an auxiliarly variable $\tilde{\mathbf{u}}$, and split the optimization into a regularization on $\tilde{\mathbf{u}}$ and data term on $\mathbf{u}$. The former can be solved using efficient algorithms [56], whereas for the latter the solution at each pixel depends only on itself, and not on its neighbors. It can hence be implemented as a massively parallel exhaustive search. Methods based on nearest neighbor fields [18, 24, 108] seed the flow with some good matches, and use search heuristics to density the flow field. The flow fields can then be refined using anisotropic diffusion [159].

Cost-volume approaches go in a similar direction [60, 281]. Chen and Koltun [60] show that optical flow can be directly optimized by exploiting its regular structure; however, computing the necessary cost volume is computationally expensive. Xu *et al.* [281] show how the cost volume can be constructed efficiently, and extend semi-global matching [114] to the 2D search space. These algorithms belong to the most accurate to date.

The most common method to tackle the problem of large displacements, however, is to incorporate separately computed sparse matches into the flow computation. In their seminal work [52], Brox and Malik propose to add an additional energy E_{match} to (2.10), evaluating the consistency of the flow with the computed matches $\mathcal{M}$:

$$E_{match} = \sum_{(\mathbf{q},\mathbf{q}')\in\mathcal{M}} \rho\left(\|\mathbf{u}(\mathbf{q}) - (\mathbf{q}' - \mathbf{q})\|_2\right) \tag{2.18}$$

This soft constraint encourages the flow to agree with the previously computed sparse matches. Since the sparse matches provide a useful initial flow estimate, many fewer iterations are required. Furthermore, they often latch on to the motion of small objects, allowing them to be accurately modelled.

Similarly, many current methods use sparse features in a sequential *matching-interpolation* scheme, in which first sparse matches are established. In a second step, the flow is then densified or interpolated to yield a complete flow field, using for example variational methods such as [50]. Most work in this direction has concentrated on improving feature descriptors and feature matching algorithms. Examples for such methods are DeepFlow [267], which uses matches with a hierarchical, deformable structure, DiscreteFlow [174] which uses a global optimization over possible matches to establish better correspondences of DAISY [243] descriptors, and DeepDiscreteFlow [101] which uses a Convolutional Neural Network to transform the images into a feature space, in which the matches can be found using a simple nearest neighbor method.

Occluded regions

Another difficulty arises if a surface is occluded [241], that is, it is visible in I_t but occluded by another surface[2] in I_{t+1}. In such areas, $I_t(\mathbf{x}) \neq I_{t+1}(\mathbf{x} + \mathbf{u}(\mathbf{x}))$, even for the true motion $\mathbf{u}(\mathbf{x})$. The data term is therefore unable to provide any useful information and, in fact, often hurts the performance, since it still tries to find a $\mathbf{u}(\mathbf{x})$ for which the brightness constancy is valid. This will almost always be the wrong flow.

When using quadratic penalties [116], the data term is dominated by measurements with high error, which are usually caused by outliers, as they might appear in occluded regions. This influence of strong outliers is proportional to the value of the *influence function* $\psi(z) = \frac{\partial \rho(z)}{\partial z}$ [105], which, for a quadratic penalty, is unbounded. One common option to alleviate this is to use a robust penalty function in the data term [39]. For such a robust penalty function, $\psi(z)$ either saturates or redescends to zero, effectively reducing the influence of strong outliers. Intuitively, such functions try to correct smaller errors, such as those occurring from slight mis-estimations of the flow field, more than large errors which are caused for example by occlusions. Since such functions also increase robustness against other non-linear effects, for example specular highlights, using them has become standard practice in optical flow computation [50, 229]. Figure 2.2 shows examples of typical penalties and their influence functions.

An alternative, sometimes used in conjunction with robust error functions, is to estimate the occluded regions and deal with them explicitly. The simplest way to detect occlusions is to use the resulting image error directly, *i.e.* mark a pixel $\mathbf{x}$ as occluded if

$$\rho\left(I_t(\mathbf{x}) - I_{t+1}(\mathbf{x} + \mathbf{u}(\mathbf{x}))\right) > \tau_{image} \tag{2.19}$$

for some chosen threshold τ_{image} [226, 279]. In the next iteration, the flow can then

[2]Note that pixels leaving the frame are a special case of occlusion, and can be handled by treating the image values and the optical flow outside of the visible image as unknown.

be properly handled, for example by filling in the occluded regions using a bilateral filtering step [279]. Silva and Victor [223] mark a pixel as occluded if the minimal match in a region around $\mathbf{x}$ exceeds a threshold; this makes iterative refinement unnecessary. Ayvaci *et al.* [15] reason that the photometric error in occlusions is high, but in the whole frame occlusions are sparse. They integrate this into the objective function by imposing an L^0 prior on the occlusion map, and disable the data term in regions where an occlusion is detected.

A different approach is to detect the occlusions based on the flow itself, for example using the flow estimated in a previous iteration. A commonly used technique is to detect occlusions through violations of forward-backward consistency [8]. Let $\bar{\mathbf{u}}$ be the backward flow, *i.e.* in the ground truth case $I_t\left(\mathbf{x} + \bar{\mathbf{u}}(\mathbf{x})\right) = I_{t+1}\left(\mathbf{x}\right)$ outside of occlusions. Then, an occlusion is detected if

$$\|\mathbf{u}(\mathbf{x}) + \bar{\mathbf{u}}\left(\mathbf{x} + \mathbf{u}(\mathbf{x})\right)\| > \tau_{cons} \tag{2.20}$$

Again, this detection depends on the flow itself, and can hence only be used in an iterative framework. Bailer *et al.* [18] extend this to multiple scales, and detect an occlusion only when the forward-backward consistency is violated for backward flow of more than one scale at the same time. Black and Anandan [37] detect occlusions by "splatting", that is, they detect if a location at time $t+1$ is "hit" by multiple incoming motions; if so, it is marked as a potential occlusion boundary. Alvarez *et al.* [9] use a similar approach, but in backward direction, and define a pixel as occluded in the foreground direction if it is not "hit" (assuming appropriate interpolation) by any flow vector from $\bar{\mathbf{u}}$. Lastly, Sand and Teller [206] propose to combine photometric and flow-based occlusion estimators, and detect an occlusion where the divergence of the flow field is negative and the photometric error is high.

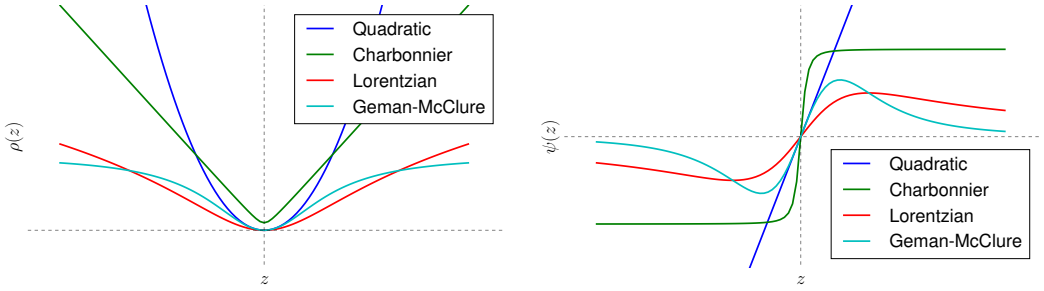
Motion boundaries and spatial smoothness

In the real world, the optical flows of both sides of an object boundary are usually independent of each other. Knowing how the surface on one side of the boundary moves tells us very little about the motion of the surface of the other side. To a lesser degree, this also holds for depth boundaries within the same object. Here, the flows are not independent, since both are caused by the camera motion; yet, the difference in depth across the boundary can result in a significant motion discontinuity due to parallax. Sharp motion discontinuities are hence an important and naturally occurring feature of optical flow fields.

However, when enforcing spatial smoothness in the classical way, *i.e.* $\rho_{reg}(z) = z^2$, large spatial flow gradients cause a disproportionate increase in energy, and are hence eliminated as much as possible. This manifests as blurring across motion boundaries, often leading to an unnatural, oversmooth appearance of the optical flow field.

Table 2.1: Examples for penalty functions

	$\rho(z)$	$\psi(z) = \frac{\partial \rho(z)}{\partial z}$
Quadratic	z^2	$2z$
Charbonnier	$\sqrt{z^2 + \epsilon}$	$\frac{z}{\sqrt{z^2+\epsilon}}$
Lorentzian	$\frac{\sigma^2}{2} \log\left(1 + \left(\frac{z}{\sigma}\right)^2\right)$	$\frac{\sigma^2 z}{\sigma^2+z^2}$
Geman-McClure	$\frac{z^2}{z^2+\sigma^2}$	$\frac{2z\sigma^2}{(z^2+\sigma^2)^2}$



(a) Penalty functions $\rho(z)$ (b) Influence functions $\psi(z)$

Figure 2.2: Examples for commonly used penalties and their influence functions. For the Charbonnier (green), $\epsilon = 0.01$. For the Lorentzian (red) and Geman-McClure (cyan) penalties, $\sigma = 1.0$. The Lorentzian and Geman-McClure penalties suppress large errors, making them robust against outliers; however, this results in non-convex energies.

Several approaches have been proposed to alleviate this problem. Similar to occlusion handling as described above, one way to sharpen the motion boundaries is to choose penalty functions other than quadratic, so called *robust functions* [39, 75, 169, 221, 290].

These functions do not explicitly extract or reason about the locations of motion boundaries but simply allow discontinuities in the motion field to appear [41]. As described above, the impact strong gradients of the flow field have on the total energy can be measured using the influence function $\psi(z) = \frac{\partial \rho}{\partial z}$. For robust functions, the influence of strong gradients (*i.e.* discontinuities) either saturates, or even decreases to zero. This can be interpreted as a weaker incentive to reduce the gradient when it is high, thereby preserving discontinuities. Table 2.1 gives examples for a few commonly used robust penalties and their influence functions,

the latter of which can be interpreted as the desire of the function to remove strong discontinuities. Figure 2.2 shows the functions in their graphical form.

Note that this demonstrates a trade-off, namely, the more robust a function, the less convex it is. Hence, the L^1 norm is a common choice, as it represents the most robust function that is still convex [290, 264], resulting in a Total Variation (TV) prior [134] on the optical flow field. In terms of discontinuity preservation, however, this choice is suboptimal, as the L^1 norm penalizes a sharp discontinuity by the same amount as a gradual transition. If more robust functions such as Geman-McClure or the Lorentzian are used, a global optimum cannot be guaranteed anymore. In this case, one possibility is to use a graduated non-convexity scheme [44, 229], in which the penalty function is a weighted sum of a quadratic penalty and a robust, non-convex penalty, and the weight of the robust penalty is gradually increased.

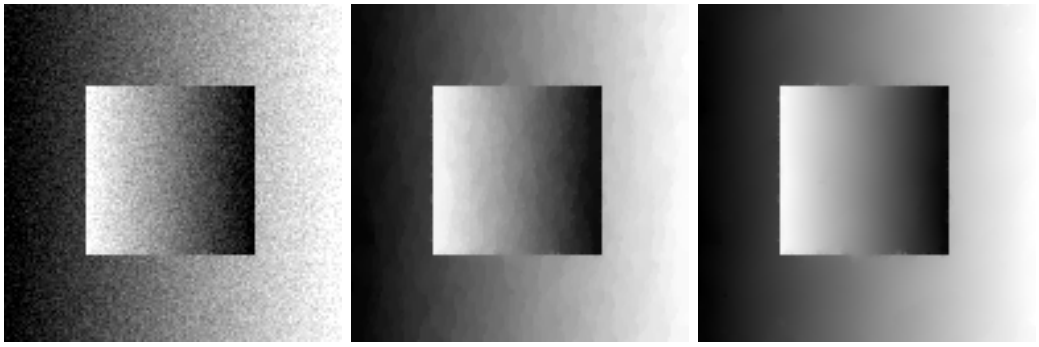
A different approach is to explicitly reason about the location of motion discontinuities based on boundaries in the image, since one would expect object boundaries to coincide with image edges. Cornelius and Kanade [68] test for each image boundary whether keeping the smoothness constraint intact or breaking it across the boundary results in lower energies. Nagel and Enkelmann [179] propose to disable the smoothness constraint only in the direction orthogonal to an image edge, and keep it in the direction parallel to the image edge, which resembles anisotropic diffusion processes [10].

Models employing non-local regularization [145, 229, 270] do not explicitly reason about the presence of boundaries, but change the regularization so that pixels are only assumed to have a similar flow if their appearance is sufficiently similar. Since it is assumed that pixels belonging to the same surface are locally more similar to each other than those from a different surface, this effectively excludes flow from the other surface from distorting the flow of a pixel near a boundary. Consequently, the neighborhood over which the flow is regularized can be increased dramatically [145] without loosing details in the flow. Revaud *et al.* [196] use a similar reasoning, and restrict the pixels that have an influence on the regularization to those that can be reached via geodesic paths on the image.

All these approaches deal with the effect of hard motion discontinuities. Even if those discontinuities could be correctly detected and handled, two neighboring pixels that belong to the *same* surface would be encouraged to have a flow that is as similar as possible. In terms of the surfaces of the world, this effectively means that all surfaces are assumed to be oriented parallel to the image plane and only undergo motion parallel to the image plane, which is obviously a strong oversimplification. This will be addressed in the following section.

2.1.3 Higher-order structure

First order priors, *i.e.* spatial energies that either depend on only pairwise relationships between flow values or that assume that pixels within a neighborhood



(a) Noisy image (b) 1st order prior (c) 2nd order prior

Figure 2.3: Effect of different priors on a denoising example. Using the 1st order prior shows a piecewise-constant structure, while a second-order prior maintains a piecewise-planar structure. Image from [1]

undergo similar motion are common due to their computational simplicity. However, such priors have the disadvantage that they can only model fronto-parallel motion; since they only take two pixels at a time into consideration, they cannot impose constraints on derivatives of a higher order. The effects of this are apparent in Fig. 2.3, where an example of image reconstruction using a first-order (total variation, TV) and second-order [48] prior is shown. Instead of the flow field, in this example the prior penalizes the gradients of the image intensities. The TV reconstruction displays a strong piecewise-constant bias, which in the case of flow translates into the assumption of frontoparallel motion in the scene. Using the second-order prior allows piecewise-planarity, which is a better prior for natural (in particular man-made) scenes [3].

The most obvious way is to directly integrate a second-order prior into the variational framework. Yuan *et al.* [289] penalize the gradients of the divergence and curl of a flow field. This still allows for effects like vortices, which is especially important for highly nonlinear motions such as those present in fluids. Trobin *et al.* [249] propose an unbiased second-order prior that decorrelates the second derivatives. They integrate this into a variational optimization algorithm, achieving good results on standard flow benchmarks. Braux-Zin *et al.* [47] take a slightly different approach, and use an extension to Total Variation, the so-called Total Generalized Variation (TGV) [48]. When using it as a second order prior, the corresponding regularization energy is defined as

$$E_{TGV2} = \min_{\mathbf{v}} \alpha_1 \int_\Omega |\nabla \mathbf{u} - \tilde{\mathbf{u}}| \, \mathrm{d}\mathbf{x} + \alpha_0 \int_\Omega |\nabla \tilde{\mathbf{u}}| \, \mathrm{d}\mathbf{x}, \tag{2.21}$$

[3] Strictly, the connection between vanishing second derivatives and a piecewise-planar scene structure only holds in the case of orthographic projection and not for a perspective camera model. However, locally it is still a good approximation.

again using $\tilde{\mathbf{u}}$ as an auxiliary variable. Vogel *et al.* [255] combine a TGV prior with a CENSUS-based data cost, which is more appropriate for natural images. Lastly, Ranftl *et al.* [193] integrate TGV with non-local regularization, allowing them to accumulate evidence for the higher-order flow field from a larger, image-modulated neighborhood.

Another option to impose higher-order constraints is to over-parameterize the flow. While the methods mentioned above impose second-order priors on the flow field which assigns a translational motion to each pixel, the idea behind over-parameterization is to represent the motion at each pixel using some higher-order quantity (*e.g.* a full homography). These quantities are then ideally easier to regularize, for example piecewise constant even in cases of slanted surfaces or complex three-dimensional motion.

Ju *et al.* [135] parameterize the motion at each pixel using an affine transformation. They first compute affine motion for large patches on a regular grid in the image, and then allow the affine motion at each pixel to deviate from the patch motion. The regularization is done directly on the parameters of the transformation. Nir *et al.* [182] directly integrate an affine motion representation into a variational framework. However, as the reference points for the affine transformations (*i.e.* the center of rotation), they always use the center of the image, leading to significant inaccuracies even in the presence of a true piecewise-affine flow field [249]. Leordeanu *et al.* [152] first establish sparse correspondences and then use the affine motion representation to interpolate between those matches to densify the flow field. In the end, however, they refine using a standard variational technique, falling back on the purely translational motion representation. Hornacek *et al.* [118] go even further, and represent the motion of each pixel using a full homography, parameterized via the translation, rotation, and plane normal in the three-dimensional world. They choose this representation mainly for optimization reasons; the actual regularization of the flow field is still done on the induced 2D translations.

An alternative to such a-priori, handcrafted over-representations is to learn spatial priors from ground truth data. This easily allows to learn the spatial statistics of optical flow over a larger spatial extent, leading to implicit higher order representations. Roth and Black [200] learn a set of 3×3 filters, and model the flow statistics through a linear combination of the responses to these filters, leading to an Fields-of-Experts model [201]. Jia *et al.* [131] use a similar approach, but enforce sparsity on the patches, enforcing each flow neighborhood to be captured by a few elements of an offline learned dictionary. Furthermore, they increase the patch size to 9×9 pixels, allowing them to capture more complex spatial relationships. Gibson and Marques [94] use a similar sparsity-enforcing step to denoise the flow field after computing it using a traditional, variational method; this allows them to learn the dictionary based on the frame itself, thereby better adapting to the data at hand.

Besides learning filters, the parameters of the distribution of flow have been mod-

elled directly. Li and Huttenlocher [155] learn the parameters of a continuous MRF they use to model the flow. Their MRF employs higher-order connections, that is, cliques of three nodes in horizontal and vertical directions, effectively allowing for second-order regularization. Sun *et al.* [230] model the spatial statistics of optical flow using a Gaussian Scale Mixture (GSM) with learned parameters, but do not include higher-order connections. Rosenbaum *et al.* [199] learn a spatial model of flow in patches of size 8×8 pixels, thereby capturing higher order correlations, but use this for flow *denoising* rather than estimation.

All these methods have the advantage that they can capture non-frontoparallel surface orientations in the real world, and are not unduly biased towards piecewise-constant flow fields. However, by integrating the higher order flow priors into the common variational framework, they keep one main disadvantage of this framework, namely the treatment of motion boundaries, which have to be captured either using robust functions or using image-guided modulation, which is difficult in the presence of effects such as motion or focus blur.

2.1.4 Segments

The methods described until now mostly reason about one property of surfaces, the slant. Another important property has so far been neglected, namely, the extent of a surface. After all, all surfaces in the real world have a certain finite spatial extent. Intuitively, reasoning over this extent would allow us to impose stronger constraints within the pixels corresponding to the same surface, and decrease or even remove the regularization between flows belonging to different surfaces.

This brings us to methods based on an explicit segmentation of the scene. Such methods can be roughly divided into methods that first compute a segmentation based on the image and use this segmentation in the flow estimation process, methods that estimate a segmentation based on pre-computed optical flow, and methods that simultaneously reason about the segmentation and the optical flow.

Flow computation under given segmentation

The simplest way to include an explicit segmentation into the estimation of optical flow is to first compute such a segmentation based on the image, in the hope that the resulting image-based segments correspond to segments that have a similar motion, and then compute the flow based on this segmentation.

Fuh and Maragos [87] first extract image regions, based on the sign of the Difference-of-Gaussian operator. They then establish correspondences between the regions using a coarse region-based motion estimate, which is subsequently refined using anisotropic diffusion. Black and Jepson [33] start with a segmentation based on image intensities and estimate a parametric motion model within each segment while allowing for small local deformations from the parametric motion.

For computational reasons, computing optical flow for a pre-segmented image can make discrete optimization approaches feasible. Lei and Yang [150] construct a hierarchical segmentation of the image, resulting in a tree. Since each tree level's motion is relative to its parent, it is generally small; in [150], the relative motion of each tree level is parameterized as a lookup table, greatly reducing the search space and making discrete optimization possible. Kennedy and Taylor [141] use a similar approach, but using a soft constraint, effectively regularizing the flow of a child in the tree to be as similar as possible to its parent. Le *et al.* [148] first track sparse features on the image edges, and subdivide the image into triangles spanned by the sparse feature locations. Within each triangle, the flow is then assumed to be affine. Similarly, Kennedy and Taylor propose to use a fine, triangle-based tesselation of the image domain [140]. They impose 1st- and 2nd-order smoothness priors not between pixels, but between the motion of triangles. The advantage of their triangle-based representation lies mainly in the treatment of occluded regions, since the triangles usually only gets partially occluded. This leaves some pixels of partially occluded triangle still visible, making it possible to assign good flow values even for occluded pxiels.

Going beyond such low-level segmentation cues, recent work has incorporated semantic segmentation. Sevilla *et al.* [216] first use a CNN to segment the reference frame into semantically meaningful segments (*e.g.* road, car, etc.). For some semantic classes, such as the road, which is usually a planar surface, the motion can then be modeled using a parametric representation. The motion of segments that correspond to objects is computed using a two-layer method (see next section). Bai *et al.* [17] similarly use semantic segmentation to split the scene into semantically meaningful objects, and use a rigid body assumption for each object. Hur and Roth [119] extend their method by enforcing the semantic segmentation to be temporally consistent, leading to more accurate flow estimation for objects and better segmentation performance.

Segmentation under given flow estimation

While some of the previously mentioned methods refine the segmentation during the flow estimation process (*e.g.* [216]), they all assume a sufficiently accurate segmentation to be given a-priori; this given segmentation is then used to improve the flow. However, the reverse is true as well, and optical flow can be used to improve a segmentation. Consider a heavily textured object. Static segmentation algorithms often struggle to distinguish edge patterns that are part of the texture from the true object boundaries, resulting in wrong segmentations. In contrast, discontinuities in the (true) flow field only correspond to motion boundaries, *i.e.* boundaries of objects or within the geometry. This can help to compute a better object segmentation, and is exploited by most video segmentation algorithms (*e.g.* [163, 242, 263]).

In flow estimation, a similar line of thinking is employed in the so-called fusion approaches. The idea here is to first compute a set of candidate flow fields, each spanning the full frame. In a subsequent step, a segmentation is computed which assigns a candidate flow field to each pixel, commonly under constraints such as spatial smoothness of the segmentation. This amounts to a standard multi-label discrete optimization, for which efficient methods are known and well-established. Lempitsky *et al.* [151] compute the candidate flow fields using a set of simple and fast flow methods, and select the best at each pixel. Mac Aodha *et al.* [161] similarly use a set of candidate flow fields computed using multiple simple methods, but learn a classifier based on the local image content to predict which flow method performs best at a given location.

Simplifying the flow candidate computation even further, Wills *et al.* [275] first cluster sparse feature matches together, and estimate a single translation for each cluster. Each cluster translation is then used as a candidate flow. Xu *et al.* [282] push this further and generate a translational flow hypothesis for each single feature match. They then use a labeling approach to assign a pixel to one of those hypotheses, or to a traditional variational flow as a fall-back. Chen *et al.* [62] generate for each segment two candidate flow fields, modeling the motion using either a similarity transform or a simple translation, and use the fusion step to choose between both. Vogel *et al.* [256] first segment the scene into small segments, estimate a planar motion for each segment, and use a fusion step to re-assign each pixel to one of the planar motions.

Simultaneous flow estimation and segmentation

Combining both approaches from above leads to methods that *jointly* reason about segmentation and motion, combining both in a unified objective function.

Methods based on level sets represent the segmentation using a continuous function on the full image domain; the segmentation is implicitly defined by thresholding this function [254]. Cremers and Soatto [70] propose a level-set based energy function and show how the level set function itself can be regularized to impose a smooth boundary prior on the segments. However, within each segment, they only consider parametric motion. Similarly, Sekkati and Mitiche [214] use level sets to split the scene into objects, but model the motion of each segment by 3D rigid motion parameters. Amiaz and Kiryati [11] extend this to non-parametric motion, but consider only two segments; Brox *et al.* [51] estimate the flow of multiple segments by estimating a separate level set for each segment; they split a segment if this decreases the energy. Instead of level sets, Darrell and Pentland [73] represent each segment as a separate robust estimator; the segmentation is then implicitly computed through the assignment to these estimators. Oron *et al.* [186] explicitly estimate a segmentation mask, and extend the parametric Lucas-Kanade algorithm to simultaneously reason about object motion and object segmentation. This sta-

bilizes the tracking, since only pixels that are part of the object are included in the cost function.

All these methods create a fairly coarse segmentation, yielding large, object-like segments. Especially when constraining the flow within each segment to a parametric model, this limits the complexity of a scene a method can handle. It can therefore be advantageous to segment the scene into smaller parts; assigning a parametric motion to each still yields a fairly good approximation of the flow field. Memin and Perez [172] cluster the flow field into small segments that adhere to affine motion, and jointly refine the flow and segmentation in a multi-scale framework. Zitnick *et al.* [294] jointly segment a pair of images into temporally consistent, matching, small regions, and model the motion of each region using a translation; due to a lack of explicit occlusion reasoning, however, they cannot deal with large motion of the segments. Vazquez *et al.* [253] uses a generic Cosine basis to model the motion of each segment, but require the number of segments to be given and fixed. Yang and Li [286] model the motion within each segment as a homography and fuse multiple small segments if their motion is similar enough, thereby adapting to the complexity of the scene.

By incorporating a segmentation of the scene, all these methods include an important aspect of the real world, that of the spatial coherence of surfaces and of the similarity of motion based on object ownership. However, all segments are based on isolated pieces, and do not have a depth ordering assigned to them. In the real world, however, coherent surfaces partially occlude each other, which can provide valuable information in the absence of visual cues in occluded areas, as well as areas where a background segment is "divided" by a foreground object. To model such phenomena requires going one step further, and upgrade a segment-based representation to a layer-based representation.

2.1.5 Layers

Segmentation-based approaches create a parsing of a scene into objects. However, they have one big disadvantage, in that they do not model occlusions and any depth relationship between segments. Layers address this shortcoming by representing the scene as a set of segments together with their associated depth ordering, thereby implicitly representing occlusions. Hence, layers can be seen as moving the representation closer to the actual geometry of a scene, since they provide an approximation of the three-dimensional scene structure, in which the background and all objects are considered as flat, pop-up like objects [115].

Existing methods for optical flow computation in layers can be roughly classified along two dimensions. The first is how motion within a layer is represented, either using a simple parametric motion (most often affine), or by a dense flow field at each layer. Using a parametric motion most closely corresponds to a "fixed layer" assumption in the world, such as a plane in 3D under orthographic or perspective

projection. If a dense flow field is used, the advantage of using layers is more in their clean treatment of occlusions; however, since each layer can deform independently, there is no direct geometric connection.

The second axis along which to classify layered methods is the way in which they determine the number of layers, which depends heavily on the complexity of the scene. A scene that contains only a faraway landscape, for example, can be well approximated using only a single layer, even when using parametric motion; a scene with complex 3D geometry and nonrigidly moving objects, on the other hand, requires many layers (if layers are an appropriate representation at all). Interestingly, fixing the number of layers to be small (*i.e.* 2 or 3 layers) still provides a good approximation for a surprising number of scenes. Alternatively, the number of layers can be estimated as part of the algorithm, for example by iteratively increasing or decreasing the number of layers. In this case, the number of layers is initialized as very large or small, and layers are successively removed or added as needed if it reduces the energy. To prevent trivial solutions containing a layer at each pixel, a penalty is usually incurred by each layer that is added. Similarly but more principled, some methods use a so-called Minimum-Description-Length principle, which minimizes the number of layers while still explaining the data.

Parametric layer motion

Shizawa and Maze [220] propose an early layer formulation, but are mostly concerned with motion transparency, that is, estimating translational motion of overlapping, transparent surfaces. To make their formulation well-behaved, they require a large amount of texture on the objects, and can only handle small motions. The seminal paper on explicit layer-based scene representations comes from Wang and Adelson [260]. They establish the common representation, in which each layer is associated with an appearance (*i.e.* the pixel intensity values within the layer), a layer support mask, and a motion, here parameterized as an affine transform. However, they require the number of layers to be known, and use their representation mostly to fill in background regions that are occluded in some frames. Weiss and Adelson [269] describe a neuromorphic architecture to realize a layer-based motion computation. Their model contains separate neural sheets, each tuned for a specific motion direction and magnitude, effectively implementing a translational layer motion, and a gating network that determines which layer of neurons to use in which area of an image. Jojic and Frey [133] use a layered representation for graphics applications. They decompose the scene into the background and a fixed set of so-called "sprites", objects with only slightly varying appearance and masks that can change over time, for example to model the motion of a person. This decomposition is then used for tasks such as object removal or background reconstruction.

Jepson and Black [128] propose a representation of the layered scene based on mixtures, where each layer corresponds to a single mixture component. They use

two mixture components, enough to capture the motion of simple scenes and in small patches (*e.g.* around a motion boundary), and optimize for the motion using an Expectation-Maximization (EM) algorithm. Ayer and Sawhney [14] build on their work, and automatically determine the number of layers using the MDL principle, which tries to minimize the number of layers while still explaining the data as well as possible. Darrell and Pentland [74] use a similar method to determine the number of layers, but allow higher order and rigid body motion on each layer. They use layers mostly for segmentation and to propagate information around and through occlusions, and do not explicitly reason about layers behind occlusions. Jepson and Black [129] study general fitting of layered models using the example of depth maps. They point out that the commonly used EM algorithm often suffers from local optima, leading to segmentation failures, and propose an annealing-based algorithm that can make big changes to the segmentation early in the optimization. To arrive at the right number of layers, they start with a single layer and successively add layers until the data is sufficiently well explained.

In follow-up work, Jepson *et al.* [130] propose layers as object parts, so-called "polybones". Each polybone consists of a parametric shape and motion, and can model, for example, a single limb of a person. Through the use of a layered framework, effects such as self-occlusion can be modelled naturally. Kumar *et al.* [191] use a similar model, and incorporate effects such as per-segment motion blur and contrast changes; however, they do not reason about the mixing of these effects at layer boundaries (for example, the motion blur between foreground and background), and do not compute the flow.

Zhou and Tao [293] use layers for tracking through occlusions They model the scene as a set of object layers, initialized via a change-detection step in the video, each undergoing a 2D similarity transform. These object layers are then interleaved with background layers representing the static parts of the scene; all background layers undergo a common affine transformation. This interleaved architecture allows them to reason about and track objects through occlusions. Xiao and Shah [280] dynamically estimate the number of layers by region-growing from a set of sparse matched features, and explicitly reason about the temporal evolution of occlusions across several frames under the assumptions that objects are wider than their frame-to-frame motion. Similarly, Bleyer *et al.* [45] explicitly model occlusions in a MRF framework. They simultaneously reason about the motion of both pixels and layers, which allows them to exploit the higher robustness of reasoning across segments while at the same time allowing small, per-pixel occlusions. They initialize their layers as the result of an image-based superpixelization, and successively merge superpixels to arrive at larger layer extents. A similarly fine layer segmentation is proposed by Glocker *et al.* [96], who first triangulate the image domain, estimate an affine motion for each triangle, and successively merge triangles that undergo similar motion.

Dense layer motion

All previous methods use a parametric representation to model the motion of a layer. This severely restricts the amount of complexity a method can handle; in a complex scene, many parametric layers would be necessary to accurately capture the motion. An alternative is to assign each layer a full flow field, which then captures the deviations from an idealized, planar motion.

Weiss [268] takes a first step in this direction, and models the general flow within each layer as a linear combination of a large number of Green's functions, which approximates a general quadratic smoothness constraint. His method is initialized with a large number of layers, which are successively removed and merged together. Similar to common practices in object tracking, Yalcin *et al.* [283] combine dense flow within every layer with a model of the temporal change of appearance of the individual layers. However, their method only uses a single foreground layer and a temporally persistent background.

A series of articles by Sun and colleagues [232, 231, 233, 228] constitute the most recent treatment of layered methods. In [232], they propose a unified energy function, jointly reasoning about the motion, segmentation, occlusion and depth ordering of layers. One novelty here is that they enforce a temporal consistency of the segmentation, that is, the layer segmentation from t to $t+1$ should be equal to that from $t+1$ to t; violations of this are detected as occlusions. To use a continuous optimization framework, the segmentations are parameterized as continuous maps, similar to level set methods, and the motion within each layer is parameterized as affine plus small deviations [135]. However, they only consider a fixed number of 3 layers.

In [231], they extend this work by dynamically choosing the number of layers; they start with 10 layers, and successively remove layers that do not provide sufficient benefit to the model. Furthermore, they replace the continuous layer support maps by a set of ordered MRFs. This allows them to use discrete optimization in the layer support computation, reducing the problem of local minima. Lastly, they show that accurately resolving occlusions requires the processing of > 2 frames at a time. However, while producing accurate results, their method is computationally inefficient, requiring as much as 22 hours for a sequence of 5 frames.

In [233], they show how using a densely connected Conditional Random field (CRF) [144] to compute the layer support yields extremely fine layer segmentation while at the same time being much more computationally efficient; processing a sequence of 5 frames takes them 22 minutes. However, they only demonstrate their approach using two layers, which limits the applicability for complex scenes. Lastly, in [228], they turn to *local* layers, that is, use layer reasoning across small spatial neighborhoods. An initial segmentation is computed using SLIC superpixels [236], and a small layered model is computed at each superpixel boundary. By grouping superpixels of similar appearance, they ensure a global consistency in the layering.

2.1.6 Temporal structure

So far, we have described approaches that impose a spatial structure on the projected motion field. Beyond this spatial structure, however, classical dynamics dictate that any observed scene also has a temporal structure. Objects cannot vanish into thin air, energy and momentum have to be conserved, inertia plays a big role in the change of motion that objects undergo, air provides resistance etc. All of these phenomena can be seen as constraints on the temporal evolution of the optical flow; hence, reasoning about them requires taking into account more than two frames at a time.

Methods for optical flow estimation that use more than a pair of frames at a time can be roughly subdivided into the following four categories.

Semi-dense trajectories

These methods live in the space between fully dense optical flow estimation and sparse tracking of objects or features, in that they compute motion only for a subset of pixels, usually between 10 % and 50 % of pixels. Sand and Teller [206] first explored this approach. They treat each of their semi-sparse points as a particle, and track those particles over time, using a variational formulation which includes both a data term of the particles as well as a smoothness constraint, encouraging nearby particles to move similarly. In occlusions, particles are removed; when an area becomes disoccluded, new particles are spawned, but particles are not tracked through occlusions. Sundaram *et al.* [235] start with optical flow computed a priori [52] and use it as a basis to compute semi-sparse trajectories. The flow at sub-pixel locations is interpolated, and trajectories are removed when an occlusion occurs. Their goal lies primarily in the computation of trajectories for clustering and subsequent motion segmentation, and less in the refinement and temporal consistency of the motion itself. Lang *et al.* [146] start with sparse feature tracks over an image sequence and then use an image-guided filtering approach to interpolate between the features. While they obtain dense flow, the temporal consistency is only enforced at the locations of the sparse features.

Data conservation over time

A second approach when computing dense optical flow under temporal constraints is to include more than two frames in the data term. The reasoning here is that the appearance of a surface point is consistent over time, and that at the same time using a higher number of measurements can increase robustness against effects such as noise.

Taking inspiration from the human visual system, Heeger [113] proposes to use a bank of oriented Gabor filters in the spatio-temporal volume to detect moving regions of different spatial frequencies, velocities, and orientations; the flow can then

be computed from the filter for which the motion energy is highest. Nagel [178] implicitly makes the assumption of constant motion in the data term by computing the temporal derivative $\frac{\partial I}{\partial t}$ in Eq. (2.17) across three frames. This resembles the filters that are used to compute the spatial derivatives, and that generally use a larger neighborhood than just two pixels to increase robustness when computing the derivatives. Computing the temporal derivative using multiple frames like this implicitly assumes the flow to be constant across the number of frames taken into account.

Wang *et al.* [259] uses pairwise temporal derivatives, but accumulates the data error across multiple frames, downweighting the error belonging to frames that are further away from the reference frame. They again assuming constant velocity within a window, and do not explicitly model effects such as occlusions, which become more prominent as the number of frames that a method takes into account increases. Similarly, Werlberger *et al.* [271] test their method with a data term stretching across three frames under the hard assumption of constant velocity, but report no improved results. Janai *et al.* [126] also use a constant velocity assumption. However, they compute flow for videos that are recorded using several hundred frames per second; due to inertia, the deviation of the true optical flow from constant velocity is much smaller in this setting. By accumulating the data term over more than two frames, the basic assumption of all these method is that the appearance of a surface point changes slowly, even across longer periods of time.

Temporal smoothness of motion

An alternative approach is to not impose temporal consistency on the appearance of the point, but on the motion itself, thereby implicitly and approximately including inertia and the conservation of momentum by forcing the flow to vary smoothly. Murray and Buxton [177] were the first to propose a temporal smoothness constraint on the optical flow. They assume constant velocity at each point, and only model the motion of simple scenes containing fronto-parallel surfaces. Black and Anadan [37] generalize this in a Markov Random Field framework, and robustly penalize flow that deviates from an accumulated history of motion at a given location. Bergen *et al.* [31] proposes an algorithm that uses three frames to decompose a scene into two overlaying, moving regions under the assumption of constant velocity. In an iterative framework, they construct two temporal difference frames with one motion removed, and use these difference frames to compute the second motion. Chin *et al.* [64] impose a simple quadratic regularization of the flow in the temporal dimension, and solve for the flow using an approximate Kalman filter. Chaudhury and Mehrotra [58] explicitly constrain the trajectories of all points to be as smooth and short as possible, and integrate this assumption into a discrete optimization framework. However, they only consider a small, discrete label space, and thus cannot handle large displacements. Weickert and Schnörr [266], similar

to [64], use a variational model and extend the regularization to the temporal domain, but include a robust error function, which allows them to properly account for occlusions. They consider a full sequence at once, and solve for the flow using a three-dimensional diffusion process in the resulting spatiotemporal volume.

Kennedy and Taylor [140] compute predictions of the flow fields by extrapolating the flow from previous frames, and use these as additional flow proposals in a fusion step. In follow-up work [141], the same authors directly integrate a constant flow assumption into a hierarchical, discrete flow optimization.

An issue that all these methods have in common is that they impose temporal smoothness on the flow at a fixed image location. Instead of regularizing the motion of a *fixed surface point*, they thereby impose a temporal coupling of the motion of *all surface points moving through a given image location*. Hence, temporal smoothness and spatial smoothness assumptions are not properly separated from each other.

Black [35] proposes to address this problem by warping the neighboring flow fields before imposing temporal smoothness; Salgado and Sanchez [205] integrate this approach into a variational framework. This decouples the spatial and temporal assumptions, but imposes high computational costs.

All preceding methods assumed or penalized deviations from constant velocities, and therefore could not model effects such as acceleration or even constant motion under a perspective projection. Black and Anadan [38] instead assume constant acceleration. They propose an online algorithm that predicts the current optical flow field by adding an estimated acceleration to the previous flow field, and initialize their flow estimation using the predicted flow. However, they suffer from the same issue of temporal consistency in fixed image locations that was mentioned above; in follow-up work [36], they fix this issue and use a warping-based formulation. Volz *et al*. [257] address this issue by parameterizing all flow fields within a temporal window with respect to the reference frame. This is effectively the same warping-based approach of [205], but does not require the re-warping steps. They impose first and second-order smoothness on the optical flow, and include data from all frames into their robust, image-guided regularization, effectively imposing a temporal consistency constraint on object boundaries that should not be blurred in the regularization step.

Coherence in grouping

Several methods combine the idea of temporal consistency of object boundaries with an explicit segmentation of the image, using the assumption that the segmentation of an image into meaningful parts is consistent over time. Shi and Malik [219] first compute autocorrelation matrices at each pixel, and use this as a feature to perform clustering in a spatio-temporal volume. This enforces similarity in both time and space within each segment, while at the same time automatically estimating the number of segments. Irani [121] explicitly formulates constraints on the optical

flow of a rigid object moving in 3D space. In the most general case she considers (perspective projection, arbitrary camera calibration, and small rotation and small motion of the camera in depth), she shows that a matrix of the optical flow vectors has at most rank 9. With two measurements (u_1 and u_2) per pixel, at least five frames are therefore required to decide whether a set of optical flow vectors belong to an object undergoing such a motion.

Yalcin *et al.* [283] integrate longer-range temporal consistency in a segmentation-based method, by explicitly modeling how the motion and appearance of the individual segments can change over time. Similar to [146], Xiah and Shah [280] start out with sparse feature tracks across multiple frames, but then use an explicit region-growing algorithm to densify their flow field under the assumption of planar motion within each segment. Brox *et al.* [51] integrate multiple frames in a segmentation-based method that uses level sets to represent the segments. They impose the TV prior on the level set in time as well as in space, ensuring a smoothly varying segmentation. However, they use the unwarped temporal regularization, which is problematic, as pointed out above. Sun *et al.* [231] integrate temporal consistency into a fully layered method, by encouraging temporal coherence in the data, segmentation, and flow. Furthermore, they point out that to accurately reason about occlusions in the presence of multiple layers, at least three frames are required.

It should be noted that despite the works mentioned here, the vast amount of papers on optical flow still only consider the flow between a pair of frames, mainly for two reasons. First, computing the flow under higher-order temporal constraints obviously increases the computational burden, since multiple flow fields (*i.e.* the flow at multiple points in time) have to be estimated simultaneously. Second, most datasets such as Middlebury [23], KITTI [90], and Flying Chairs [76] either only provide or only evaluate on a single pair of frames per sequence, encouraging research to concentrate on flow estimation from just two frames. Using only two frames at a time effectively amounts to no temporal regularization at all, since, even for a longer sequence, every optical flow field is estimated separately from all others.

This concludes our discussion of optical flow methods that impose regularization on the image plane. Arguably, this is the easiest case of regularization, since the representation (the optical flow on the pixel grid) is readily available and assumptions about further properties of the recording setup such as the shutter time and the focal length do not have to be made. Yet, since these regularizations only reason about the motion of pixels, it is often difficult to say how they constrain the motion in the real world, *i.e.* the motion of surfaces and objects in three dimensions.

In the following, we will hence look at methods that explicigly model the imaging process, thereby taking into account effects such as a rolling shutter or motion blur, and methods that explicitly reason about the motion in three-dimensional space.

2.2 Constraints on the observation process

The previous sections described constraints that were directly imposed on the two-dimensional motion fields corresponding to general, three-dimensional scenes. Of course, these motion fields are never immediately observed, but are always computed from recorded images. The image recording process itself, however, is influenced by a number of effects, which in turn are results of the structure and dynamics of the world. If we can isolate those properties of the recording process, we can thus hope to use them as additional cues to the optical flow or to obtain additional information about the world.

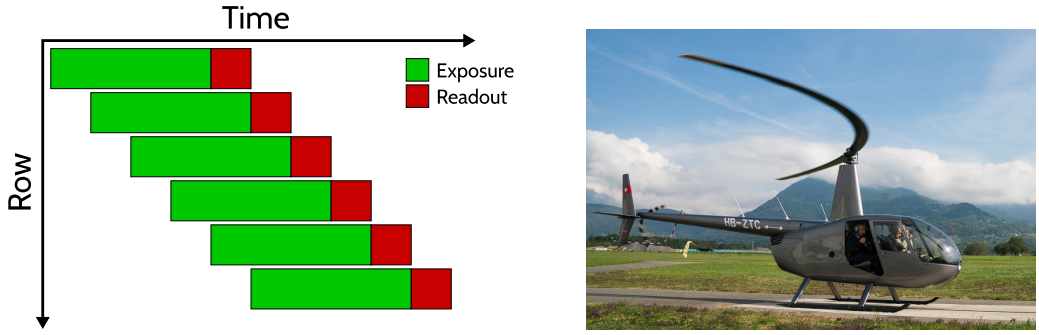
In this section, we will concentrate on two particular effects of the recording process, rolling shutter and motion blur. The first is an artifact of the way in which image are recorded in current CMOS sensors, such as the ubiquitous cameras on mobile phones. Since the images are read out line-by-line from the sensor circuit, any object with horizontal optical flow (such as rotating objects or the whole scene if the camera undergoes a horizontal pan) will appear skewed.

The second effect, motion blur, appears because the camera always has to record the scene for a finite amount of time, so that enough light can hit the sensor. While this finite temporal window is usually short, it can extend over longer periods of time, especially in low-light settings, for example when recording a scene at night. If motion occurs during the time the sensor captures the light, a given surface point in the image will project to multiple locations in the frame, effectively tracing the path the projection of the point took during the recording period.

In the following, we will review previous work on these effects, with a particular focus on optical flow computation. Note that other side-effects of the recording process exist as well, such as transparency effects at pixel boundaries from focus blur and aliasing (*i.e.* object discontinuities that does not fully cover the site of a pixel's recording). However, since treatment of these effects in optical flow algorithms is extremely rare, we will omit them here.

2.2.1 Rolling shutter

The image sensors in most digital consumer cameras and phones are based on the CMOS (Complementary Metal-Oxide Semiconductor) technique. CMOS sensors are cheap to produce and allow a tight spatial component footprint, since the components that store the electric charges are directly integrated into the sensor. One side effect of this, however, is that the pixels cannot be read out at the same time; instead, the image is read sequentially, one row at a time. The recording times between adjacent image rows are hence shifted by a few fractions of a second, as illustrated in Fig. 2.4(a). This leads to the so-called "rolling-shutter" effect: If the projected motion of an object is very large (because the object is either moving very fast, or the camera is rotating quickly), the image contains unnatural skew, as



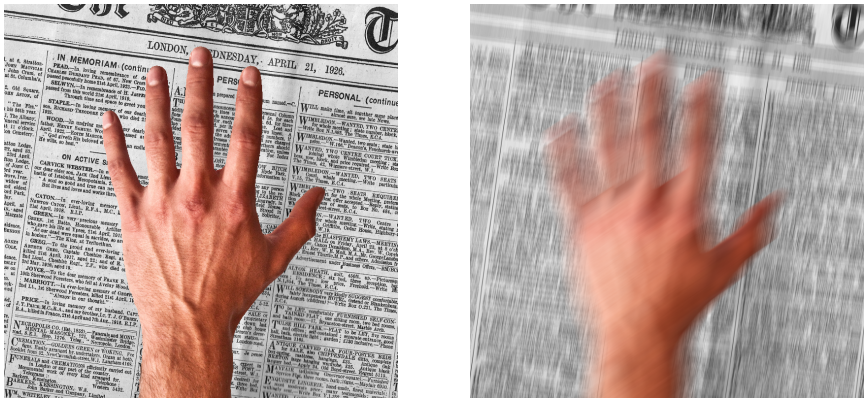
(a) Readout pattern (b) Rolling shutter example

Figure 2.4: Rolling shutter. (a) In cameras using CMOS chips, the scanlines are read out sequentially, leading to temporal offsets among neighboring image rows. (b) The effects are especially visible in fast moving objects such as propellers. Image source: Flickr.

seen in Fig. 2.4(b). Mathematically, this is usually modelled using extrinsic camera parameters that vary over time and therefore are slighty different for each image row [157].

Commonly, rolling shutter effects are considered a nuisance, and a number of papers focus on removing its effects from an image sequence by re-aligning the image rows [20, 99, 157, 234], thus eliminating the skew. This is often formulated as a temporal super-resolution problem, since each scanline is sampled at a different location in the spatio-temporal volume. By interpolating the full volume and re-sampling it at slices at the same point in time, the rolling shutter effect can be removed [20, 234]. Alternatively, the effects of rolling shutter can be considered as spatially varying image shake. Grundman *et al.* [99] take this approach, and compute motions independently for small image patches. This allows them to simultaneously remove rolling shutter effects and to stabilize the video.

Explicitly modeling the rolling shutter becomes important in Structure-from-Motion scenarios, where a goal is to explicitly estimate the projection matrices at each point in time. As mentioned above, these projection matrices vary with each scanline, even during the recording of a single frame. A common approach is to first estimate the global camera motion between frames using a global shutter assumption. From this global motion, the camera motion during the recording of a frame can be interpolated, often using simplified camera motion assumptions such as fronto-parallel translation [171] or pure rotation [111]. Mailand *et al.* [170] estimate a 6 DoF camera motion and also model motion blur. However, none of these approaches explicitly deal with parallax, *i.e.* the effects of different depth on different motions, and the subsequent varying strength of rolling shutter distortion even within a single scanline. Sauer *et al.* [208] address this issue, and model the effects of both parallax and lens distortion; the latter has to be accounted for, since



(a) Sharp image (b) Blurry image

Figure 2.5: Motion blur is a cue to motion. Compared to a sharp image (a), an image containing motion blur (b) creates a strong impression of motion in the scene.

after removal of lens distortion (a common preprocessing step), the scan lines that are read out at the same time do not correspond anymore to image rows in the undistorted image.

While these approaches include rolling shutter effects into their models of the projective geometry, their ultimate goal is to not suffer from such effects. However, a rolling shutter can also be seen as a blessing in disguise; after all, it provides both a temporally higher sampling of at least *some* information about the image, as well as temporal information from even just a single image. Some works aim to exploit this. Ait-Aider *et al.* [6] use the additional temporal information included in a single image to compute the 3D pose and translational motion of an object from a single frame; however, they require known correspondences between 2D pixel coordinates and 3D points on the surface of the object. Gu *et al.* [100] modify the readout pattern of the sensor in order to obtain even more information from a single frame. This allows them to record multiple sub-frames, making the optical flow problem easier, and to record a single image with different exposure times to generate High Dynamic Range (HDR) output. Lastly, Su and Heidrich [227] use the information contained in a rolling shutter image to help remove motion blur. Assuming a planar scene and translational motion, they estimate the camera motion from the rolling shutter image, and use it to remove the motion blur. They then iteratively improve both the estimated camera motion and the sharp image.

2.2.2 Motion Blur

Similar to rolling shutter, motion blur is another artifact of the recording process. In order to collect a sufficiently large amount of photons, the sensor in a digital

camera has to be exposed for a non-finite amount of time, illustrated via the green bars in Figure 2.4(a). If an object or the camera itself moves during this exposure period, the light reflected by a single surface point gets recorded by multiple sensing elements of the sensor, resulting in a "smeared" appearance in the recorded frame, as shown in Fig. 2.5(b).

Besides reducing the visual quality, this presents three challenges for the computation of optical flow. First, the motion blur effectively acts as a one-dimensional lowpass filter on the image, and reduces visibility of details that are smaller than the extent of the blur. This can be seen on the back of the hand in Fig. 2.5, where the hair and skin pores are much harder to see in the blurry image. Such regions can become effectively untextured; as we have seen, this presents problems for optical flow algorithms due to the aperture problem. Second, if an object is not moving on a linear path, the surface points that are recorded by a pixel (*i.e.*, that "move through" the pixel during the period of exposure) vary from one frame to the next. In this case, the appearance of corresponding pixel locations across multiple frames changes, and the brightness constancy assumption is not valid anymore [195]. Third, during exposure a pixel might receive light from two different surfaces, for example near depth discontinuities or edges of moving objects. This is shown around the hand in Fig. 2.5. In such pixels, the background and foreground blend together, again violating the brightness constancy assumption, and making segmentation and boundary detection hard.

Yet, motion blur is a manifestation of motion in a single image, and hence contains information about the motion of the world. Figure 2.5 shows this effect. In the sharp case of 2.5(a), one cannot tell whether there is motion occurring, and if so, in which direction. Figure 2.5(b), in contrast, not only creates a strong impression of the presence of motion, but also lets us determine that the hand must be moving diagonally and the background roughly vertically (although the exact direction of either remains ambiguous). Therefore, the effect of motion blur is similar to that of a rolling shutter. It is an artifact of the way cameras are built, commonly treated as a distortion to be removed from an image, yet contains information about the motion of a scene.

The treatment of motion blur has focused mostly on removing blur from single images captured in low-light scenarios. The scene is usually assumed to be static, and the blur to be caused by camera shake. The blur process is modeled either as a complex but spatially-invariant blur kernel [65, 81] or as a spatially-varying kernel generated by possible camera motions in 3D [274]. Levin [153] considers images consisting of two segments, one that undergoes a purely translational motion and is distorted by motion blur, and another that is sharp. Using a prior on image statistics, she then segments the image and deblurs only the moving object. Similarly, Chunhe *et al.* [67] use a layered formulation for single image deblurring. Their work considers non-overlapping layers, more akin to a segmentation, and adapts the spatial prior accordingly. These and similar methods all explicitly estimate the

blur kernel, but only implicitly reason about the actual motion of the scene. A full review is hence beyond our scope here, as are methods relying on special hardware for multiple exposures [215, 29].

A different scenario is the treatment of motion blur in video sequences, since the availability of multiple frames allows an algorithm to explicitly reason about the motion within a scene and include this knowledge into the deblurring process. Commonly, this is done sequentially, and the motion of the blurred sequence is first estimated using sparse feature tracking [110], shapes [28], or optical flow [149, 285]. The estimated motion is then used to synthesize the blur kernel and deblur the input images non-blindly. Layered methods [261] were originally invented to address the motion blur problem in case of multiple moving objects. The idea was to first compute motion using a layered model and then, given the layer segmentation, model the blur process. All these methods fail in the case of strong blur where accurate tracking becomes infeasible [183]. A different approach to multi-frame deblurring is to not model the motion at all, but use co-occurences across multiple scales in space and time, for example by using deconvolution in a coarsely upsampled spatio-temporal volume [239] or by using patch-based synthesis [66]. While such methods give good deblurring results even in the presence of independently moving objects, they yield neither motion information nor scene segmentation.

The problem of accurate optical flow estimation in the presence of motion blur has been rarely studied so far. One way to integrate motion blur into the optical flow computation is to include an additional regularization term for the motion, minimizing the difference between a latent, sharp image and the observed, blurry one [183]. This approach requires such a latent image, however, which might not always be available. Alternatively, the motion blur can be integrated directly into the data term by modulating the brightness constancy constraint, either globally [192] or only in regions that are previously detected to be blurry [250]. A similar approach has also been used in sparse tracking [132], where the motion is used to modify the expected appearance change of features. However, these methods usually assume a smooth or segmented flow field and do not explicitly reason about the effects of blur at overlapping surfaces; consequently, they do not give good results at object boundaries.

The methods mentioned so far concentrate on either estimating the motion or on deblurring the scene. An alternative approach to treating motion blur in a video is to formulate a joint energy function over the latent, sharp images and the motion parameters, and iteratively solve for both. Bar *et al.* [25] take such an approach and use a layered framework to model the difference of motion and blur between several objects; however, they only consider translational motion and require the background to be known and static, simplifying the problem. Li *et al.* [154] use a more generic general optical flow to model the motion, but use an external acceleration sensor to estimate the camera motion, which is the largest source of motion blur. Similarly, Li *et al.* [156] optimize a joint energy using gradient descent, but

do not rely on external sensors. However, they assume a single camera motion and cannot handle independently moving objects. Furthermore, their goal is not explicit estimation of motion, but the removal of motion blur to obtain images that can be stitched together to create sharp panoramas. Paramanand and Rajagopalan [189] use a layered scene representation to capture the spatial variation of blur due to parallax, but they only consider a static scene with camera shake, model the motion as one similarity transform per layer, and do not explicitly model layer interactions at the object boundaries. Kim and Lee [142] model the motion as general optical flow, and iteratively compute the flow and use non-blind deblurring to remove the blur induced by the flow. Like [189], they do not explicitly reason about depth discontinuities and motion blur in overlapping regions.

Lastly, and similar to rolling shutter, one can consider motion blur to be a source of information. One example for this is to compute motion from a *single blurred image*, either using the image directly [61], analyzing its Fourier spectrum [212], or by extracting a matte of a motion-blurred object [71, 218]. In the latter case, the alpha values of the matte at the object boundaries directly correspond to the strength of the blur, and can be used to extract spatially varying motion information. Such mattes from motion blurred foreground objects can also be used in graphics applications, for example to replace the background [72, 158]. Matting-based approaches, however, require the user to provide at least a trimap, *i.e.* an image where regions that are clear foreground and clear background are marked, and often assume a static background; yet even without such user intervention motion blur can be used to segment moving objects [55].

2.3 Constraints on the geometry

The previous section showed how constraints on the optical flow field can be moved "into the world" by considering temporal effects of the image formation process. However, so far we have ignored another big source of constraints on the optical flow field, namely, the three-dimensional structure of the world[4]. This world is what the camera sees, and optical flow is the projection of motion that occurs in it. Taking properties of the world into account such as temporal coherence or the deformation of objects can hence simplify and constrain optical flow estimation. If properly implemented, such properties restrict optical flow fields to those that are physically plausible. This stands in contrast to, say, an optical flow field in which the motion is random and independent at each pixel; one would be hard pressed to find a world that is compatible with such a motion. Furthermore, reasoning about the flow purely on the image plane sometimes throws together things that do not belong together. For example, the motion of a scene is often a combination of the

[4]While layers can be seen as approximating this structure, they are usually treated from a two-dimensional, image centric perspective.

motion of the camera and the motion of objects. Treated separately, both can be very simple, and potentially trivial to compute. When combined, however, complex motion patterns can arise that make the inference process hard.

Here, we will describe two different scenarios in which properties of the three-dimensional world can be integrated into optical flow computation, namely the case of a static scene and rigidly moving objects, and the case of nonrigidly moving objects. In the first case, the optical flow is commonly constrained by epipolar constraints, while in the second the deformation that an object can undergo is modelled using a low-dimensional subspace, and the flow is restricted to lie in this subspace.

2.3.1 The static scene and rigid objects

When considering constraints that the three-dimensional structure of the world imposes on the optical flow field, a simple and yet powerful case is to assume a static world, *i.e.* a world in which no objects move except for the observer. In this case the optical flow of the projection of a point is fully determined by the motion of the camera and the distance of the point to the camera. The problem is hence more constrained and (theoretically) computationally simpler than general optical flow, since only a single number (instead of a 2D vector) has to be estimated per pixel. While the problem of estimating the camera motion and the three-dimensional structure of a static scene is closely related to Structure-from-motion (SfM), there are several important differences between optical flow and SfM. Generally, SfM methods require purely rigid scenes and use sparse point matches, wide baselines between frames, solve for accurate camera intrinsics and extrinsics, and exploit bundle adjustment to optimize over many views at once. This makes it possible, for example, to accurately reconstruct geographical landmarks from random collections of photos [224]. In contrast, optical flow is applied to generic scenes containing nonrigidly and independently moving objects, exploits continuous optimization, makes weak assumptions about the scene (*e.g.* that it is piecewise smooth), works with small baselines, and typically processes only pairs of video frames at a time.

Many early works on motion estimation are concerned purely with the computation of the motion of the observer (*i.e.*, the translational and rotational velocities of the camera). One way to compute the ego-motion is to first compute feature matches [147] or general optical flow [262, 112], and compute the ego-motion in a subsequent step. Adiv [5] additionally segments a given optical flow field into rigidly moving objects, and estimates the rotations and translations of all these objects relative to the observer. All these methods require accurate motion to be computed a-priori, and do not refine it to be compatible with the static scene / rigid objects assumption.

In contrast, *direct methods* process the pixel intensities directly, and do not require precomputed correspondences. They compute the optical flow only implicitly,

while reasoning directly about the camera motion given two or more frames. Several early works go this path, but often put additional assumptions on the world, such as planarity [180] (approximately valid if the depth variation of the scene is small compared to the motion of the camera in depth), or on the camera motion [117]. An algorithm that estimates *general* camera motion was proposed by Taalebinezhaad [238], but not verified experimentally. Hanna [106] provides an analysis of different image structures (such as corners and edges) and their influence on the egomotion estimation, and constrains the depth structure to be locally planar. Instead of using the simple pixel intensity error, Mandelbaum *et al.* [166] show how to integrate correlation across a larger patch size into a framework that simultaneously reasons about the inverse depth and the camera motion; however, their iterative algorithm requires a good initialization, for example from an accelerometer. Direct SLAM-methods such as [78] also estimate the observer motion from direct image measurements; however, similarly to SfM methods, they usually require tens or hundreds of frames and make additional assumptions, such as smoothness of the camera motion.

Insofar as they explicitly compute the motion of the observer in 3D space, these methods explicitly compute the translational and rotational components of the motion, assume calibrated camera intrinsics (in particular, a known focal length), and only reason about the *instantaneous* motion, that is, the derivative of the motion. Optical flow, however, is concerned with the motion from one frame to the next, and reasoning about instantaneous motion is only valid if the rotation and camera translation in depth are small, or the scene is very far away from the camera [291]. Furthermore, as pointed out by Horn and Weldon [117], reliably distinguishing rotational and translational observer motion requires a wide field of view, and becomes unstable in the case of long focal lenghts. Lastly, as mentioned above, these approaches never explicitly estimate the optical flow field, but directly compute the observer motion and scene structure in 3D. Bergen *et al.* [30] attempt to bridge this gap, and propose a unifying framework ranging from affine motion and the motion of rigid objects to general optical flow; however, they still treat the latter two as separate.

A different approach is to restrict the optical flow of the scene (or of rigidly moving objects within a scene, given a segmentation) to lie in a low-dimensional subspace, since it has been shown that the motion of such objects can be factored out into the three-dimensional structure of an object and the rigid body motion [244]. Following this idea, Zelnik-Manor and Irani [291] use linear subspace constraints to restrict the motion of a planar objects in multiple frames; explicitly using the subspace constraints to restrict the optical flow computation increases robustness, and allows the sequence to be registered to even a small planar object. Since in the general case their subspace has a dimensionality of 6, they require motion measurements from at least 6 frames and thus process longer sequences of frames compared to classical optical flow algorithms. This approach can be extended to the motion

of general (*i.e.* non-planar) rigid objects, in which case the subspace has a dimensionality of 9 [121]. Similarly, by choosing affine parametric motion to represent the flow at each pixel, the method of Nir *et al.* [182] implicitly impose a subspace constraint corresponding to planar surfaces under an orthographic projection.

More recently, there have been approaches to integrate epipolar geometry into optical flow computation by enforcing the epipolar constraint $\mathbf{x}'^{\top}\mathbf{F}\mathbf{x} = 0$, with $\mathbf{F}$ being the fundamental matrix encapsulating the intrinsic parameters of the camera at both points in time as well as the relative motion of the observer. Oisel *et al.* [185] first compute the fundamental matrix using sparse matches and then re-formulate a variational optical flow objective to only allow correspondences that are compatible with the epipolar constraint. Wedel *et al.* [265] relax this hard constraint to allow moving objects, and enforce the epipolar constraint as a soft prior in a duality-based optical flow method. Valgaerts *et al.* [252] jointly optimize for $\mathbf{F}$ and the optical flow, but only test on static sequences. Recently, Yamaguchi *et al.* [284] use a similar approach, while at the same time integrating a piecewise-planar constraint on the depth structure of a scene. They achieve top results on the KITTI-2012 benchmark [91], but only consider fully static scenes without moving objects. To handle sequences containing moving objects, Wedel *et al.* [264] propose an adaptive method. If such objects are detected (*i.e.* too many objects move relative to the background), the epipolar constraint is completely switched off, allowing them to adaptively enforce the constraint on a per-frame basis.

Mostly, however, these methods are concerned about the motion of a static scene, and moving objects are considered a failure of the model. In the case of rigidly moving objects such as cars, however, the same basic epipolar geometry applies – the motion within each object can still be constrained by a fundamental matrix, which now has to be separately estimated per object together with an object segmentation. Roussos *et al.* [203] assume a known calibrated camera and solve for depth, motion and segmentation of a scene with moving objects. They perform batch processing on sequences of about 30 frames in length, making this more akin to SfM methods. While they have impressive results, they consider relatively simple scenes and do not evaluate flow accuracy on standard benchmarks. Menze and Geiger [173] use a piecewise-planar scene model and segment the scene into objects and background, but require stereo inputs and a largely planar, urban environment. More general and using monocular input, Bai *et al.* [17] segment the scene into semantic objects and estimate a separate fundamental matrix for each. This independently constrains the motion on a per-object basis, but requires full rigidity within the objects. Hur and Roth [119] extend this by enforcing a temporal consistency of the segmentation.

Unfortunately, all methods relying on the fundamental matrix suffer from a common issue, namely that it can be hard to estimate, particularly in case of small baselines [185]. For some scenarios this does not pose a problem; for a fast driving car in automotive vision, for example, the baselines are usually large. For general

scenes, however, this limits the applicability of epipolar based methods, or requires strong assumptions about the camera motion [287].

2.3.2 Non-rigid motion

The works described in the previous section all assume either a purely static scene or a scene with rigidly moving objects. The common assumption in the latter case is that the motion of each object can be expressed using few, object-independent parameters, namely translation and rotation. In case of deformable objects, the motion is still restricted (there is one coherent surface of the object, usually no self-penetration etc.), however, the manifold on which this motion lives is much more complicated and potentially object-dependent. A tree swaying in the wind can be considered as moving with few degrees of freedom, and so can a person; yet, the degrees of freedom of their respective motions differ greatly. Exploiting this, however, requires both a semantic understanding of the scene as well as a determining the constraints on the motion that the semantics induce; both are hard problem in themselves. Therefore, only very few methods consider non-rigid motion constraints when estimating optical flow, and those that do, usually either do not take the semantic class into account or restrict themselves to *e.g.* human bodies.

One approach is to extend the subspace methods used in motion estimation for rigid objects. In the rigid case, a matrix containing the 2D motion vectors is factored into a constant shape vector and a time-varying, low-rank motion matrix [244]. A three-dimensional, deformable object on the other hand can be modelled as a linear combination of basis shapes, so-called blend shapes, together with time-varying coefficients determining the weights [49]. Under weak perspective projection, this also holds in the 2D image plane; a matrix containing the 2D motion of a deformable object can thus be factored into two low-rank matrices, one containing the projected basis shapes, and the other the coefficients. Torresani and Bregler [247] propose to turn this around, and consider the coefficient matrix as a trajectory basis. The motion of each point on the deformable object then moves according to a linear combination of the trajectories in this basis. Garg *et al.* [88, 89] extend this idea to dense optical flow by integrating the nonrigid subspace constraint into a variational algorithm. They first track sparse features across a sequence of frames, use those to compute the subspace, and enforce the dense optical flow to lie on this subspace [88]. In later work [89], they relax this into a soft constraint, accounting for errors in their subspace computation.

However, all these methods have been demonstrated on restricted domains only, such as single deforming objects (flags, sheets of paper, etc.) or specific semantic categories, such as faces or bodies. Furthermore, they all require the deformable object to be segmented a priori, and few works have attempted to overcome this. Fragkiadaki *et al.* [86] cluster sparse feature tracks into groups corresponding to

separate, non-rigidly moving objects, and use per-object subspace constraints to densify the motion and structure predictions and handle occlusions and missing data. Russel *et al*. [204] use a hierarchical segmentation, and first segment the scene into the background and moving objects, followed by a segmentation of each object into rigidly moving parts, for which they enforce a classical epipolar constraint. Both methods show good results on unsegmented video sequences containing a single dominant object, however, neither has been demonstrated to work on complex scenes like those in the Sintel flow benchmark.

2.4 Summary

This chapter has described several constraints on optical flow fields, where in the image formation process they arise, and what some of their properties and shortcomings are. Specifically, we have seen that these constraints can be divided into three broad categories: constraints arising from the structure of the two-dimensional optical flow field, constraints that are temporal side-effects of the actual image recording process, and constraints that pertain to the fact that any video is the 2D projection of the three-dimensional world with clear rules and laws of physics.

However, we have also seen a number of shortcomings and over-simplifications of current models, leading to artifacts in the computed results or limited applicabilities of proposed algorithms. Some examples of those were the lack of efficiency in current layered optical flow methods, since the flow within each layer is commonly computed using a generic variational flow method; the lack of treatment of motion blur at object boundaries, which leads to severe artifacts both in deblurring results as well as in the flow computed from a motion-blurred sequence; and the limited applicability of methods including geometric constraints in the flow estimation procedure, since those methods commonly make overly strong assumptions about either the scene or the camera motion.

The following chapters will propose possible solutions to these issues. Using a layered scene as a core representation, we will show how higher-level constraints on the optical flow field can be integrated into the layered framework, leading to robust, accurate, and efficient optical flow methods.

Chapter 3

Modeling blurred video with layers

3.1 Introduction

To make the concept of layers more concrete, and to provide a motivating example of how reasoning about processes in the real world can benefit the computation of optical flow in challenging scenarios, we first consider the case of motion blur in videos[1]. Motion blur is one of the most common degradations of both still images and video, and makes not only optical flow computation harder, but also other tasks such as the segmentation of images into regions corresponding to objects. Yet, when a dynamic scene is captured by a camera with finite shutter speed, motion blur will always be present.

In such a setting, traditional assumptions of brightness constancy are violated, particularly at motion boundaries where the pixel values combine information from multiple surfaces blurring into each other. This is unfortunate since accurate motion boundaries are one of the most important properties of the scene that an optical flow algorithm helps to detect. Additionally, even for a single surface, optical flow estimation in the presence of motion blur in itself is hard, since the apparent (blurred) structure of the surface changes at every frame, depending on the motion itself. As a result, the performance of current optical flow algorithms decreases in the presence of motion blur [54].

To address these problems, we propose a novel layered model of images that explicitly models the motion of the layers, the blur induced by this motion, and the un-blurred appearance of each layer (Fig. 3.1). A key observation is that both the motion blur and the displacement of a surface are results of the same process in the world: the motion of the surface relative to the camera. Hence, the motion blur of a surface is completely determined by the motion of that surface – *estimating the optical flow gives us the blur kernel.* Unlike previous work we formulate this as a generative model and jointly solve for all unknowns. This produces an accurate

[1]This chapter is based on [277]. We thank D. Sun for discussions on optical flow, T. Adelson for insights into motion blur and layers, and R. Zavada for the JFK video.

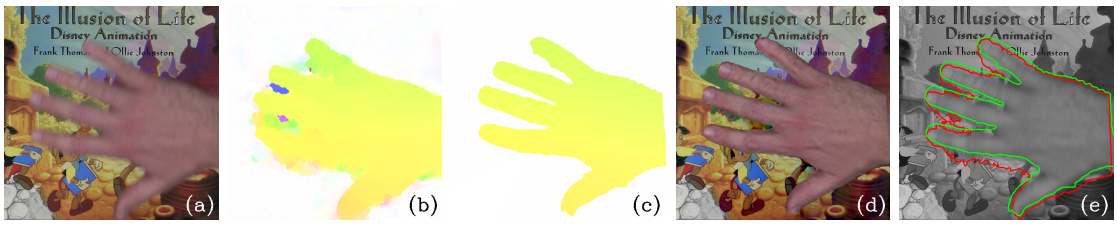


Figure 3.1: When computing optical flow from motion blurred video (a), existing methods [282] fail at object boundaries ((b) and (e), red). Our method is able to accurately estimate optical flow (c), deblurred frames (d), and object boundaries ((e), green).

layer segmentation and precise motion boundaries from a motion-blurred image sequence (Fig. 3.1(c) and (e)). To this end, layers provide a natural framework because they directly model surface interaction at occlusion boundaries. Here we focus on parametric motion within layers and use a two-layer model, as is common in recent approaches [231]. While being limited in terms of motion complexity, we nevertheless find that this model is able to analyze real scenes with different foreground and background motion.

As described in Sec. 2.2.2, much of the work on motion blur focuses on the problem of *deblurring*, particularly in single images where the blur is caused by camera shake [81, 153, 67]. These approaches either assume homogeneous blur across the whole image or restrict the blur kernels to those caused by common camera motion paths. On the other hand, existing work on deblurring in the case of object motion is restricted to the case of static backgrounds [25]. Beside these limitations, deblurring methods do not make use of the key fact that both the blur and the optical flow arise from the same process in the world, that is, the movement of the world relative to the camera. We do not address single-image deblurring, but focus on optical flow estimation in sequences with motion blur and motion of both the foreground and the background. In particular, we deal with spatially-varying blur kernels that are determined by the layer motions.

Closely related to our work are recent works by Schoenemann & Cremers [211] and Kumar *et al.* [191]. Schoenemann & Cremers [211] propose a layered framework for the task of super-resolution, in particular for removing focus blur. Their algorithm uses a video sequence as input but does not consider motion blur; consequently, the blur kernels have to be modeled explicitly. Furthermore, unlike ours, their model does not reason about regions of overlapping blur near layer boundaries. In [191] the authors segment the video into layers, estimate the appearance of layers, and model motion blur in the estimation of flow. Our method differs in several important ways. First, they estimate the appearance of a layer as the mean of the aligned image pixels within the layer. This process *does not model how the appearance is blurred by motion* and consequently cannot deblur the appear-

ance. Second, they *model the blur of each layer independently.* This ignores the critical effect of blur at layer boundaries where the appearance of two elements of the scene are combined. This further means that information about motion blur is not properly incorporated into the segmentation of the layers. Third, their method for estimating flow relies on normalized cross correlation while our method is fully generative, modeling the full appearance of each image from the model. Fourth, they do not directly parameterize the blur kernel based on the motion. In contrast, our explicit parameterization of blur facilitates a simpler unified formulation and optimization scheme.

Figure 3.2 illustrates how a scene is generated as a composition of layers, each of which is *individually* warped and blurred by its motion. This compositing from layers that are independently blurred captures what happens at boundaries while simplifying optimization compared with previous work. We explicitly model the blur as a function of the estimated motion, resulting in an elegant formulation of the problem. Thus, given the estimated motion, the blur within each layer is known, and the latent, sharp, appearance of each layer can be reconstructed. As a result we can reconstruct accurate motion at the boundaries, as well as deblurred estimates for both layers.

To summarize, this chapter demonstrates a way to estimate optical flow in video sequences in the presence of multiple motions and motion blur. We treat motion blur in a layered framework, allowing us to simultaneously infer the sharp layer segmentation, the object motion, the corresponding motion blur, and the latent (deblurred) object appearances. Our formulation is the first fully generative model of blurred video sequences using a layered framework. We demonstrate the effectiveness of the approach using synthetic and real sequences containing multiple motions. In addition to improving optical flow accuracy, we can deblur sequences that previous methods cannot handle, and show accurate estimation of layer boundaries despite heavy motion blur. We show how it is robust to noise by modeling a degraded sequence of the assassination of John F. Kennedy.

3.2 A Layered Representation of Motion-Blurred Video

3.2.1 Notation

A superscript denotes a layer l, $0 \leq l \leq L-1$, where larger l's are closer to the observer. Here we use a simplified model with $L = 2$ layers. A subscript denotes the image frame at time t, $1 \leq t \leq T$, for a sequence with T frames. $I_t \in \mathbb{R}^{H \times W \times 3}$ is an observed color image, $\mathbf{a}^l \in \mathbb{R}^{H \times W \times 3}$ is the unblurred color "appearance" of layer l. $\mathbf{g}^l \in \mathbb{R}^{H \times W \times 3}$ is the segmentation mask for l; for $l = 1$ (*i.e.* the background) the mask is assumed to be uniformly one. While $\mathbf{g}^l$ does not necessarily have to

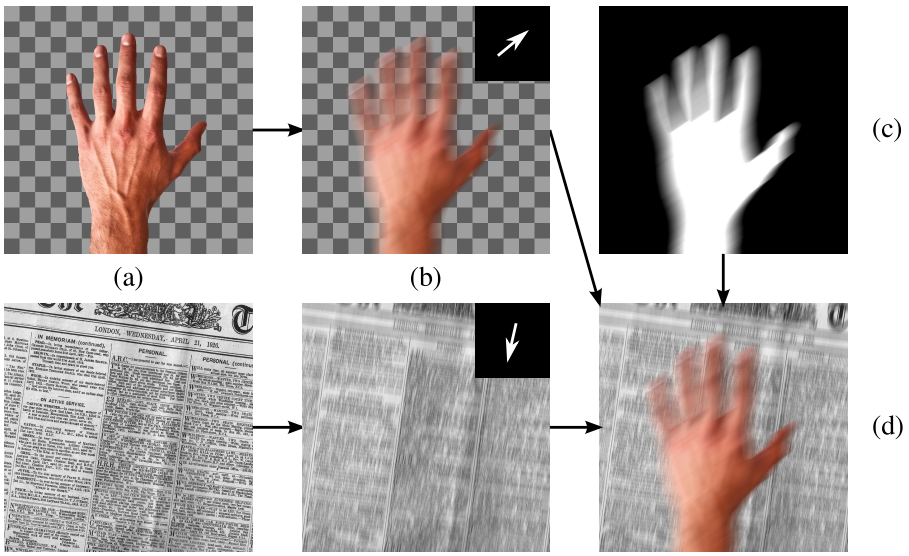


Figure 3.2: Generative Model. The sharp layers (a) are blurred (b) using the motion indicated in the top right corners. Together with the blurred layer segmentation (c), an image (d) with complex spatially-varying motion blur is generated. The checkerboard-pattern indicates transparency. Image inspired by [260].

be binary, here we only consider opaque layers. To make the dimensionalities of $\mathbf{a}^l$ and $\mathbf{g}^l$ equal, we define $\mathbf{g}^l$ to have three color channels, but enforce all three to be equal for a given pixel. This assumption could be relaxed to allow tinted layers, *i.e.* different absorption rates of different wavelengths. $\mathbf{a}^l$ and $\mathbf{g}^l$ are assumed to be constant across the sequence. For longer sequences, this limitation could either be relaxed, or the sequence could be split into smaller sub-sequences, each exhibiting approximately constant layer shape and layer appearance. Since we formulate our model in a linear algebra framework, we use the column-vectorized forms of I_t, $\mathbf{a}^l$, and $\mathbf{g}^l$; *i.e.* $\bar{\mathbf{i}}_t \in \mathbb{R}^{3HW}$, $\bar{\mathbf{a}}^l \in \mathbb{R}^{3HW}$, and $\bar{\mathbf{g}}^l \in \mathbb{R}^{3HW}$, respectively where it is appropriate.

$\boldsymbol{\theta}_t^l$ are the transformation (*i.e.* motion) parameters for layer l at frame t. The complexity of $\boldsymbol{\theta}_t^l$ depends on the motion model, and can range from a single (u_1, u_2) value pair in case of purely translational motion to displacement values for every pixel in the case of dense optical flow. Here we focus on affine motion. It provides a good middle ground by capturing the most common frame-to-frame transformations while still being a linear transformation, and is hence the transformation that is usually chosen for layered approaches [231, 268]. While more complicated motion models such as a full homography are possible to include in our model, perspective effects from frame to frame in a video are often negligible, making affine motion a good approximation. In Sec. 3.4.2, we show how our algorithm behaves if the

assumption of purely affine motion is violated. Note that $\boldsymbol{\theta}_t^l$ does not contain frame-to-frame motion parameters such as motion vectors, but instead the absolute transformation parameters from a fixed reference frame (usually the first frame in a sequence) to the frame at time t.

Figure 3.3: From top to bottom: Bike, Sign, Kennedy sequence. From left to right: deblurred frame, reconstructed foreground layer, reconstructed background layer. The red frame indicates the extent of the current image.

To cope with object areas leaving the visible frame, $W \times H$ is set to be larger than the observation data by padding in all directions. With this simple approach, we are able to reconstruct objects leaving the frame, and even reconstruct a panorama in case of a slight camera sweep. Figure 3.3 shows examples for the panoramas that our method generates.

3.2.2 A Single Layer with Motion Blur

To fix ideas, first consider the simplest case of motion blur, in which a single layer without self-occlusions undergoes an arbitrary transformation during the period of open shutter. During the period of open shutter, every point on the surface traverses on a continuous path, and every pixel of the output image is in turn traversed by a number of input surface points. A pixel in the blurred image can therefore be approximated as a linear combination of pixels of the unblurred layer [213]. Thus, we can model the blur as a blur matrix $\mathbf{H}\left(\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t-1}, \eta\right) \in \mathbb{R}^{3WH \times 3WH}$, and write the blurred image as

$$\bar{\mathbf{i}}_{singlelayer} = \mathbf{H}\left(\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t-1}, \eta\right) \bar{\mathbf{a}}. \tag{3.1}$$

$\mathbf{H}$ depends on the affine parameters of the current and the previous frame, as well as the shutter speed η, and models the motion blur induced by the motion between frame $t-1$ and t. In this work, we assume the shutter speed to be known and constant; this is a reasonable assumption for digital video cameras and even archival film footage as we show in the experiments. $\mathbf{H}$ is set to influence different color channels equally.[2]

3.2.3 Two Layers without Motion Blur

Without motion blur, an image that is composed of two different layers with appearances $\bar{\mathbf{a}}^0$ and $\bar{\mathbf{a}}^1$, where $\bar{\mathbf{a}}^0$ denotes the background, can be written as

$$\bar{\mathbf{i}}_{noblur} = \left(\mathbf{1} - \bar{\mathbf{g}}^1\right) \odot \bar{\mathbf{a}}^0 + \bar{\mathbf{a}}^1. \tag{3.2}$$

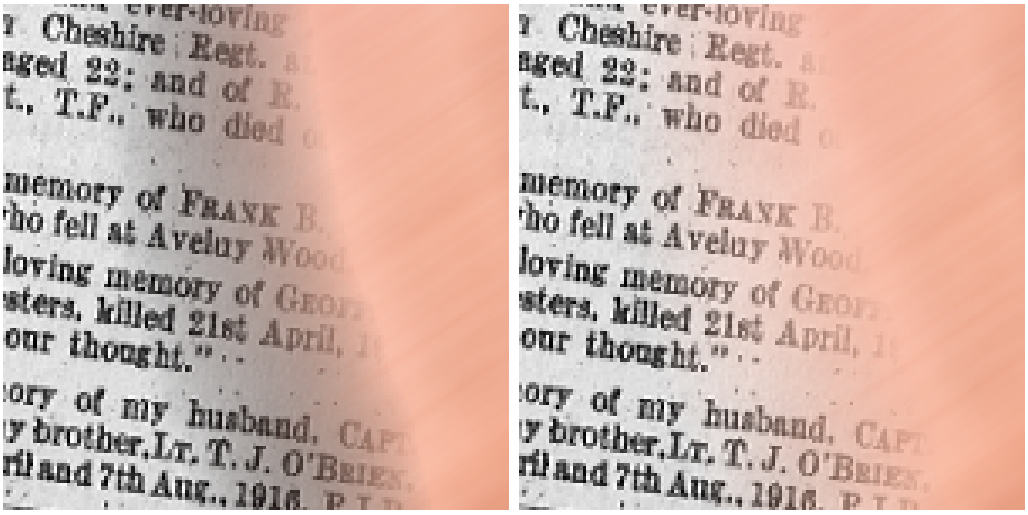
Here, $\odot$ denotes element-wise multiplication, and $\bar{\mathbf{g}}^1$ is the layer segmentation of $\bar{\mathbf{a}}^1$, and, in the case of a non-transparent $\bar{\mathbf{a}}^1$, should be binary. Similar to [25], we enforce $\bar{\mathbf{a}}^1$ to be zero everywhere where the corresponding segmentation $\bar{\mathbf{g}}^1$ is zero. However, different from normal matting approaches, we do not weigh $\bar{\mathbf{a}}^1$ by $\bar{\mathbf{g}}^1$ in the layer composition (3.2). The reason for this will become apparent once we consider the case of motion blur.

3.2.4 Two Layers with Foreground Motion and Blur

For readability, we abbreviate $\mathbf{H}^l_{t,t-1} = \mathbf{H}\left(\boldsymbol{\theta}^l_t, \boldsymbol{\theta}^l_{t-1}, \eta\right)$. Assuming a static background, we get

$$\bar{\mathbf{i}}_{fg-blur} = \left(\mathbf{1} - \mathbf{H}^1_{t,t-1}\bar{\mathbf{g}}^1\right) \odot \bar{\mathbf{a}}^0 + \mathbf{H}^1_{t,t-1}\bar{\mathbf{a}}^1. \tag{3.3}$$

[2] To perform super resolution, $\mathbf{a}$ would be larger than I and we would multiply by another matrix to decimate the blurred $\mathbf{a}$ in generating I.



(a) Weighting the foreground by its mask

(b) Using Eq. (3.3)

Figure 3.4: Difference in blending models (see text).

Now, it becomes understandable why in Eq. (3.2) the foreground layer is not weighted by its mask, but instead set to be zero outside of the mask region. In the weighted case, the effect of (3.3) would be a blurring of both the foreground $\bar{\mathbf{a}}^1$ *and* the mask $\bar{\mathbf{g}}^1$. Note that the blurred mask $\mathbf{H}\bar{\mathbf{g}}^1 \in [0, 1]$ and, hence, multiplying by $\mathbf{H}\bar{\mathbf{g}}^1$ attenuates the appearance (*i.e.* makes it darker when $\mathbf{H}\bar{\mathbf{g}}^1 < 1$). Thus, foreground pixels in the blurred transition region would be doubly attenuated, resulting in a fade-to-black effect even in the case of bright backgrounds (see Fig. 3.4).

To increase readability and flexibility, from here on we use a vector-matrix notation instead of the convolution-based notation from Eq. (3.3). Now considering multiple points t in time, Eq. (3.3) becomes

$$\bar{\mathbf{i}}_{fg-blur,t} = \left(\mathbf{1} - \mathbf{H}^1_{t,t-1}\mathbf{T}^1_t\bar{\mathbf{g}}^1\right) \odot \bar{\mathbf{a}}^0 + \mathbf{H}^1_{t,t-1}\mathbf{T}^1_t\bar{\mathbf{a}}^1. \tag{3.4}$$

In addition to the blur matrices $\mathbf{H}^l_{t,t-1}$, we use the transformation matrices $\mathbf{T}^l_t \in \mathbb{R}^{3WH\times 3WH}$ to transform a vectorized image according to the transformation parameters $\boldsymbol{\theta}^l_t$.

In the special case of pure translation (*i.e.* spatially invariant motion and motion blur), $\mathbf{H}$ and $\mathbf{T}$ are simple banded matrices, and the matrix multiplications in (3.4) are equivalent to convolutions and shifts of the corresponding images. However, since *any* blur and transformation can be expressed as linear (re-)combination of pixels in the image[3], this formulation is very flexible, allowing complex motion

[3]Assuming that the latent images are extended appropriately, so that reasoning about pixels outside of the frames is possible.

models such as homographies or dense optical flow [213].

3.2.5 A Two-Layer Model

While algorithms dealing with spatially-varying blur commonly assume a static background [25, 55], this assumption is often invalid in practice; eg. in the presence of camera motion in addition to object motion. Incorporating background motion and its corresponding blur into Eq. (3.4) yields an estimated image

$$\hat{\mathbf{i}}_t = \left(\mathbf{1} - \mathbf{H}^1_{t,t-1}\mathbf{T}^1_t\bar{\mathbf{g}}^1\right) \odot \mathbf{H}^0_{t,t-1}\mathbf{T}^0_t\bar{\mathbf{a}}^0 + \mathbf{H}^1_{t,t-1}\mathbf{T}^1_t\bar{\mathbf{a}}^1. \tag{3.5}$$

This is our generative model for a two-layer image in the presence of motion blur in both layers.

3.2.6 Data likelihood

To allow the estimation of pixels leaving the frame, we pad the input images I_t by a fixed amount, depending on the expected camera motion. Within the visible area, the image $\hat{\mathbf{i}}_t$ generated by our model should then match the vectorized observed image $\bar{\mathbf{i}}_t$. Hence to estimate the model parameters we minimize

$$E_D\left(\mathcal{I}, \mathcal{G}, \mathcal{A}, \mathbf{\Theta}\right) = \sum_t \bar{\mathbf{m}}^\top \rho\left(\bar{\mathbf{i}}_t - \hat{\mathbf{i}}_t\right) \tag{3.6}$$

summed over all pixels, where $\rho(x) = \sqrt{x^2 + \varepsilon^2}$ is a robust Charbonnier function [57]. Here, $\bar{\mathbf{m}} \in \mathbb{R}^{3WH}$ contains zeros in the outside padded area, and ones in the inside, restricting the summation to the visible part of the image. The input is the image sequence $\mathcal{I} = \left\{\bar{\mathbf{i}}_1, \cdots, \bar{\mathbf{i}}_T\right\}$. The estimated parameters are the set of segmentations (excluding the background), $\mathcal{G} = \{\bar{\mathbf{g}}^1\}$, the set of appearances, $\mathcal{A} = \{\bar{\mathbf{a}}^0, \bar{\mathbf{a}}^1\}$, and the set of transformation parameters, $\mathbf{\Theta} = \{\boldsymbol{\theta}^0_0, \boldsymbol{\theta}^1_0, \cdots, \boldsymbol{\theta}^0_T, \boldsymbol{\theta}^1_T\}$. For each layer, we obtain $T + 1$ values for $\boldsymbol{\theta}$, since each frame $\bar{\mathbf{i}}_t$, including the first, depends on $\boldsymbol{\theta}_t$ and $\boldsymbol{\theta}_{t-1}$. Here we restrict ourselves to two layers but discuss extensions to more layers in Sec. 3.6.

3.2.7 Regularization

To make Eq. (3.6) better behaved in weakly structured regions, we impose a number of regularization terms on the appearance maps $\mathbf{a}^{\{0,1\}}$ and the segmentation $\mathbf{g}^1$.

Spatial smoothness

A certain degree of smoothness is desirable, both in the estimated appearance maps and in the segmentation maps. In the former case, this prevents noise; in the latter,

it is obvious that at most pixels the gradient should be zero, because they are either completely in the foreground or completely in the background.

Hence, we regularize both the appearance images $\mathbf{a}^{\{0,1\}}$ and the segmentation maps $\mathbf{g}$. Like standard deblurring methods we model the fact that the spatial derivatives of natural images exhibit a heavy-tail distribution [153]. This can be captured using a sparse prior:

$$E_{sparse}(\mathbf{f}, \alpha) = \sum_{\mathbf{x}} |\nabla_x \mathbf{f}(\mathbf{x})|^\alpha + |\nabla_y \mathbf{f}(\mathbf{x})|^\alpha. \tag{3.7}$$

Consistent with natural image statistics, we use $\alpha_A = 0.8$ for the appearance maps. For a binary segmentation mask $\mathbf{g}$, we use the L^1 total variation prior, given by Eq. (3.7) by setting $\alpha_G = 1$. This prior prefers smooth contours, and has been successfully used in similar tasks before [25].

We approximate the non-differentiable absolute $|\cdot|$ in Eq. (3.7) with a Charbonnier function [57] with $\epsilon = 10^{-3}$.

Background preference

We assume the background to generally cover more pixels than the foreground layer. Thus, we impose a slight penalty for pixels that are assigned to the foreground.

$$E_{bg}(\mathbf{f}) = \sum_{\mathbf{x}} \mathbf{f}(\mathbf{x})^2. \tag{3.8}$$

3.2.8 Objective.

The final objective function is

$$E\left(\mathcal{I}, \mathcal{G}, \mathcal{A}, \mathbf{\Theta}\right) = E_D\left(\mathcal{I}, \mathcal{G}, \mathcal{A}, \mathbf{\Theta}\right) + E_{Reg}\left(\mathcal{G}, \mathcal{A}\right), \tag{3.9}$$

with

$$\begin{aligned} &E_{Reg}\left(\mathcal{G}, \mathcal{A}\right) = \\ &+ \lambda_{sparse,A}\left(E_{sparse}\left(\mathbf{a}^0, \alpha_A\right) + E_{sparse}\left(\mathbf{a}^1, \alpha_A\right)\right) \\ &+ \lambda_{sparse,G} E_{sparse}\left(\mathbf{g}^1, \alpha_G\right) \\ &+ \lambda_{bg} E_{bg}\left(\mathbf{g}^1\right) \ . \end{aligned} \tag{3.10}$$

We set the parameters to $\lambda_{bg} = 0.05$, $\lambda_{sparse,A} = 0.001$, $\lambda_{sparse,G} = 0.05$. The effects of changes to these parameters are investigated in Sec. 3.4.2.

3.2.9 Derivatives of objective

To minimize the energy (3.9), we will use a coordinate descent method (see Sec. 3.3.2). The required gradients of the energy (3.9) are given as follows.

Derivatives with respect to $\mathbf{a}^1$ and $\mathbf{a}^1$

We define $\mathbf{\Psi}_t = \psi\left(\bar{\mathbf{i}}_t - \hat{\mathbf{i}}_t\right)$ as a row vector, containing the element-wise derivatives of the point-wise error measure $\rho(\mathbf{x})$ with respect to the individual elements:

$$\psi(\mathbf{x}) = \bar{\mathbf{m}}^\top \frac{\partial \rho(\mathbf{x})}{\partial \mathbf{x}}, \tag{3.11}$$

with $\bar{\mathbf{m}}$ being the mask to extend the image (cf. Eq. (3.6)).

This gives the derivative of Eq. (3.9) with respect to the appearances $\mathbf{a}^0$, $\mathbf{a}^1$ as

$$\begin{aligned}\frac{\partial E}{\partial \bar{\mathbf{a}}^0} &= \frac{\partial E_D}{\partial \bar{\mathbf{a}}^0} + \frac{\partial E_{Reg}}{\partial \bar{\mathbf{a}}^0} \\ &= -\sum_t \left[\mathbf{\Psi}_t \odot \left(\mathbf{1}^\top - \left(\mathbf{H}^1_{t,t-1}\mathbf{T}^1_t \bar{\mathbf{g}}^1\right)^\top\right)\right] \mathbf{H}^0_{t,t-1}\mathbf{T}^0_{t,t-1} \\ &\quad + \lambda_{sparse,A} \frac{\partial E_{sparse}(\bar{\mathbf{a}}^0, \alpha_A)}{\partial \bar{\mathbf{a}}^0}\end{aligned} \tag{3.12}$$

$$\begin{aligned}\frac{\partial E}{\partial \bar{\mathbf{a}}^1} &= \frac{\partial E_D}{\partial \bar{\mathbf{a}}^1} + \frac{\partial E_{Reg}}{\partial \bar{\mathbf{a}}^1} \\ &= -\sum_t \mathbf{\Psi}_t \mathbf{H}^1_{t,t-1}\mathbf{T}^1_t + \lambda_{sparse,A} \frac{\partial E_{sparse}(\bar{\mathbf{a}}^1, \alpha_A)}{\partial \bar{\mathbf{a}}^1},\end{aligned} \tag{3.13}$$

where the derivative of the spatial smoothness regularization (3.7) is given as

$$\begin{aligned}\left.\frac{\partial E_{sparse}(\mathbf{f}, \alpha)}{\partial \mathbf{f}}\right|_{\mathbf{x}} =& \nabla_x^- \left[\alpha \rho_c \left(\nabla_x \mathbf{f}(\mathbf{x})\right)^{\alpha-1} \psi_c\left(\nabla_x \mathbf{f}(\mathbf{x})\right)\right] \\ &+ \nabla_y^- \left[\alpha \rho_c \left(\nabla_y \mathbf{f}(\mathbf{x})\right)^{\alpha-1} \psi_c\left(\nabla_y \mathbf{f}(\mathbf{x})\right)\right].\end{aligned} \tag{3.14}$$

Here, the gradient ∇ is approximated by a finite difference filter $[0, -1, 1]$, and the inverted gradient ∇^- is approximated by $[1, -1, 0]$. As in Eq. (3.7), we use the Charbonnier $\rho_c(z) = \sqrt{z^2 + \epsilon^2}$ with $\epsilon = 10^{-3}$ and the derivative

$$\psi_c(z) = \frac{\partial \rho_c(z)}{\partial z} = \frac{z}{\sqrt{z^2 + \epsilon^2}}. \tag{3.15}$$

to approximate the absolute value.

Derivative with respect to $\mathbf{g}^1$

To optimize the discrete-valued segmentation mask $\bar{\mathbf{g}}^1$, we approximate it as $\bar{\mathbf{g}}^1 \approx k\left(\tilde{\mathbf{g}}^1\right)$, with the element-wise heavy-side function $k(\tilde{\mathbf{g}}^1) = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{\tilde{\mathbf{g}}^1 - 0.5}{\sigma_k}\right)$ and $\sigma_k = 0.05$. The optimization is then carried out with respect to $\tilde{\mathbf{g}}^1$. Using the same notation as above, the derivative of the objective function with respect to $\tilde{\mathbf{g}}^1$ is given as

$$\begin{aligned}
\frac{\partial E}{\partial \tilde{\mathbf{g}}^1} &= \left(\frac{\partial E_D}{\partial \bar{\mathbf{g}}^1} + \frac{\partial E_{Reg}}{\partial \bar{\mathbf{g}}^1}\right)\frac{\partial \mathbf{q}}{\partial \tilde{\mathbf{g}}^1} \\
&= \left(\sum_t \left[\boldsymbol{\Psi}_t \odot \left(\left(\mathbf{H}^0_{t,t-1}\mathbf{T}^0_t\bar{\mathbf{a}}^0\right)^\top\right)\right]\mathbf{H}^1_{t,t-1}\mathbf{T}^1_{t,t-1}\right. \\
&\quad \left. + \lambda_{sparse,G}\frac{\partial E_{sparse}\left(\bar{\mathbf{g}}^1, \alpha_G\right)}{\partial \bar{\mathbf{g}}^1} + \lambda_{bg}\frac{\partial E_{bg}\left(\bar{\mathbf{g}}^1\right)}{\partial \bar{\mathbf{g}}^1}\right)\frac{\partial k}{\partial \tilde{\mathbf{g}}^1}.
\end{aligned} \tag{3.16}$$

Note that, since $k(\tilde{\mathbf{g}}^1)$ operates element-wise, only the diagonal entries of $\frac{\partial k}{\partial \tilde{\mathbf{g}}^1} \in \mathbb{R}^{3HW\times 3HW}$ are non-zero, and contain the element-wise derivatives of $k(\tilde{\mathbf{g}}^1)$.

The derivative of the spatial smoothness $\frac{\partial E_{sparse}}{\partial \bar{\mathbf{g}}^1}$ is the same as in Eq. (3.14), and the derivative of the background preference term (3.8) is given as

$$\left.\frac{\partial E_{bg}\left(\mathbf{f}\right)}{\partial \mathbf{f}}\right|_{\mathbf{x}} = 2\mathbf{f}(\mathbf{x}). \tag{3.17}$$

Derivatives with respect to Θ

Intuitively, we see that $\hat{I}_t$ is only a function of $\boldsymbol{\theta}_t^{\{0,1\}}$ and $\boldsymbol{\theta}_{t-1}^{\{0,1\}}$. In the affine case, $\boldsymbol{\theta}_t^{\{0,1\}} \in \mathbb{R}^6$. For clarity, we here give the derivatives with respect to a single components $\theta_{t,(c)}$ of $\boldsymbol{\theta}_t = \left(\theta_{t,(1)} \cdots \theta_{t,(C)}\right)^\top$.

$$\begin{aligned}
\frac{\partial E}{\partial \theta^0_{t,(c)}} &= \delta\left(t > 0\right)\left\{ -\boldsymbol{\Psi}_t \right. \\
&\qquad \left.\left[\left(\mathbf{1} - \mathbf{H}^1_{t,t-1}\mathbf{T}^1_t\bar{\mathbf{g}}^1\right) \odot \left(\frac{\partial \mathbf{H}^0_{t,t-1}}{\partial \theta^0_{t,(c)}}\mathbf{T}^0_t\bar{\mathbf{a}}^0 + \mathbf{H}^0_{t,t-1}\frac{\partial \mathbf{T}^0_t}{\partial \theta^0_{t,(c)}}\bar{\mathbf{a}}^0\right)\right]\right\} \\
&\quad + \delta\left(t < T\right)\left\{ -\boldsymbol{\Psi}_{t+1} \right.
\end{aligned}$$

$$\left[\left(\mathbf{1} - \mathbf{H}^1_{t+1,t} \mathbf{T}^1_{t+1} \bar{\mathbf{g}}^1 \right) \odot \left(\frac{\partial \mathbf{H}^0_{t+1,t}}{\partial \theta^0_{t,(c)}} \mathbf{T}^0_{t+1} \bar{\mathbf{a}}^0 \right) \right] \Bigg\} \tag{3.18}$$

$$\begin{aligned} \frac{\partial E}{\partial \theta^1_{t,(c)}} \quad = \quad & \delta\left(t > 0\right) \Bigg\{ - \mathbf{\Psi}_t \left[\mathbf{H}^0_{t,t-1} \mathbf{T}^0_t \bar{\mathbf{a}}^0 \odot \left(\frac{\partial \mathbf{H}^1_{t,t-1}}{\partial \theta^1_{t,(c)}} \mathbf{T}^1_t \bar{\mathbf{g}}^1 + \mathbf{H}^1_{t,t-1} \frac{\partial \mathbf{T}^1_t}{\partial \theta^1_{t,(c)}} \bar{\mathbf{g}}^1 \right) \right. \\ & \left. + \frac{\partial \mathbf{H}^1_{t,t-1}}{\partial \theta^1_{t,(c)}} \mathbf{T}^1_t \bar{\mathbf{a}}^1 + \mathbf{H}^1_{t,t-1} \frac{\partial \mathbf{T}^1_t}{\partial \theta^1_{t,(c)}} \bar{\mathbf{a}}^1 \right] \Bigg\} \\ & + \delta\left(t < T\right) \Bigg\{ - \mathbf{\Psi}_{t+1} \\ & \left[\mathbf{H}^0_{t+1,t} \mathbf{T}^0_{t+1} \bar{\mathbf{a}}^0 \odot \frac{\partial \mathbf{H}^1_{t+1,t}}{\partial \theta^1_{t,(c)}} \mathbf{T}^1_{t+1} \bar{\mathbf{g}}^1 + \frac{\partial \mathbf{H}^1_{t+1,t}}{\partial \theta^1_{t,(c)}} \mathbf{T}^1_{t+1} \bar{\mathbf{a}}^1 \right] \Bigg\} \end{aligned} \tag{3.19}$$

with $\delta(x) = 1$ if the argument x is true, and $\delta(x) = 0$ otherwise.

Note that it is possible to explicitly construct the $\mathbf{H}$ and $\mathbf{T}$ matrices, since they are usually sparse. However, we found that in practice a finite difference approach achieves comparable performance, and has clear speed benefits. Therefore, we use finite differences in the optimization of Θ, and separately optimize over the rotational and translational parameters of the affine transformations.

3.3 Optimization

We assume that the shutter speed is known and that there are only two moving layers. From the given shutter speed and an input video sequence consisting of multiple frames, our algorithm computes the latent (unblurred) appearance of both the background and the foreground, the motion parameters of both, and the segmentation mask for the foreground. Figure 3.5 shows an input image, and how the solution changes after each step described here. Note that it is usually possible to reconstruct the parts of the background that are visible in at least one frame of the sequence.

3.3.1 Initialization

Since the objective function (3.9) is non-convex, a good initialization of $\mathcal{A}$, $\mathcal{G}$, and Θ is important. While standard optical flow estimates from a blurred sequence are not generally accurate enough to actually perform deblurring [183], they nevertheless provide a useful initial estimate of the motion. Thus, to initialize, we first compute dense optical flow using an off-the-shelf optical flow algorithm, MDP-Flow [282]

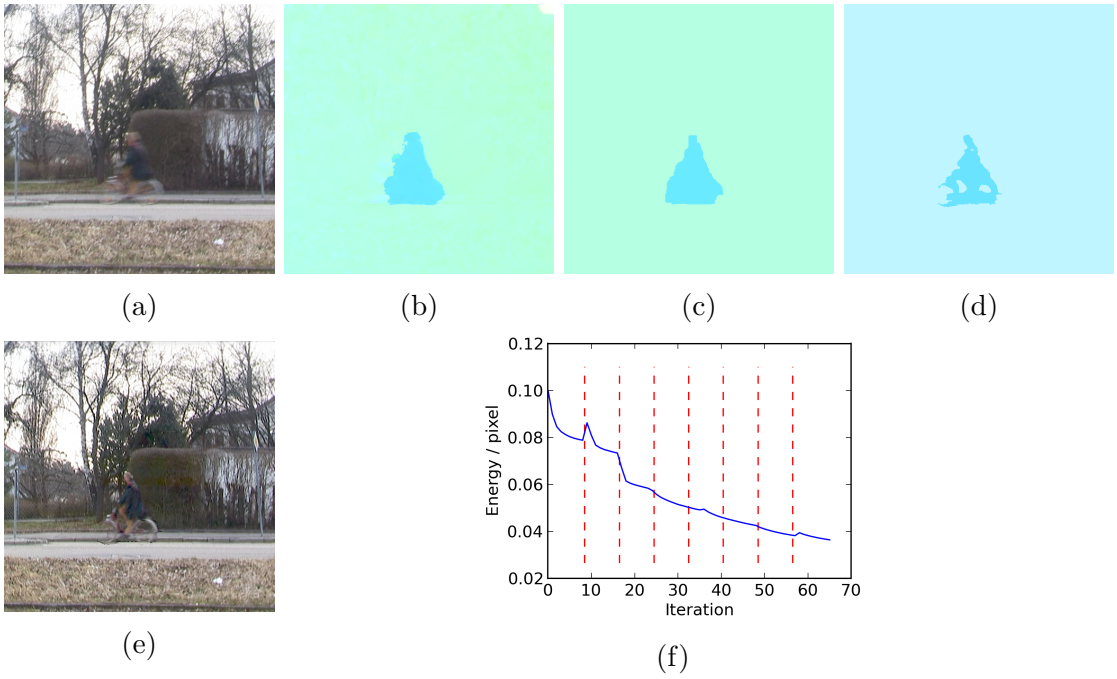


Figure 3.5: Illustration of different steps in the algorithm. (a) One of the 5 input frames in the sequence. (b) Initial flow, computed using [282]. (c) Motion initialization from optical flow. (d) Final motion estimate. (e) Deblurred image produced by the generative model. (f) Normalized energy vs. number of iterations. Red lines show transitions between the pyramid levels. See text for details.

(Fig. 3.5(b)). Note that the choice of initial optical flow algorithm is not critical, as long as it produces reasonable results. We tested a number of different optical flow algorithms, but did not observe significant differences in the end result. The reason for this is that the precise spatial configuration of the initial optical flow is less important than the extraction of the dominant motions, which current optical flow methods are generally capable of.

From the initial dense flow field, $L = 2$ dominant motions are robustly estimated using RANSAC, yielding the initial motion parameters for each frame. Additionally, this gives a per-frame pixel assignment estimate (*i.e.* foreground or background), similar to the approach used in [260] (Fig. 3.5(c)). Using the estimated motion parameters, the pixel assignments are aligned across all frames and added up. The result can be interpreted as an unnormalized foreground probability. We combine this with a spatial consistency term, and optimize via graph cuts [143], resulting in a single assignment estimate $\mathbf{g}$ for the whole sequence.

Given this segmentation and the estimated motion for each layer, we separate both layers by masking, and use a simple non-blind deblurring method [82] on each

layer to obtain initial deblurred estimates $\mathbf{a}$. While this initial deblurring causes ringing artifacts and is not strictly necessary, it speeds up convergence by providing a reasonable initialization.

3.3.2 Coordinate Descent

Starting with our initial estimate, we use an iterative, alternating optimization method. Empirically, we found this to work better than comparable optimizers such as L-BFGS [184]. We observe that the magnitude of the derivatives of Eq. (3.9) with respect to the different variables in our model differs strongly. Intuitively, one can see how changing one of the values in $\boldsymbol{\theta}$ influences both the layer displacement and blur, and thus causes a much larger energy variation than a slight change of a single pixel *e.g.* of $\mathbf{a}^1$. For this reason, and to keep the computation feasible, we optimize one variable at a time using gradient descent, but terminate the optimization after 3 iterations to avoid reaching local optima, and switch to the next variable. One optimization cycle over all variables makes up a single iteration. We iterate for at most 50 iterations, or until the relative change in energy falls below 1 percent.

To deal with large motions, we use a standard multi-scale approach with a Gaussian pyramid with P levels. We found $P = 7$ and a scaling factor of 1.5 to work well. The initialization is done at the highest resolution, and the estimated starting values are rescaled to the highest pyramid level. The complete optimization schedule is then performed at each pyramid level, and the results form the input of the next level. Figure 3.5(g) shows the energy per pixel vs. number of iterations. The red dashed lines show the transition points between pyramid levels. While the gradient descent scheme we use is not guaranteed to reach the global optimum, we nevertheless observe a well-behaved falloff.

After the optimization has converged, we obtain the final mask by smoothing and thresholding $\tilde{\mathbf{g}}^1$, and multiply the binary mask element-wise with $\mathbf{a}^1$ to compute the final foreground appearance estimate. Figure 3.5(d) shows the final estimated flow. Note that the segmentation is significantly improved, with even the shape of the person and rims of the bicycle being evident. A full composite with blur removed is shown in Fig. 3.5(e). Using unoptimized Python code, our algorithm takes around 35min per frame with a resolution of $H = W = 640$ pixels.

Algorithm 1 gives an overview of the complete optimization in Pseudocode. Here, `truncatedOptimize[`V`](`$\mathcal{P}$`)` denotes a truncated, *ie.* prematurely terminated optimization of variable V, given the parameters $\mathcal{P}$. We use a basic gradient descent optimization with line search, and terminate after 3 steps. `Energy` denotes the objective function, and unbracketed superscripts indicate the layer. Additionally, bracketed superscripts indicate the iteration number, bracketed subscripts the pyramid level. $\mathcal{A}$, $\mathcal{G}$, Θ indicate the set of appearances, segmentation maps, and parameters, over all layers, respectively, and Θ^l denotes the set of parameters of layer l over time. $\tilde{\mathcal{G}}$ denotes the set of relaxed, continuous layer segmentation masks, with

$\mathbf{g} = k(\tilde{\mathbf{g}})$, as described above.

3.4 Evaluation

3.4.1 Evaluation data

We evaluate the algorithm in terms of motion accuracy and deblurring performance. Furthermore, we compare the results of our algorithm with and without the blur model. We use a total of nine sequences, containing translational and affine motion. Three of these are synthetic sequences with moving background and foreground: `Elephant` contains a roughly circular translating motion; `Desert` contains a foreground with fine details, motion with almost constant direction, and an unstructured background; and `Market` contains a foreground object that strongly resembles the background. For all synthetic sequences, we applied a gamma correction with $\gamma = 2.2$ after compositing.

We also use six real video sequences: in `Sign`, the camera was deliberately moved but the scene was static; `Magazine` contains a planar rotational motion in front of a static background; `Hand` contains a hand undergoing affine motion, and `Bike` and `Car` contain a moving object and were filmed with a hand-held camera, resulting in slight camera shake. Furthermore, we test our method with historical footage from the assassination of John F. Kennedy (sequence `JFK`); this historical challenge is described in Sec. 3.4.5.

The length of the sequences varies from 5 to 10 frames. Two of our test sequences contain static backgrounds (`Hand` and `Magazine`), while the remaining 7 contain moving foreground and background. Three of the test sequences contain significant affine motions including scale change and rotation. Figure 3.6 shows an overview over the sequences and qualitative results for each.

3.4.2 Algorithm behavior

Parameter settings

For all sequences, we set the default parameter values to $\lambda_{bg} = 0.05$, $\lambda_{sparse,G} = 0.05$, and $\lambda_{sparse,A} = 0.001$. To analyze the sensitivity to variations of a parameter, we run our algorithm with different values within a range around the default parameter values. All other parameters are kept constant.

Figure 3.7 shows the results for $\lambda_{bg} \in \{0.01, 0.05, 0.1\}$, Fig. 3.8 shows the results for $\lambda_{sparse,G} \in \{0.01, 0.05, 0.1\}$, and Fig. 3.9 the results for $\lambda_{sparse,A} \in \{0, 0.001, 0.1\}$. The respective parameter always varies across columns. In the first row, we show a synthetic sequence (`Desert`) and in the second row a real sequence (`Bike`), illustrating the robustness of the results to changes of the individual weights. For λ_{bg} and $\lambda_{sparse,G}$ we show the segmented motion field, since changing these parameters

Algorithm 1 Optimization algorithm in Pseudocode.

Require: $\mathcal{I}, \mathcal{A}^{(init)}, \tilde{\mathcal{G}}^{(init)}, \Theta^{(init)}$

$\mathcal{A}^{(0)}_{(0)} \leftarrow \texttt{ScaleToPyramidLevel}\left(\mathcal{A}^{(init)}, 0\right)$

$\mathcal{G}^{(0)}_{(0)} \leftarrow \texttt{ScaleToPyramidLevel}\left(\mathcal{G}^{(init)}, 0\right)$

$\tilde{\mathcal{G}}^{(0)}_{(0)} \leftarrow \mathcal{G}^{(0)}_{(0)}$

$\Theta^{(0)}_{(0)} \leftarrow \texttt{ScaleToPyramidLevel}\left(\Theta^{(init)}, 0\right)$

$p \leftarrow 0$

while $p < P$ **do**

$\mathcal{I}_{(p)} \leftarrow \texttt{ScaleToPyramidLevel}\left(\mathcal{I}, p\right)$

$E^{(0)}_{(p)} \leftarrow \texttt{Energy}\left(\mathbf{a}^{0,(0)}_{(p)}, \mathbf{a}^{1,(0)}_{(p)}, \mathbf{g}^{1,(0)}_{(p)}, \Theta^{0,(0)}_{(p)}, \Theta^{1,(0)}_{(p)}\right)$

$i \leftarrow 0$

repeat

$\mathbf{a}^{0,(i+1)}_{(p)} \leftarrow \texttt{truncatedOptimize}\left[\mathbf{a}^{0}_{p}\right]\left(\mathbf{a}^{0,(i)}_{(p)}, \mathbf{a}^{1,(i)}_{(p)}, \mathbf{g}^{1,(i)}_{(p)}, \Theta^{0,(i)}_{(p)}, \Theta^{1,(i)}_{(p)}\right)$

$\Theta^{0,(i+1)}_{(p)} \leftarrow \texttt{truncatedOptimize}\left[\Theta^{0}_{p}\right]\left(\mathbf{a}^{0,(i+1)}_{(p)}, \mathbf{a}^{1,(i)}_{(p)}, \mathbf{g}^{1,(i)}_{(p)}, \Theta^{0,(i)}_{(p)}, \Theta^{1,(i)}_{(p)}\right)$

$\tilde{\mathbf{g}}^{1,(i+1)}_{(p)} \leftarrow \texttt{truncatedOptimize}\left[\tilde{\mathbf{g}}^{1}_{p}\right]\left(\mathbf{a}^{0,(i+1)}_{(p)}, \mathbf{a}^{1,(i)}_{(p)}, \tilde{\mathbf{g}}^{1,(i)}_{(p)}, \Theta^{0,(i+1)}_{(p)}, \Theta^{1,(i)}_{(p)}\right)$

$\mathbf{g}^{1,(i+1)}_{(p)} \leftarrow k\left(\tilde{\mathbf{g}}^{1,(i+1)}_{(p)}\right)$

$\mathbf{a}^{1,(i+1)}_{(p)} \leftarrow \texttt{truncatedOptimize}\left[\mathbf{a}^{1}_{p}\right]\left(\mathbf{a}^{0,(i+1)}_{(p)}, \mathbf{a}^{1,(i)}_{(p)}, \mathbf{g}^{1,(i+1)}_{(p)}, \Theta^{0,(i+1)}_{(p)}, \Theta^{1,(i)}_{(p)}\right)$

$\Theta^{1,(i+1)}_{(p)} \leftarrow \texttt{truncatedOptimize}\left[\Theta^{1}_{p}\right]\left(\mathbf{a}^{0,(i+1)}_{(p)}, \mathbf{a}^{1,(i+1)}_{(p)}, \mathbf{g}^{1,(i+1)}_{(p)}, \Theta^{0,(i+1)}_{(p)}, \Theta^{1,(i)}_{(p)}\right)$

$E^{(i+1)}_{(p)} \leftarrow \texttt{Energy}\left(\mathbf{a}^{0,(i+1)}_{(p)}, \mathbf{a}^{1,(i+1)}_{(p)}, \mathbf{g}^{1,(i+1)}_{(p)}, \Theta^{0,(i+1)}_{(p)}, \Theta^{1,(i+1)}_{(p)}\right)$

$i \leftarrow i + 1$

until $i =$ maxiter or $\frac{E^{(i+1)}_{(p)} - E^{(i)}_{(p)}}{E^{(i)}_{(p)}} < \delta$

$\mathcal{A}^{(0)}_{(p+1)} \leftarrow \texttt{ScaleToPyramidLevel}\left(\mathcal{A}^{(i)}_{(p)}, p+1\right)$

$\tilde{\mathcal{G}}^{(0)}_{(p+1)} \leftarrow \texttt{ScaleToPyramidLevel}\left(\tilde{\mathcal{G}}^{(i)}_{(p)}, p+1\right)$

$\mathcal{G}^{(0)}_{(p+1)} \leftarrow k\left(\tilde{\mathcal{G}}^{(0)}_{(p+1)}\right)$

$\Theta^{(0)}_{(p+1)} \leftarrow \texttt{ScaleToPyramidLevel}\left(\Theta^{(i)}_{(p)}, p+1\right)$

$p \leftarrow p + 1$

end while

$\mathcal{A}^{(opt)} \leftarrow \mathcal{A}^{(0)}_{(P)}$

$\mathcal{G}^{(opt)} \leftarrow \mathcal{G}^{(0)}_{(P)}$

$\Theta^{(opt)} \leftarrow \Theta^{(0)}_{(P)}$

return $\mathcal{A}^{(opt)}, \mathcal{G}^{(opt)}, \Theta^{(opt)}$

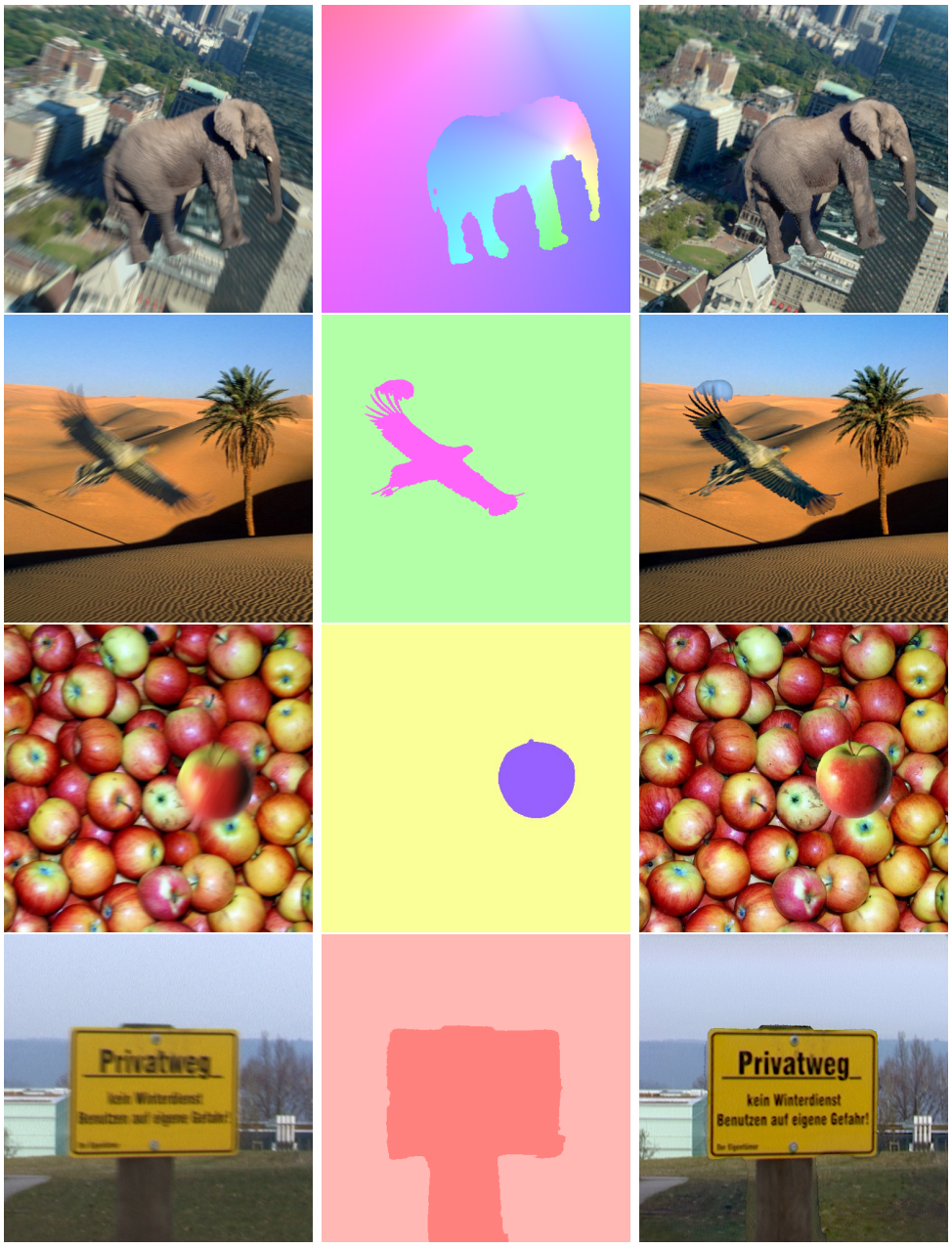


Figure 3.6: Results. From left to right: Single frame of the input sequence, computed optical flow, computed deblurred appearance. From top to bottom: `Elephant`, `Desert`, `Market`, `Sign`.

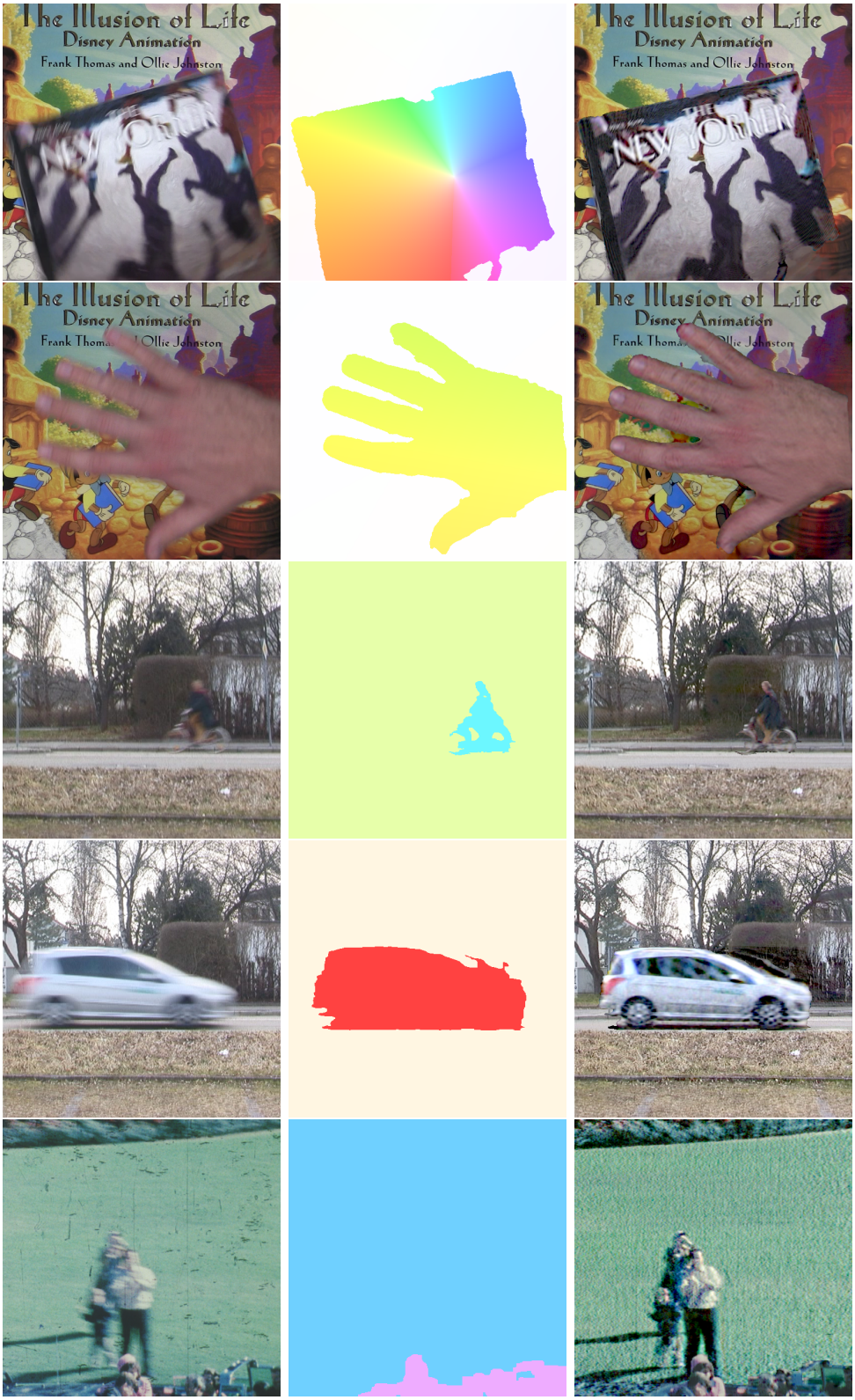


Figure 3.6: Results (continued). From top to bottom: `Magazine`, `Hand`, `Bike`, `Car`, `JFK`.

primarily affects the segmentation. For $\lambda_{sparse,A}$ we show the deblurred images. As shown, the results are robust to changes of the parameters within reasonable ranges. Small differences can be seen in the very fine details, such as the front wheel of the bicycle. However, they have very little effect on the results overall.

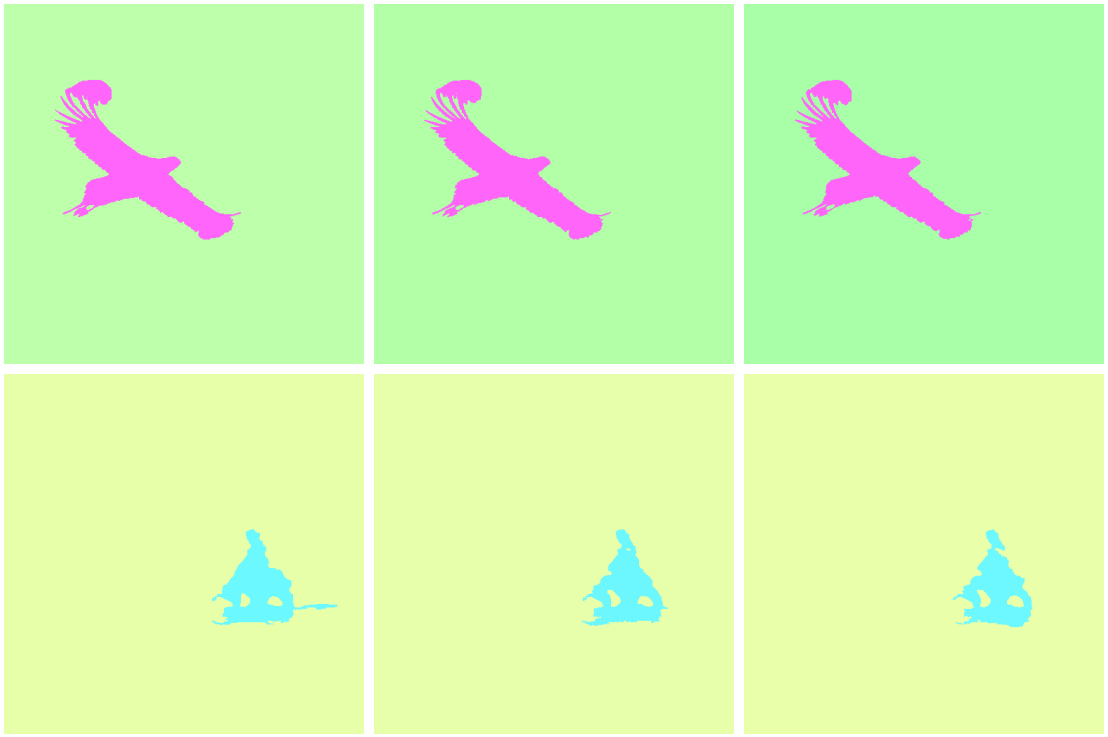


Figure 3.7: Left column: $\lambda_{bg} = 0.01$. Middle column: $\lambda_{bg} = 0.05$. Right column: $\lambda_{bg} = 0.1$. We find the results to be robust to small changes of λ_{bg}. Differences are visible in small details, primarily around the bicycle (bottom row). For all sequences, we chose $\lambda_{bg} = 0.05$.

Effect of out-of-plane rotations

To test the effects perspective transformations, we used a simple affine sequence with a rotating background and a zooming foreground, and added an out-of-plane rotation of gradually increasing magnitude, from 0 to 30 degrees over a sequence length of 8 frames. The focal length is kept short in order to create perspective foreshortening. Figure 3.10 shows the resulting errors of flow and the image reconstruction, and a linear fit to both.

The effect are as expected: With increasing out-of-plane rotation, the flow error generally increases, while the PSNR decreases. However, note that the overall impact is fairly low. When going from 0 to 30 degrees out-of-plane rotation, the

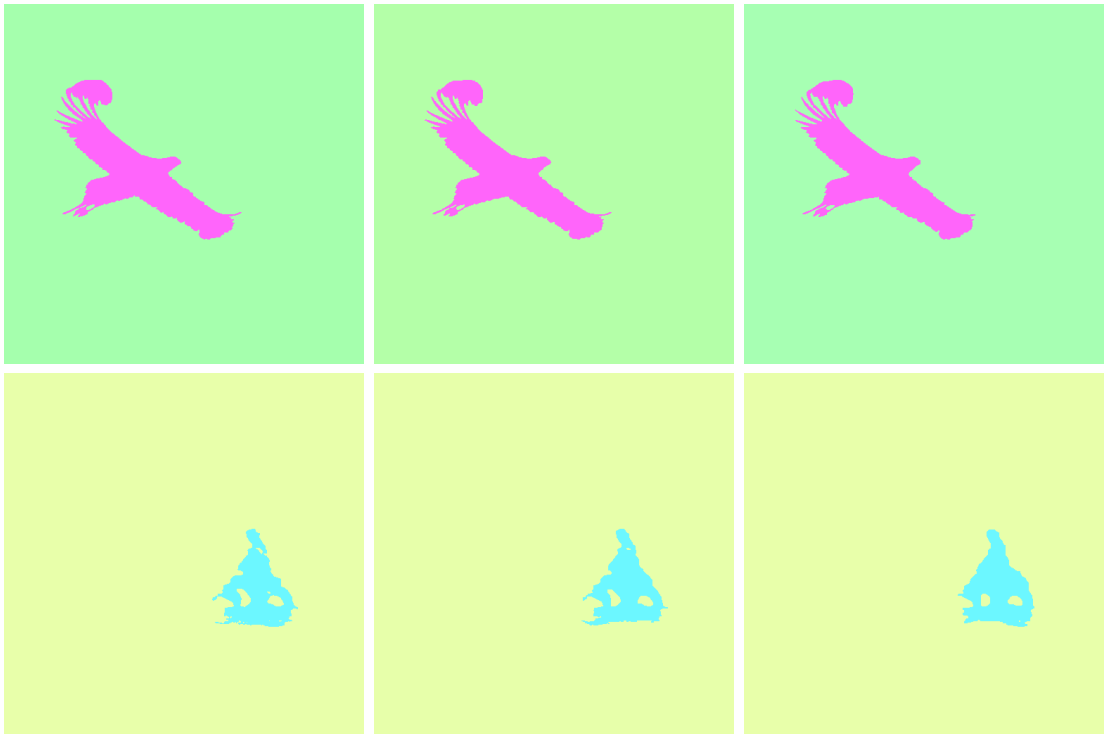


Figure 3.8: Left column: $\lambda_{sparse,G} = 0.01$. Middle column: $\lambda_{sparse,G} = 0.05$. Right column: $\lambda_{sparse,G} = 0.1$. The results are stable with respect to the choice of $\lambda_{sparse,G}$, as long as it does not become too large (right column, bottom). For all sequences, we chose $\lambda_{sparse,G} = 0.05$.

average endpoint error increases from 1.4 pixel to 1.8 pixel; the PSNR decreases from 23.7 to 22.8.

3.4.3 Quantitative results: Motion Estimation Accuracy

We compare our motion estimation accuracy with different methods from the optical flow literature: Sun et al. [233] use a layered optical flow model; Portz et al. [192] incorporate motion blur into an algorithm for optical flow computation, but do not take layers into account; Xu and Jia [282] are representative of a non-layered, but accurate optical flow method. The implementations were either obtained from the author's websites, or provided upon request. In all cases, we used the default parameters. The only exception was [192], for which we were unable to compute reasonable results using the included initialization optical flow method. Therefore, we use the same initialization [282] as for our method.

To quantitatively compare the results, we use two metrics on the synthetic sequences. First, we compare the full dense flow fields of all methods using the average endpoint error ("Error frame" in Table 3.1). Second, we fit two parametric motions

Figure 3.9: Left column: $\lambda_{sparse,A} = 0$. Middle column: $\lambda_{sparse,A} = 0.001$. Right column: $\lambda_{sparse,A} = 0.1$. Again, we observe a high robustness with respect to the choice of $\lambda_{sparse,A}$. The differences are only visible in small details, like the front wheel of the bicycle. For all sequences, we chose $\lambda_{sparse,A} = 0.001$.

to all flow fields, as described in Sec. 3.3, and compare those ("Error fitted" in Table 3.1). Figure 3.11 shows an example. As can be seen in Fig. 3.11, our method is able to extract even very fine details, however, in the absence of texture in the sky, the segmentation of foreground and background is ambiguous. Table 3.1 shows the average errors over all sequences. Flow accuracy with our method improves significantly over the other techniques.

To investigate the effect of our explicit motion blur model, we compare the motion estimation accuracy for our method with and without the motion blur modeling enabled. Disabling the motion blur is equivalent to setting the shutter speed to be infinitely short. Without modeling blur, the accuracy of our method is comparable to the other methods. The results in Table 3.1 clearly show the critical role that the blur model plays in motion estimation accuracy.

Note that compared to other optical flow methods, our method computes good segmentations, as implicitly shown in the optical flow maps. More explicitly, consider Fig. 3.1(e). Here, the object boundaries extracted from the standard optical flow (as described in Sec. 3.3) are shown in red, while the layer boundaries extracted

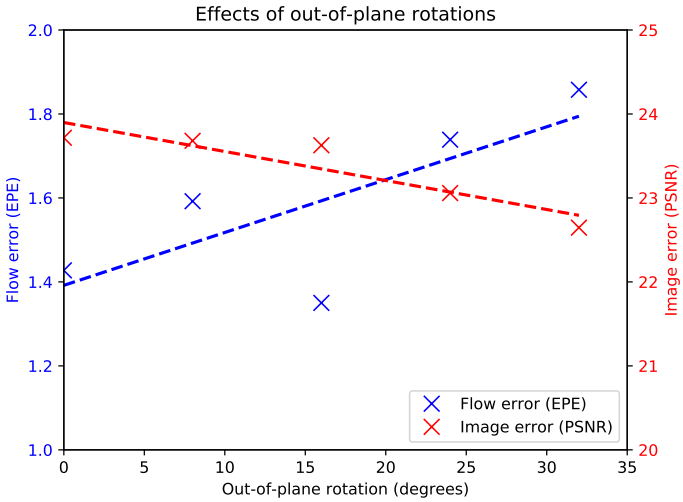


Figure 3.10: Effects of increasing out-of-plane rotation. While the quality is impacted, the overall degradation is farily low.

(a) Input frames (b) Xu and Jia [282] (c) Portz *et al.* [192] (d) Sun *et al.* [233] (e) Ours

Figure 3.11: Conventional optical flow estimation methods fail at object boundaries in the presence of motion blur, since the motion in those areas is a combination of two motion-blurred image regions. (e) shows our result.

from our method are shown in green. While the standard flow method [282] fails to capture the fine details between the fingers due to the motion blur, our method is capable of recovering those details. A similar effect can be observed in Fig. 3.5(d), and Fig. 3.11(e).

3.4.4 Quantitative results: Deblurring Accuracy

We compare the results of our approach with those of [65] for deblurring. Figure 3.12(a) shows an input from a scene in which both layers only undergo translational motion. Within each layer, the blur kernel is therefore spatially invariant. Note that [65] is designed for spatially invariant blur caused by camera shake. To apply it to these images with multiple motions, we use the approach described in Sec. 3.3

Table 3.1: Optical flow accuracies (average endpoint error).

	Error frame	Error fitted
MDP-Flow [282]	3.58	3.28
Blurflow [192]	4.59	4.92
Layerflow [233]	3.27	4.15
Ours-noblur	3.06	3.42
Ours	**0.73**	**1.25**

(a) One input frame (b) [65], segmented (c) Our result

Figure 3.12: Deblurring example (one frame from sequence). Note that there is different foreground and background blur. Conventional deblurring methods suffer from discontinuously varying motion blur at object boundaries.

to obtain a rough estimate of background and foreground segmentations. Both segments are then separately deblurred, and the result is composed again. Figure 3.12 shows a visual comparison of deblurring results. On average, this modified version of [65] achieves a PSNR of 18.34 dB, while our generative model has a PSNR of 29.31 dB. Since [65] was not intended to be used in a spatially-discontinuous setting, this is not an entirely fair comparison. However, by the same token it is not our goal to present a perfect deblurring method. Rather, we show that our generative model of layered motions effectively deblurs the layers resulting in better deblurred images.

3.4.5 Historical Challenge

Figure 3.13 demonstrates the method for an extremely challenging archival video with large blur, film grain, and extreme noise – the famous Zapruder film of the John F. Kennedy assassination. Using 7 frames, our method removes the motion blur in the scene, both in the rightward driving car and in the background. The

(a)
One input frame and magnification

(b)
Deblurred frame, magnification, and computed motion and layer segmentation

Figure 3.13: Kennedy Assassination. Our method is robust to severely degraded input images. *Zapruder Film ©1967 (Renewed 1995) The Sixth Floor Museum At Dealey Plaza.*

(a)
One input frame

(b)
Motion estimation

(c)
Deblurring result

Figure 3.14: Failure case. A reflection in the windshield causes a changing appearance of the foreground layer, leading to an incorrect segmentation.

non-rigid motions of the people in the scene (e.g. the child's legs) produce some artifacts and require a more flexible motion model. However, even without this our method performs surprisingly well.

3.5 Limitations of the proposed approach

To demonstrate our approach, we assume parametric models of the layer motion, which currently restricts our method to scenes with suitable motion. Our model formulation however is general and just as valid for smoothly varying flow within a layer. The model also assumes a two-layer scene. While such a model is frequently used in comparable work [189, 233], it is again somewhat restrictive.

Second, our model assumes constant layer appearance over the sequence. Figure 3.14 shows a case in which reflections in the windshield of the car cause a change in appearance over time. This leads to ringing artifacts in the segmentation and foreground layer estimation.

The limitations mentioned above should be considered in the light of our claims. We do not claim that we have developed either a full motion estimation or a full deblurring system. Instead we propose a new layered generative model of blurred video and show its feasibility for motion estimation and deblurring. Recent work on layered flow has addressed the estimation of the number of layers, their depth order, and the use of static image cues to improve layer segmentation [231]. Our framework is consistent with these techniques and could be incorporated into them.

3.6 Conclusion

In this chapter we have developed a principled formulation of motion blur in layers, have shown how it can be used to jointly estimate parametric motion, deblurred appearance, and scene segmentation, and have demonstrated the effectiveness of this approach using synthetic and real video sequences with appropriate motion. The results point to the value of incorporating a model of motion blur into the formulation of optical flow. A key insight is that, given the optical flow and the shutter speed, the blur is completely determined. This simplifies the modeling and estimation problem. Additionally, we argue that the scene structure, resulting in occlusion, has to be modeled in order to capture the complex interplay between surfaces of different depths, especially if motion blur causes the appearance of such surfaces to smear into each other.

To this end, we have shown that a layered model of the scene is an appropriate representation for the scene structure. While layers only provide a coarse approximation of the full 3D geometry of the scene, they are well suited to reason about the effects at occlusions between different overlapping surfaces in the image. These regions of overlap are the main challenge when modeling motion blur in dynamic scenes, and using layers allow us to properly treat the blur at boundaries between surfaces. By modeling the blur process in a layered framework, we therefore achieve better results for the tasks of motion estimation, layer segmentation, and layer deblurring.

Chapter 4

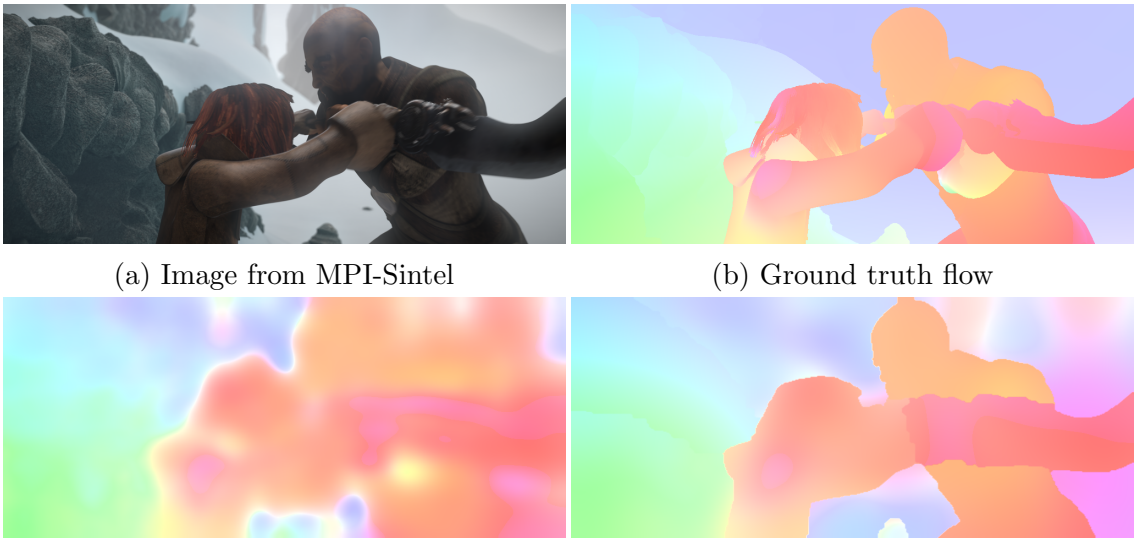
Learning the structure of layered optical flow

4.1 Introduction

In the previous chapter, we saw that layers are a useful approximation of the structure of the scene[1]. They especially improve results near object boundaries, since they provide a convenient way of modeling the effects of occlusions. Traditional layered methods, however, are slow. The method described in the previous chapter takes several minutes per frame; methods that estimate dense (*i.e.* non-parametric) optical flow for each layer often have runtimes of tens of minutes [233] to tens of hours [231].

The issue of runtime is closely related to the way in which flow is represented. The representation of an optical flow field can be thought of as lying on a continuum. On one side are purely parametric approaches, *i.e.* a global translation or global planar motion. These representations are fast and robust to compute, since they only have very few degrees of freedom; in particular, the number of degrees of freedom is much lower than the number of measurements. However, parametric representations are often not expressive enough and cannot model complex geometries or deformations, for example the motion of a walking person. On the other side of the continuum lies fully dense optical flow, represented via a displacement vector for each pixel. This representation can express all motions up to the sampling limit of the pixels. However, due to the higher number of free variables it is slow to compute, requires spatial regularization, and is susceptible to noise. Traditional layered methods use the second representation, and model the flow on each layer as a full optical flow field. The flow of each layer is computed using roughly the same approach as for non-layered optical flow, *i.e.* using a per-pixel data term and a local spatial regularizer (see [229] for a summary). This classic formulation is inherently inefficient, since the spatial regularization propagates information only gradually.

[1] This chapter is based on [276].



(a) Image from MPI-Sintel (b) Ground truth flow

(c) Result of PCA-Flow (d) Result of PCA-Layers

Figure 4.1: Result overview.

How, then, are we to parameterize the optical flow on each layer, given that we want it to be both expressive and not limited to certain geometrical configurations, but also fast to compute? Figure 1.4 points to a potential solution to this dilemma. It shows that *given the layer segmentation*, the flow on each layer usually varies smoothly. Within a layered framework, we can hence use an optical flow method that only captures the low spatial frequencies of the flow field, since the high spatial frequencies (*i.e.* motion discontinuities) are captured by the layer segmentation. This chapter proposes such a method, and shows how it can be integrated into a layered framework.

The method itself, called *PCA-Flow*, is based on a low-dimensional basis for optical flow fields, trained via Principal Component Analysis (PCA). To estimate a flow field, our method first matches sparse features across two frames. Sparse features are efficient to compute robustly and can capture long-range motions. Our method then robustly estimates the optical flow field in the subspace spanned by the low-dimensional basis that best explains the motion of the matched features. An alternative perspective on this process is that of interpolation: To compute dense flow efficiently from sparse matches, an algorithm must interpolate between the matches. However, due to outliers in the sparse matches and uneven covering of the images, generic interpolators do not work well. Our learned basis, on the other hand, robustly interpolates between the matches, and takes the learned structure of optical flow fields into account.

The idea of learning linear models of flow is not new [43, 83], but previous work applied such models only in image patches, not to full images. To train our

PCA model we use optical flow computed from 8 hours of video frames from four commercial movies using an existing flow algorithm (GPUflow [271]). To deal with noise in the training flow we use a robust PCA method that scales well to our huge training set [107].

Our method computes dense flow by estimating the location in the PCA subspace that best explains the sparse matches (Fig. 4.1(c)). At first it is not immediately obvious that one can represent generic flow fields using a low-dimensional PCA basis constructed from computed flow. We show that this works if we impose a prior on the flow basis based on types of scenes and motions. This demonstrates that the PCA flow basis is a powerful regularizer that captures long-range structure present in real flow fields.

This approach is very efficient. Our novel flow algorithm, called *PCA-Flow*, has a runtime of about 190 ms per frame on a standard CPU; this is the fastest CPU-based method on both KITTI [90] and MPI-Sintel [54]. While there is a trade-off between accuracy and speed, and PCA-Flow cannot compete with the most accurate methods, it is significantly more accurate than the next fastest method on KITTI and is more accurate than recent, widely used methods such as LDOF [52] and Classic+NL [229] on MPI-Sintel. Interestingly, by learning from enough data, *we obtain a lower error than the algorithm used to generate the training data for our PCA basis.*

Using this PCA-Basis, approximate optical flow is very compact and fast to compute. While sufficiently accurate for many tasks, PCA-Flow does not contain high-frequency spatial information and consequently the computed optical flow fields appear blurry, and the errors near motion boundaries are high. To obtain sharp motion boundaries while retaining efficiency, we propose to integrate PCA-Flow into a layered model of the scene, where each layer is a PCA-Flow field estimated from a subset of the sparse matches. While previous layered models are computationally expensive [231, 233], by working with sparse matches and the learned PCA interpolator, the motion of each layer can be efficiently computed using Expectation Maximization (EM) [127].

We therefore extend the previous approach by simultaneously computing *multiple* approximate optical flow fields, each based on a subset of the tracked features. For each subset, this yields an oversmooth flow field, spanning the whole image. To compute a final dense flow field, we must combine the flow fields estimated for each layer. We do so using a Markov Random Field (MRF) that incorporates image evidence to select among PCA-Flow fields at each pixel. This *PCA-Layers* method computes optical flow fields with much sharper motion boundaries and reduces the overall errors (Fig. 4.1(d)). At the same time, it is still reasonably fast, taking on average 3.2s/frame on MPI-Sintel. On Sintel it is more accurate than methods like MDP-Flow2 [282], EPPM [24], MLDP-OF [176], Classic+NLP [229] and the traditional layered approach, FC-2Layers-FF [233], which is a least two orders of magnitude slower. Most interestingly, PCA-Layers is particularly good in occluded

(unmatched) regions, achieving lower errors there than DeepFlow [267] on Sintel.

Our motivation for developing a fast, and reasonably accurate, flow algorithm is severalfold. First, YouTube states that users upload 100 hours of video every minute[2]. If one wants to understand what is going on *inside* this video, one needs to compute motion. This is not practical today with algorithms that are several orders of magnitude slower than real time. Second, the experience in the vision community is that many problems work better with large datasets [104]. This is true for all kinds of image-based retrieval problems, and recent large datasets confirm this also for the application of optical flow [76, 126, 168]. However, due to the difficulty of measuring optical flow in real scenes, most of the available large datasets for optical flow are based on synthetic scenes of varying, but usually low complexity. We believe that fast optical flow algorithms can play a crucial role in generating more realistic datasets, and hence will open many new research directions for *using* optical flow.

For research purposes, the code for both methods and the learned optical flow basis are available at [2].

4.1.1 Previous work

Chapter 2 provided a review of dense optical flow methods, but did not focus on methods combining sparse feature tracking with dense optical flow estimation. Since our method follows the general paradigm of establishing sparse feature matching and a following densification, here we give an overview of similar methods, in addition to flow methods concentrating on computational efficiency.

The idea of using tracked features to estimate motion has its roots in early signal processing [26]. Early optical flow methods used correlation matching to deal with large motions [12]. Since most matching methods are only pixel accurate, the field moved away from matching methods to focus on differential frameworks [116].

Gibson and Spann [93] describe a two-stage method that first estimates sparse feature tracks, followed by an interpolation stage. Their tracking stage uses an MRF to enforce spatio-temporal smoothing, while the interpolation phase essentially optimizes a traditional dense optical flow objective. This makes the method computationally expensive. Nicolescu and Medioni [181] use feature matching to get candidate motions and use tensor voting for interpolation. They then segment the flow into regions using only the flow information. Our PCA-Flow method has similar stages but uses a learned PCA basis for densification.

One advantage of methods that consider small neighborhoods is that they can often be computed relatively fast. Zach *et al.* [290] were the first to demonstrate optical flow computation on a GPU. Using a traditional objective function they show how a total variation approach can be parallelized with a shader. They achieve re-

[2] `http://www.youtube.com/yt/press/statistics.html`, March 7, 2013.

altime performance for small resolutions (320×240 pixels). Werlberger *et al.* [271] extend this approach to robust regularization. On a recent GPU, their algorithm takes approximately 2 seconds per frame at a resolution of 1024×436 pixels. Rannacher [194] presents an extremely fast method, but requires stereo images and a pre-computed disparity field (for example, using an FPGA). Sundaram *et al.* [235] port the large-displacement optical flow method [52] to a GPU, with a runtime of 1.8 seconds for image pairs of 640×480 pixels. The reported runtimes depend on image resolution and the type of GPU. In our algorithm, only the runtime of the feature matching and the layer assignment depend on the image resolution. In both cases, low-resolution versions of the input image can be used. This affects the number of features and resolution of the boundaries, but not the full resolution of the output optical flow.

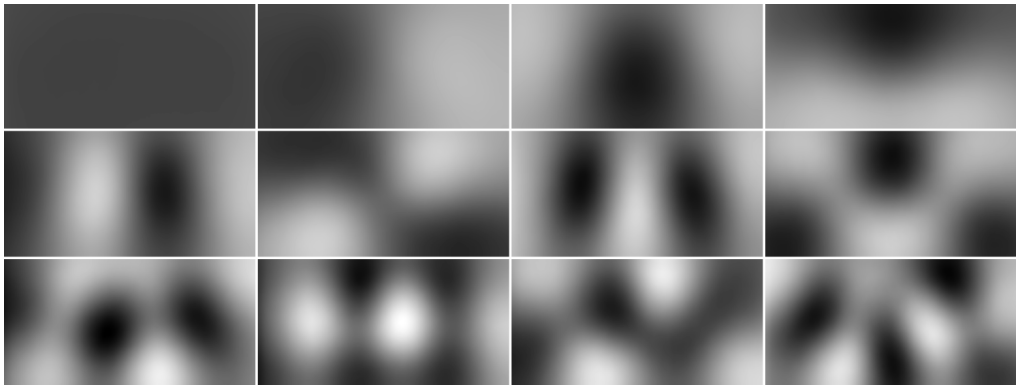
Tao *et al.* [240] propose an algorithm that scales sub-linearly with image input resolution by computing the flow on a selected subset of the image pixels and interpolating the remaining pixels. However, it still has a running time of around 2 seconds on Middlebury image pairs. Bao *et al.* [24] use a GPU to make their recent EPPM method run at about 0.25s/frame on Sintel. Recent optical flow methods based on convolutional neural networks provide superior performance and runtimes in the order of a few hundred milliseconds per frame [76, 120], but require powerful GPUs and sophisticated training procedures, and often do not generalize well across datasets. Our basic method is less accurate but equally fast, and does not rely on a GPU.

4.2 A learned basis for optical flow

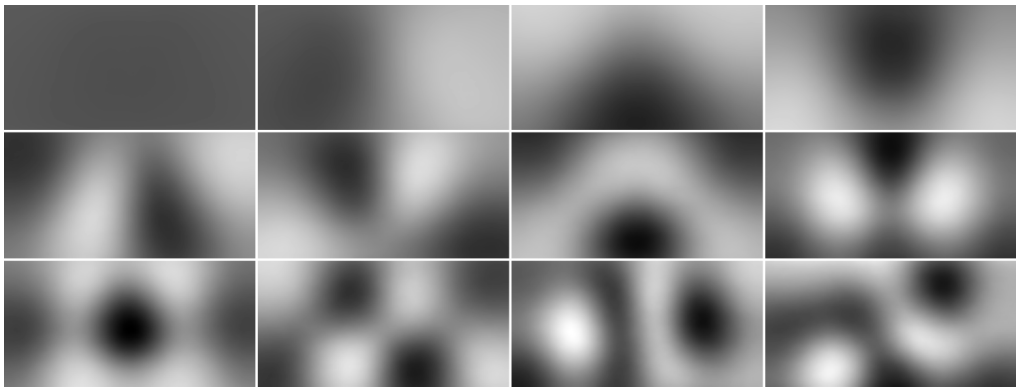
Our basic assumption is that optical flow fields can be approximated as a weighted sum over a relatively small number of basis flow fields $\mathbf{b}_i, i = 1 \dots B$, with corresponding weights θ_i

$$\mathbf{u} \approx \sum_{i=1}^{B} \theta_i \mathbf{b}_i. \tag{4.1}$$

As before, $\mathbf{u}$ and $\mathbf{b}_i$ are 2D optical flow fields. We assume separate basis vectors for the horizontal and vertical flow components, so that the horizontal motion is spanned by $\{\mathbf{b}_i\}_{i=1,\dots,\frac{B}{2}}$, and the vertical by $\{\mathbf{b}_i\}_{i=\frac{B}{2}+1,\dots,B}$. As before, the weights $\boldsymbol{\theta}$ can be interpreted as the parameters of a parametric model. The difference is that this time we use a number of parameters that is much larger than traditional parametric motion models.



(a) Principal components for horizontal motion

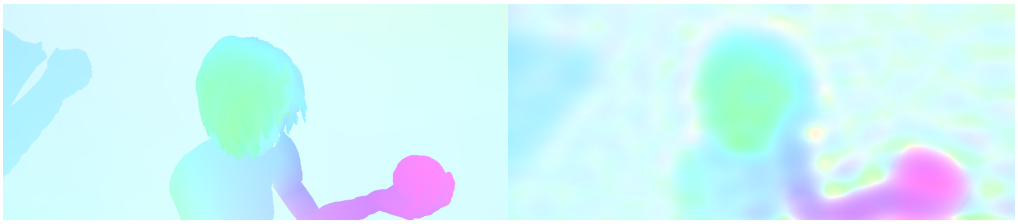


(b) Principal components for vertical motion

Figure 4.2: First 12 components for horizontal and vertical motion. Contrast enhanced for visualization.

4.2.1 Learning the flow basis

To learn the basis flow fields, we use data from four Hollywood movies spanning several genres (*Star Wars*, *Babel*, *The Constant Gardener*, *Land of Plenty*). For each movie, we compute the optical flow using GPUFlow [271]. This method is not the most accurate (as we will see, it is less accurate than our PCA-Flow algorithm, which we train using the output of GPUFlow). However, at the time of writing, it was the fastest method with an available reference implementation, and has a runtime of approximately 2 seconds per frame. Computing the optical flow takes approximately 4 days per movie. The computed flow is then resized to 512×256 pixels and the magnitudes of the flow values are scaled accordingly; this is the same resolution used in [267]. Since we only use a fraction of the computed principal components, high frequency components are absent from the reconstructed flow fields; see Fig. 4.3. It is therefore possible to use this smaller spatial resolution without sacrificing detail.



(a) Ground truth optical flow (b) Projected optical flow

Figure 4.3: Example of projecting Sintel ground truth flow onto the first 500 principal components.

From the computed optical flow fields, we randomly select 180,000 frames, limited by the maximum amount of memory at our disposal. We first subtract the mean flow, which contains some consistent boundary artifacts caused by the GPU-Flow method. Note that here, the dimensionality of our components is higher than the number of datapoints. However, compared to the theoretical dimensionality, we extract only a very small fraction of the principal components, here $B = 500$, 250 for the horizontal motion and 250 for the vertical. Since the computed optical flow contains outliers due to editing cuts and frames for which the optical flow computation fails, we use a robust PCA method to compute the principal components [107]. The total time required to extract 500 components is approximately 22 hours. However, this has to be done only once and offline; we make the learned basis available [2]. Figure 4.2 shows the first 12 flow components in the horizontal and vertical directions. Note that one could also train a combined basis for vertical and horizontal motion. In our experiments, however, separate bases consistently outperformed a combined basis. Note also that the first six components do not directly correspond to affine motion, in contrast with what was found for small flow patches [83].

Figure 4.2 reveals that the resulting principal components resemble the basis functions of a Discrete Cosine Transform (DCT). In order to achieve comparable performance to our learned basis with the same number of components, we generated a DCT basis with ten times more components and used basis pursuit to select the most useful ones. However, as will be shown in Sec. 4.5.1, the DCT basis gave slightly worse endpoint errors in our experiments and so we do not consider it further.

Figure 4.3 shows the projection of a ground truth flow field from Sintel onto the learned basis. Note that no Sintel training data was used in learning the basis, so this tests generalization. Also note that the Sintel sequences are quite complex and that the projected flow is much smoother than the ground truth flow. We discard the higher principal components and only use the lower 500; since the higher components correspond to higher spatial frequencies, the blurry appearance

of the flow field in Fig. 4.3 is to be expected. For the impact of the number of principal components on the reconstruction accuracy, as well as for a quantitative comparison with a DCT basis, please see Sec. 4.5.1.

To compute the flow bases, one obviously has alternatives, such as methods enforcing sparsity in either the components [295] or the coefficients [164]. So far, we did not employ those for two reasons. First, compared to PCA, which is efficient both in terms of computation time and memory requirements, even simple sparse methods require a more complex optimization machinery. Second, a preliminary test on a smaller dataset and smaller image sizes did not result in improvements compared to the PCA basis.

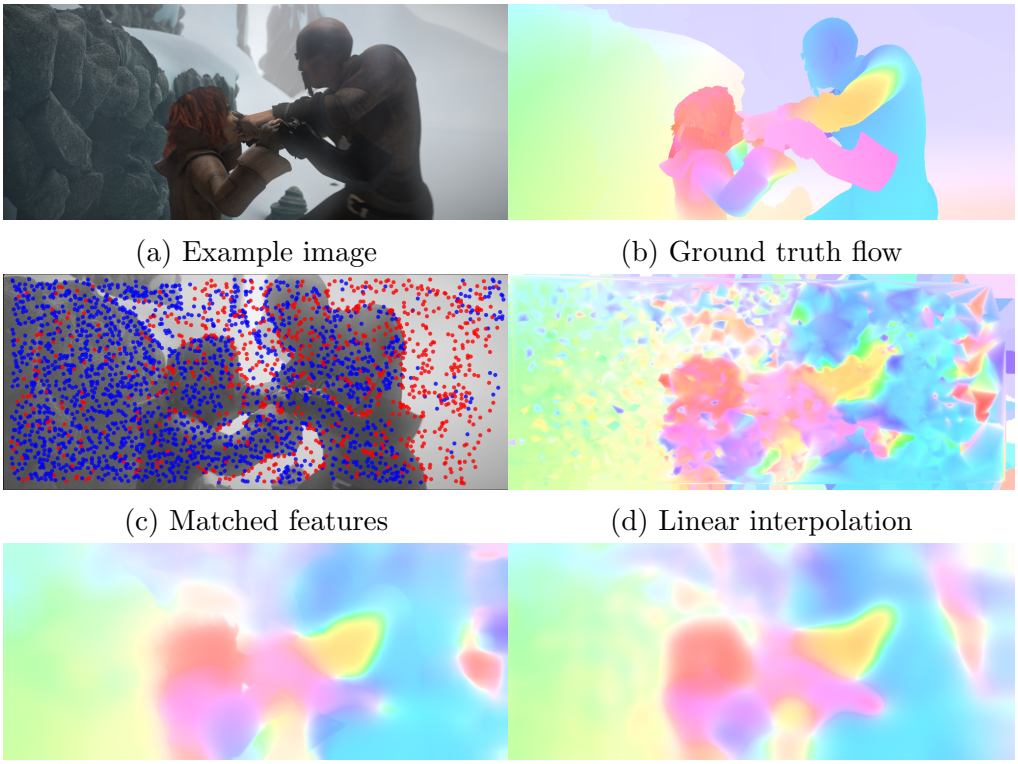
4.3 Estimating flow

Given an image sequence and the learned flow basis, we estimate the coefficients that define the optical flow. To that end, we first compute sparse feature matches to establish correspondences of key points between both frames. We then estimate the coefficients that produce a dense flow field that is consistent with both the matched scene motion and with the general structure of optical flow fields.

4.3.1 Sparse feature matching

Our algorithm starts by estimating M sparse feature matches across neighboring frames; *i.e.* pairs of points $\{(\mathbf{q}_m, \mathbf{q'}_m)\}, m = 1 \dots M$. $\mathbf{q}_m$ is the 2D location of a (usually visually distinct) feature point in frame 1, and $\mathbf{q'}_m$ is the corresponding feature point location in frame 2. Each of these correspondences induces a displacement vector $\mathbf{v}_m = \mathbf{q'}_m - \mathbf{q}_m = (v_{m,1}, v_{m,2})^\top$. Using sparse features has two main advantages. First, it provides a coarse estimate of image and object motions while being relatively cheap computationally. Second, it establishes long range correspondences, which are difficult for traditional, dense flow methods to estimate [52, 267, 282].

As preprocessing, we normalize the image contrast using CLAHE [296] to increase detail that can be captured by the feature detectors. Then, we use the features from [92], which are designed for visual odometry applications. We found that to match features across *video* frames, these features work much better than image matching features such as SURF or SIFT. The latter are invariant against a wide range of geometric deformations which rarely occur in adjacent frames of a video, and hence return a large number of mismatches. Furthermore, the features we use are computationally much cheaper: currently, matching across two frames in MPI-Sintel takes on average 80 ms. Using sparse features creates the problem of low coverage in unstructured image regions. However, this problem also exists in classical optical flow: If image structure is missing, the data term becomes



(a) Example image (b) Ground truth flow

(c) Matched features (d) Linear interpolation

(e) Guided interpolation (f) PCA-Flow (ours)

Figure 4.4: Sparse features and possible interpolations.

ambiguous, and flow computation relies on the regularization, just as our approach relies on the learned basis for interpolation.

Feature matches will always include outliers. We account for these in the matching process by formulating our optimization in a robust manner below. Figure 4.4a shows a frame and Fig. 4.4c the corresponding features. Features shown in blue have an error of less than 3 pixels; features with greater errors are red.

4.3.2 Estimating the dense flow field

Estimating a dense optical flow field from the sparse feature matches can be seen as a regression problem. Using our learned flow basis vectors $\mathbf{b}_i$, this can be formulated as finding the weighted linear combination of flow basis vectors that best explains the detected feature matches. The weights then define the dense flow field.

First, consider a basic version of the method. This can be expressed as a simple least squares problem in the unknown $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_B)^\top$:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\mathbf{A}\boldsymbol{\theta} - \bar{\mathbf{v}}\|^2 \tag{4.2}$$

with

$$\mathbf{A} = \begin{pmatrix} (\mathbf{b}_1(\mathbf{q}_1))_1 & & (\mathbf{b}_B(\mathbf{q}_1))_1 \\ \vdots & & \vdots \\ (\mathbf{b}_1(\mathbf{q}_M))_1 & \cdots & (\mathbf{b}_B(\mathbf{q}_M))_1 \\ (\mathbf{b}_1(\mathbf{q}_1))_2 & & (\mathbf{b}_B(\mathbf{q}_1))_2 \\ \vdots & & \vdots \\ (\mathbf{b}_1(\mathbf{q}_M))_2 & & (\mathbf{b}_B(\mathbf{q}_M))_2 \end{pmatrix} \tag{4.3}$$

and

$$\bar{\mathbf{v}} = \begin{pmatrix} (v_1)_1 \\ \vdots \\ (v_M)_1 \\ (v_1)_2 \\ \vdots \\ (v_M)_2 \end{pmatrix} \tag{4.4}$$

Here, $(\mathbf{b}_i(\mathbf{q}_m))_1$ is the horizontal motion at location $\mathbf{q}_m$ according to basis flow field $\mathbf{b}_i$, and $(\mathbf{b}_i(\mathbf{q}_m))_2$ is the vertical motion. We use nearest neighbor interpolation to compute the elements at fractional coordinates; bilinear and bicubic interpolation did not increase the accuracy in our experiments. Note that the vertical components of $\{\mathbf{b}_i\}, i = 1 \ldots \frac{B}{2}$ are zero, as are the horizontal components of $\{\mathbf{b}_i\}, i = \frac{B}{2} + 1 \ldots B$. $\bar{\mathbf{v}}$ contains the motion of the matched points.

Solving Eq. (4.2) yields $\hat{\boldsymbol{\theta}}$, and thus the estimated dense optical flow field

$$\hat{\mathbf{u}} = \sum_{i=1}^{B} \hat{\theta}_i \mathbf{b}_i. \tag{4.5}$$

Unfortunately, the sparse feature matches usually contain outliers. Since the search for feature matches is done across the whole image (*i.e.*, the spatial extent of feature motion is not limited), the errors caused by bad matches are often large, and thus can have a large influence on the solution of Eq. (4.2). Therefore, we solve a version of Eq. (4.2) in which each residual is penalized robustly,

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{2M} \rho\left(\| \left(\mathbf{A}\boldsymbol{\theta} - \bar{\mathbf{v}}\right)_i \|\right) \tag{4.6}$$

where $\rho(z)$ is the robust Lorentzian penalty

$$\rho(z) = \frac{\sigma^2}{2} \log\left(1 + \left(\frac{z}{\sigma}\right)^2\right). \tag{4.7}$$

The parameter σ controls the sensitivity to outliers. Note that (4.7) is just one of

many possible robust estimators [40]. We found the Lorentzian estimator to work well.

To further increase robustness, we adapt the number of estimated weights (*i.e.*, the number of bases used) to the number of found features; the more features are detected, the more bases we use. The reasoning in this is that if we only small number of features are found, it is better to accurately estimate the coarse scene motion, *i.e.* the weights of the first principal components, and avoid overfitting. Using the MPI-Sintel training set, we found that limiting the number of estimated weights to one third the number of matched features works well.

If the input images have a different resolution than the flow basis, we first detect the features at full resolution, and scale their locations to the resolution of our flow basis. The weights are then estimated at this resolution, and the resulting optical flow field is upsampled and scaled again to the resolution of the input images.

Note that one could also estimate the coefficients using classical, warping-based dense estimation of parametric optical flow [160]. This is the approach used in [83]. We implemented this method and found, surprisingly, that its accuracy was comparable to our PCA-flow method with sparse feature matches. Because it is much slower, we do not consider it further here; see Sec. 4.5.1 for results and comparisons.

4.3.3 Imposing a prior

Equation (4.6) does not take any structure of the distribution of $\boldsymbol{\theta}$ in training into account. The simplest prior on $\boldsymbol{\theta}$ is given by the eigenvalues computed during PCA on the training flow fields. New sequences may have quite different statistics, however. In KITTI, for example, the motion is caused purely by the egomotion of a car, and thus is less general than our training data. While KITTI and MPI-Sintel contain training data, the amount of data is insufficient to learn the full flow basis. We can, however, keep the basis fixed and adapt the prior. Since the prior lies in the 500-dimensional subspace defined by our flow basis, adjusting the prior requires much less training data. Given ground truth flow fields (*e.g.* from KITTI or Sintel), we project these onto our generic flow basis and compute $\boldsymbol{\Gamma}$, the inverse covariance matrix of the coefficients. We then express our prior using a Tikhonov regularizer on $\boldsymbol{\theta}$:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{2M} \rho\left(\| \left(\mathbf{A}\boldsymbol{\theta} - \bar{\mathbf{v}}\right)_i \|\right) + \lambda \|\boldsymbol{\Gamma}\boldsymbol{\theta}\|. \tag{4.8}$$

Intuitively, if a certain coefficient does not vary much in the projection of the training set onto the flow bases, we restrict this coefficient to small values during inference. When training data is available, this regularizer improves performance significantly.

We solve Eq. (4.8) using Iterative Reweighted Least Squares and refer to the

method as **PCA-Flow**. Figure 4.4 shows the results of our method (4.4f) in comparison to two simpler methods that interpolate from sparse features to a dense flow field, nearest-neighbor interpolation (4.4d) and image-guided interpolation [109] (4.4e). These generic interpolation methods are oblivious to the structure of optical flow and cannot detect and eliminate outliers caused by wrong feature matches. Thus, their average endpoint errors on the Sintel test set (*linear*: 9.07 px; *guided*: 8.44 px) are higher than our basic method, PCA-Flow (7.74 px).

4.4 PCA-Flow within a layered framework

While the smooth flow fields generated by PCA-Flow may be appropriate for some applications, many applications require accurate motion boundaries. As mentioned above, however, *within* a layer the true flow is usually smooth; strong motion discontinuities are captured by the transitions between layers. This indicates that it should be possible to compute a good overall flow field by computing several smooth flow fields, and combining them using a layered scene model.

4.4.1 Layers from sparse matches

Here, we assume that a full optical flow field is composed of L simpler motions[3], where one of the motions is assigned to each pixel. The flow in each layer is represented by our learned basis as above with one modification: Since the motion of each layer should be simpler than for the full flow field, we change the prior by computing it on a layered flow representation. To obtain this representation for training, we first cluster the motion fields in our training set into layers with similar motions. Then, we compute $\boldsymbol{\theta}$ for each of the layers, compute the covariance matrix $\boldsymbol{\Sigma}$ from the weights of all segments across the whole training set, and use $\boldsymbol{\Gamma} = \boldsymbol{\Sigma}^{-1}$ in Eq. (4.8).

To compute the simpler motions at test time, we first cluster sparse feature matches using an EM algorithm with hard assignments[4]. To initialize, we cluster the features into L clusters using K-Means. The assignments of features to layers at iteration i are represented as assignment variables $c_m^{(i)}, m = 1 \dots M$, where $c_m^{(i)} = l$ means that feature point $\mathbf{q}_m$ is assigned to layer l. Given a set of layers, the distance of a feature point $\mathbf{q}_m$ to a layer l in iteration i is given as

$$\begin{aligned} \texttt{dist}^{(i)}\left(\mathbf{q}_m, l\right) = &\|\mathbf{u}_l^{(i-1)}\left(\mathbf{q}_m\right) - \mathbf{v}_m\| + \\ &\alpha\|\mathbf{q}_m - \texttt{median}\left(\mathbf{q}_m | c_m^{(i-1)} = l\right)\| \end{aligned} \tag{4.9}$$

[3] In our experiments, we found $L = 6$ to work well, with $B = 100$ principal components per layer

[4] Soft assignments did not significantly change the results, and increased the runtime.

where

$$\mathbf{u}_l^{(i-1)}(\mathbf{q}_m) = \sum_{j=1}^{B} \left(\theta_l^{(i-1)}\right)_j \mathbf{b}_j(\mathbf{q}_m) \tag{4.10}$$

is the optical flow field of layer l, evaluated at $\mathbf{q}_m$, and $\mathbf{v}_m$ are the feature displacements as defined above. The right part of Eq. (4.9) is the distance of point $\mathbf{q}_m$ to the median of all features assigned to l in the previous iteration; initially, the medians are initialized to the center of the image. α is a weighting factor.

The features are then hard-assigned to the layers

$$c_m^{(i)} = \arg\min_l \texttt{dist}^{(i)}(\mathbf{q}_m, l) \tag{4.11}$$

and the layer motions are updated to

$$\boldsymbol{\theta}_l^{(i)} = \texttt{estimate}\left(\left\{\mathbf{q}_m | c_m^{(i)} = l\right\}\right) \tag{4.12}$$

where $\texttt{estimate}(\cdot)$ computes PCA-Flow (Eq. (4.8)) using a given subset of features. Since the motion of each layer is simpler than the motion of the whole image, the layers do not have to capture fine spatial detail. Consequently we reduce the number of linear coefficients from 500 to 100; this is sufficient for good results. We iteratively solve Eqs. (4.9)–(4.12) for 20 iterations, or until the assignments c_m do not change anymore.

4.4.2 Combining the layers

The estimated layers give motion for their assigned features but we have created a new interpolation problem. We do not know which of the non-feature pixels correspond to which layer. Consequently we develop a method to combine the layers into a dense flow field. Several methods have been propose in the literature for related problems [151, 161]. Here we use a simple MRF model.

The layer estimation step generates L approximate optical flow fields, represented by their coefficients $\boldsymbol{\theta}_l$, and the final assignment variables c_m, denoting which sparse feature belongs to which motion model. We treat each layer's motion as a proposal. In addition to these L flow fields, we compute two additional flow proposals: a simple homography model, robustly fit to all matched features, and the full approximate flow field, *i.e.* solving Eq. (4.8) with all features and 500 principal components ("global model"). Therefore, $\tilde{L} = L + 2$.

At each image location $\mathbf{x}$, the task is now to assign a label $c(\mathbf{x}) \in 1 \ldots \tilde{L}$, with the best flow model at this pixel. Then, the final optical flow field $\mathbf{u}_{final}(\mathbf{x})$ is given as

$$\mathbf{u}_{final}(\mathbf{x}) = \sum_{l=1}^{\tilde{L}} \delta\left[l = c(\mathbf{x})\right] \mathbf{u}_l(\mathbf{x}). \tag{4.13}$$

Finding $c(\mathbf{x})$ can be formulated as an energy minimization problem, which can be solved via multi-class graph cuts [46]:

$$\hat{c} = \arg\min_{c} \sum_{\mathbf{x}} E_u(\mathbf{x}, c(\mathbf{x})) + \gamma \sum_{\mathbf{y}\in\mathcal{N}(\mathbf{x})} E_p(\mathbf{x}, \mathbf{y}, c(\mathbf{x}), c(\mathbf{y})) \tag{4.14}$$

where E_u and E_p are unary and pairwise energies, respectively, and $\mathcal{N}(\mathbf{x})$ denotes the 4-neighborhood of $\mathbf{x}$. Omitting the arguments $\mathbf{x}, c(\mathbf{x})$, the unaries E_u are defined as

$$E_u = E_{warp} + \gamma_c E_{col} + \gamma_l E_{loc}. \tag{4.15}$$

Warping cost. The warping cost E_{warp} is a rectified brightness and gradient constancy term:

$$E_{warp}(\mathbf{x}, c(\mathbf{x})) = 1 - \exp\left(-\left(\frac{\texttt{cost}(\mathbf{x}, c(\mathbf{x}))}{\sigma_w}\right)^2\right) \tag{4.16}$$

$$\texttt{cost}(\mathbf{x}, c(\mathbf{x})) = \left\| \begin{pmatrix} I_1(\mathbf{x}) - I_2\left(\mathbf{x} + \mathbf{u}_{c(\mathbf{x})}(\mathbf{x})\right) \\ \nabla_{x_1} I_1(\mathbf{x}) - \nabla_{x_1} I_2\left(\mathbf{x} + \mathbf{u}_{c(\mathbf{x})}(\mathbf{x})\right) \\ \nabla_{x_2} I_1(\mathbf{x}) - \nabla_{x_2} I_2\left(\mathbf{x} + \mathbf{u}_{c(\mathbf{x})}(\mathbf{x})\right) \end{pmatrix} \right\|_2 . \tag{4.17}$$

We experimented with changing the weight of the gradients in Eq. (4.17), but did not find it to improve results.

Color cost. We build an appearance model for each layer using the pixel colors at the feature points assigned to this layer. This helps especially in occluded regions, where the warping error is high, but the color provides a good cue about which layer to use.

$$E_{col}(\mathbf{x}, c(\mathbf{x})) = -\log p_{c(\mathbf{x})}(I_1(\mathbf{x})) \tag{4.18}$$

$$p_{c(\mathbf{x})}(I_1(\mathbf{x})) = N\left(\boldsymbol{\mu}_{c(\mathbf{x})}, \boldsymbol{\Sigma}_{c(\mathbf{x})}\right) \tag{4.19}$$

where $\boldsymbol{\mu}_{c(\mathbf{x})}, \boldsymbol{\Sigma}_{c(\mathbf{x})}$ are computed from the pixels $\{I_1(\mathbf{q}_m) | c_m = c(\mathbf{x})\}$ (*i.e.* from the feature matches assigned to layer l) and $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the multivariate Gaussian distribution. Here, we use simple multivariate Gaussian distributions as color models; we found these to perform as well as or better than multi-component Gaussian

Mixture Models. For the homography model, we fit the distribution to all inlier features; for the global model, we fit it to all features.

Feature location cost. Lastly, we note that the features assigned to a given layer are often spatially clustered, and the quality of the model decreases for regions far away from the features. Therefore, we encourage spatial compactness of the layers using

$$E_{loc}(\mathbf{x}, c(\mathbf{x})) = 1 - \sum_{\{m|c_m=c(\mathbf{x})\}} \frac{1}{\sqrt{2\pi\sigma_l^2}} \exp\left(-\frac{(\mathbf{x}-\mathbf{q}_m)^2}{\sigma_l^2}\right) \tag{4.20}$$

For the homography model, we again use only the inlier features, for the global model we use all.

Image-modulated smoothness. To enforce spatial smoothness, we use the image-modulated pairwise Potts energy from GrabCut [202]:

$$\begin{aligned} &E_p\left(\mathbf{x}, \mathbf{y}, c\left(\mathbf{x}\right), c\left(\mathbf{y}\right)\right) = \\ &-\delta\left[c\left(\mathbf{x}\right) = c\left(\mathbf{y}\right)\right] \exp\left(-\frac{\left(I_1\left(\mathbf{x}\right) - I_1\left(\mathbf{y}\right)\right)^2}{2\mathbb{E}\left[\|\nabla I_1\|_2\right]}\right) \end{aligned} \tag{4.21}$$

with $\mathbb{E}[\cdot]$ denoting the expected value. This energy encourages spatial smoothness of the layer labels between pixels, unless there is a strong gradient in the image. It thus allows the layer labels to change at image boundaries.

4.5 Experiments

This section describes the performance of our algorithm in terms of accuracy on standard optical flow datasets. Additionally, we provide runtime information, and relate this to other current optical flow algorithms. All parameters are determined using cross validation on the available training sets, and set to the numbers given in Tables 4.1 and 4.2. For PCA-Layers, we use $L = 6$ layers.

4.5.1 Algorithm behavior

Number of principal components

Fig. 4.5a shows the average endpoint error across the whole training set as a function of the number of principal components that were used. The projected ground truth is given in green, the estimation results using the learned basis are given in blue, and the estimation results using the DCT are given in red. Here we plot the results

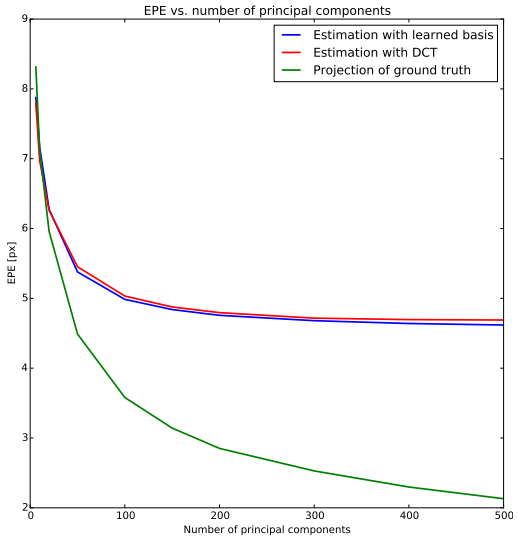
Table 4.1: Parameter values for PCA-Flow

Description	Symbol	Value `Sintel`	Value `KITTI`
Noise estimation for robust function	σ	1.0	0.6
Regularization	λ	0.2	0.4

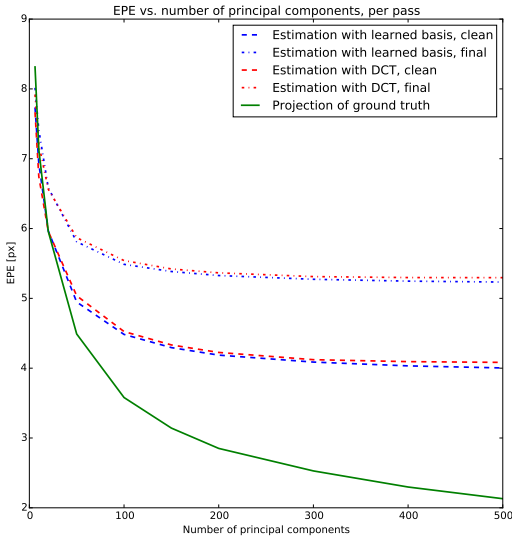
of PCA-Flow, i.e. only a single layer, since this is the part of our pipeline that is most directly affected by the maximum number of principal components.

Note that with very few principal components, the projected ground truth result is worse than the estimated results. The reason for this is that the projection minimizes the distance of the ground truth flow field to the projected ground truth field on the optical flow subspace. This does not necessarily minimize the average endpoint error.

While the results using the DCT basis are very close to the results using our learned basis, they are consistently slightly worse. Therefore, we prefer our basis.



(a) Average endpoint error as a function of the number of principal components



(b) Average endpoint error as a function of the number of principal components, split by pass

Fig. 4.5b shows the same results, split between the final pass (dotted) and the clean pass (dashed). Due to the more accurate feature matching, the results on the clean pass are much better. The shape of the curves, however, is not significantly different from Fig. 4.5a.

Table 4.2: Parameter values for PCA-Layers

		Value	
Description	Symbol	`Sintel`	`KITTI`
Noise estimation for robust function	σ	0.1	0.3
Regularization	λ	0.002	0.3
Weight of color unary cost	γ_c	3.0	3.0
Weight of location unary cost	γ_l	9.0	40.0
Weight of pairwise cost	γ	450	250
Scaling in warping energy	σ_w	3.0	0.7
Scaling in feature distance cost	σ_l	15	

Feature quality and density

One important source of error are incorrect or insufficient feature matches. Here, we show two experiment analyzing this effect, first the influence of errors in the feature matching, and second the influence of a low number of (potentially very good) feature matches.

Fig. 4.6 shows how the average endpoint error within a frame, as computed using PCA-Flow, changes with the average error of all features founds in that frame.

Fig. 4.7a and Fig. 4.7b shows the relationship between the number of features found in a single frame and the average endpoint error across this frame. Here, blue points correspond to the found feature matches. Yellow points correspond to the *ground truth matches* at the locations of the found matches. To obtain those ground truth matches, we first compute the matches, and then replace them with the ground truth flow at the detected feature locations.

The more features are found, the better the reconstruction generally is. Reversely, if only few features are found, they usually do not sufficiently cover the image, causing high errors. Additionally, the matching quality in frames with lower feature density is also lower, since those frames do not contain enough structure everywhere to reliably match features.

An additional source of errors are motion and camera blur and atmospheric effects. The final pass (shown in Fig. 4.7b) contains such artifacts, while the clean pass (Fig. 4.7a) does not. Consequently, the feature match quality is generally

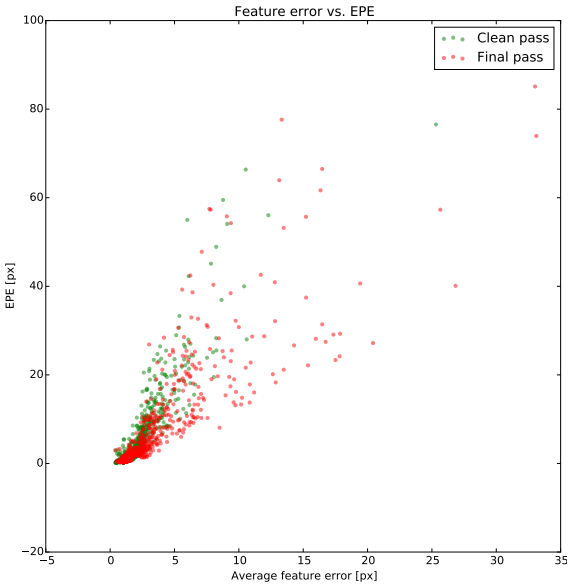


Figure 4.6: The feature error and the computed endpoint errors are very correlated. The feature errors are higher on the final pass.

Table 4.3: Errors on MPI-Sintel for estimated and ground truth features

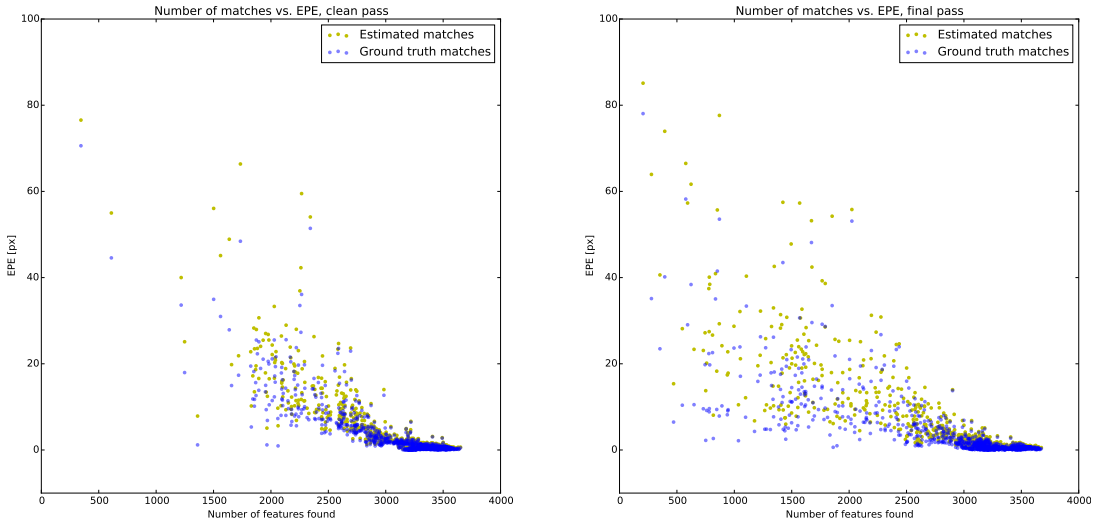
	Sintel, clean	Sintel, final	Average
Error of estimated features	1.83 px	2.67 px	2.25 px
Error over full frame, using estimated features	4.00 px	5.23 px	4.62 px
Error over full frame, using ground truth features	3.20 px	3.60 px	3.40 px

lower (in addition to more frames with only very few available features), leading to a higher error rate (See Table 4.3).

Warping-based estimation

As mentioned Sec. 4.3.2, it is also possible to use a warping-based approach to estimate the coefficients $\boldsymbol{\theta}$. Such an approach does not rely on feature matches, but instead iteratively rewarps the image to minimize the brightness constancy error [22]. It is commonly used in patch-based motion subspace methods, e.g. [34].

To be able to cope with large motions, a multiscale framework is usually used. Here, we use 7 pyramid levels, at a scale factor of 1.5 per pyramid level. Furthermore, the error term is robustified, and the same prior as in the feature-based



(a) Clean pass (b) Final pass

Figure 4.7: Number of features against average error. To work well, our algorithm requires the features to sufficiently cover the image.

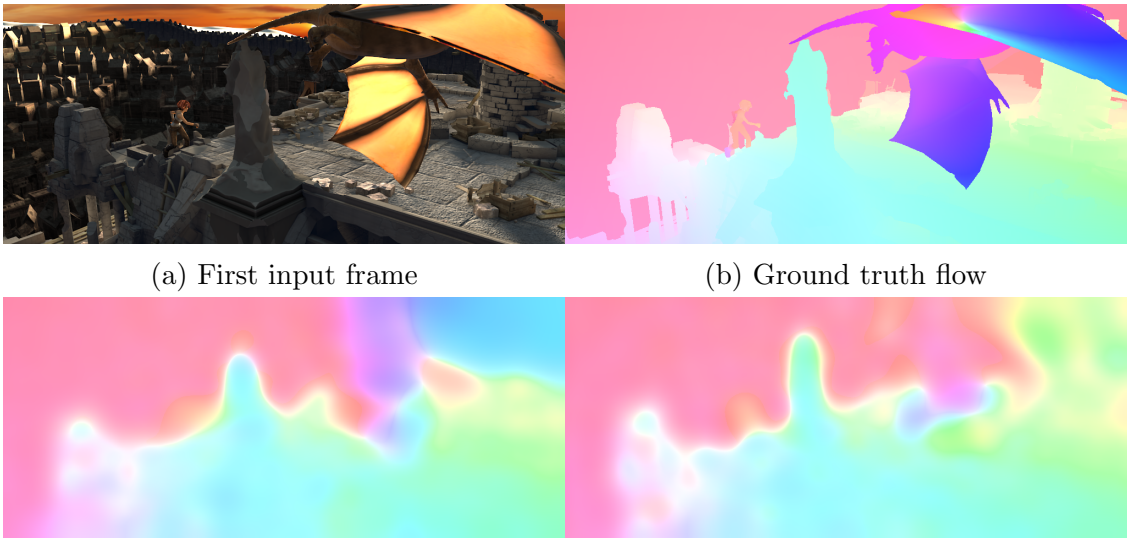
Table 4.4: Errors on MPI-Sintel for PCA-Flow and PCA-Warp.

	Sintel, clean	Sintel, final	Average
PCA-Flow	4.00 px	5.23 px	4.62 px
PCA-Warp	7.16 px	7.21 px	7.19 px

approach is taken into account. We refer to this approach as *PCA-Warp*.

Table 4.4 shows the results for the feature-based PCA-Flow and the warping-based PCA-Warp. PCA-Warp results in significantly higher errors, mostly due to large motions. At the same time, it is much slower, and takes approximately 30 seconds per frame, as compared to 190 ms for PCA-Flow. An interesting observation is that when using PCA-Warp, the difference between the clean and the final pass is much smaller, since the increased difficulty of finding features in the final pass does not affect PCA-Warp.

The Figures 4.8 and 4.9 show the *best* examples for PCA-Warp relative to PCA-Flow, that is, the frames for which the difference $\text{EPE}_{PCA-Warp} - \text{EPE}_{PCA-Flow}$ is lowest. We show the top example for both the clean pass and the final pass. The main advantages of PCA-Warp is that it does not rely on matched features, and hence is able to estimate motion in regions that are not sufficiently textured to extract feature points. Nevertheless, as shown in Table 4.4, the average error in the Sintel training set is significantly higher for PCA-Warp compared to PCA-Flow.



(a) First input frame (b) Ground truth flow

(c) PCA-Warp, EPE=13.6 (d) PCA-Flow, EPE=14.1

Figure 4.8: Clean pass, best result for `PCA-Warp` relative to `PCA-Flow`

4.5.2 Quantitative evaluation on optical flow datasets

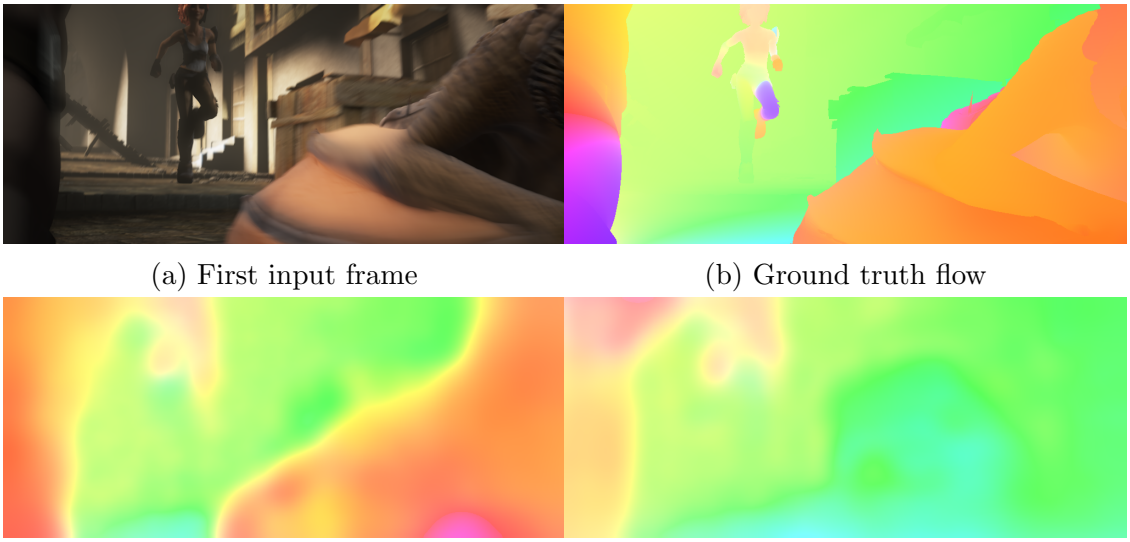
MPI-Sintel

Figure 4.10 shows an example from the clean pass of the training set of the MPI-Sintel optical flow benchmark for both PCA-Flow and PCA-Layers. PCA-Flow produces an oversmoothed optical flow field, but can correctly estimate most of the long-range motion. By computing and combining multiple layers, PCA-Layers is able to compute good motion and precisely locate motion boundaries.

On the Sintel test set, at the time of writing PCA-Flow ranked at place 22 of 36 on both the final pass (EPE = 8.65 px) and the clean pass (EPE = 6.83 px). While not the most accurate method, it only requires 190 ms per frame, while consistently outperforming the widely used methods LDOF and Classic+NL. Notably, we outperform GPUFlow [290], which we used to generate the training data. GPUFlow takes 2 s per frame, and achieves an average EPE of 12.64 px (clean) and 11.93 px (final).

Since the optical flow field generated by our method has a low spatial resolution, we compare it to Classic+NLP at an image resolution of 64x32 px. At this resolution, Classic+NLP achieves an EPE of 10.01 px, significantly worse than PCA-Flow, and requires 1.9 s per pair of frames.

PCA-Layers performs much better, with place 10 of 36 on the final pass, and 9 of 36 on the clean pass. It performs particularly well in the unmatched regions, where it ranks 5 of 36 on both passes. This demonstrates that our learned basis captures the structure of the optical flow well enough to make “educated guesses”



(a) First input frame (b) Ground truth flow

(c) PCA-Warp, EPE=31.2 (d) PCA-Flow, EPE=49.3

Figure 4.9: Final pass, best result for `PCA-Warp` relative to `PCA-Flow`

about regions that are only visible in one frame of a pair.

KITTI

In addition to Sintel, we tested our method on the KITTI benchmark [90]. Since KITTI contains scenes recorded from a moving vehicle, we expect the subspace of possible motions to be relatively low-dimensional. Figure 4.11 shows an example. Note how we are able to accurately estimate the motion of the hedge on the right side, and how the boundaries are much sharper using PCA-Layers.

On the KITTI test set we obtain an average EPE (`Avg-All`) on all pixels of 6.2 px for PCA-Flow and 5.2 px for PCA-Layers. While the flow in KITTI is purely caused by the low-dimensional motion of the car, a segmentation into layers helps to better capture motion boundaries. At time of writing, all other published methods faster than 5 s per frame perform worse in average EPE, the next best being TGV2CENSUS with 6.6 px at a runtime of 4 s. No CPU-based method with similar accuracy to ours is faster than 10 s per frame.

In the `Out-Noc` metric (percentage of pixels with an error > 3 px), PCA-Flow ranks 40 of 63 (15.67%) and PCA-Layers ranks 34 of 63 (12.02%). These results reflect the approximate nature of our flow fields.

4.5.3 Qualitative evaluation

In the following section, we show additional results from the training sets of MPI-Sintel (both clean and final passes) and KITTI. For each example, we show:

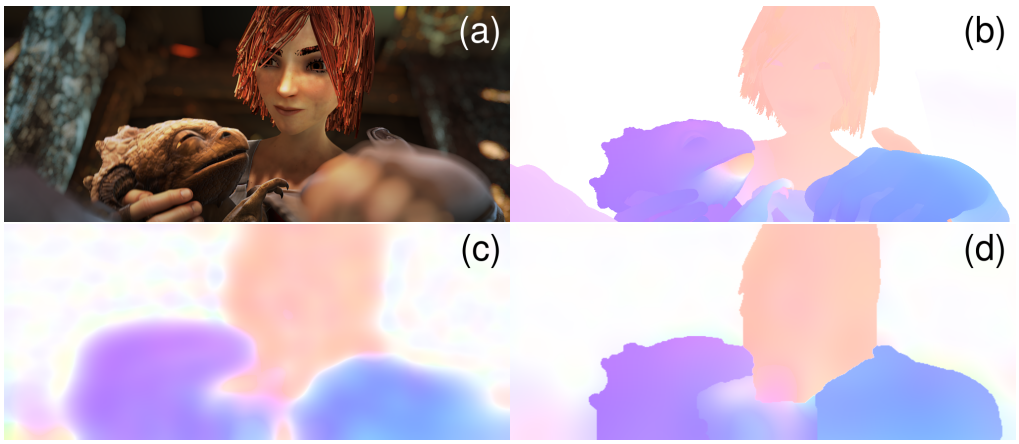


Figure 4.10: Results on MPI-Sintel: (a) Image; (b) Ground truth flow; (c) PCA-Flow; (d) PCA-Layers.

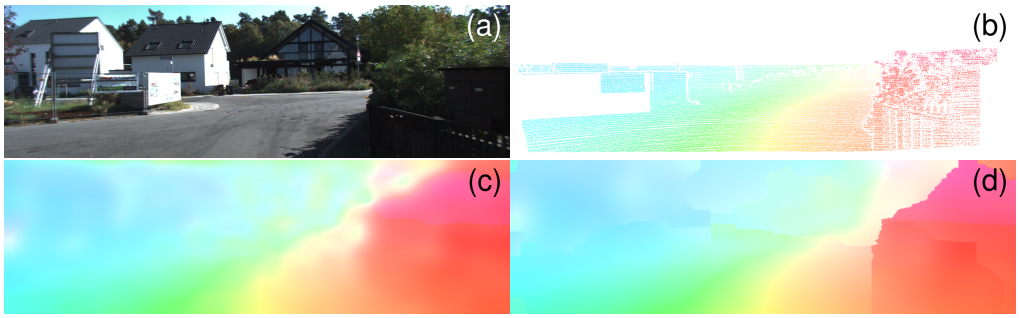


Figure 4.11: Results on KITTI: (a) Image; (b) Ground truth flow; (c) PCA-Flow; (d) PCA-Layers.

- (a) The first of the two input frames.
- (b) The ground truth optical flow.
- (c) The model assignment at each pixel. This can be seen as a coarse motion segmentation.
- (d) The estimated optical flow.
- (e) The homography, robustly fitted to all matched features.
- (f) The result of PCA-Flow, added as an additional motion proposal.
- (g)-(l) The individual motion models computed by our hard EM algorithm; each model also shows which tracked feature point contributes to it. If too few features are assigned to a given model, it is removed from the estimation, and not shown here.

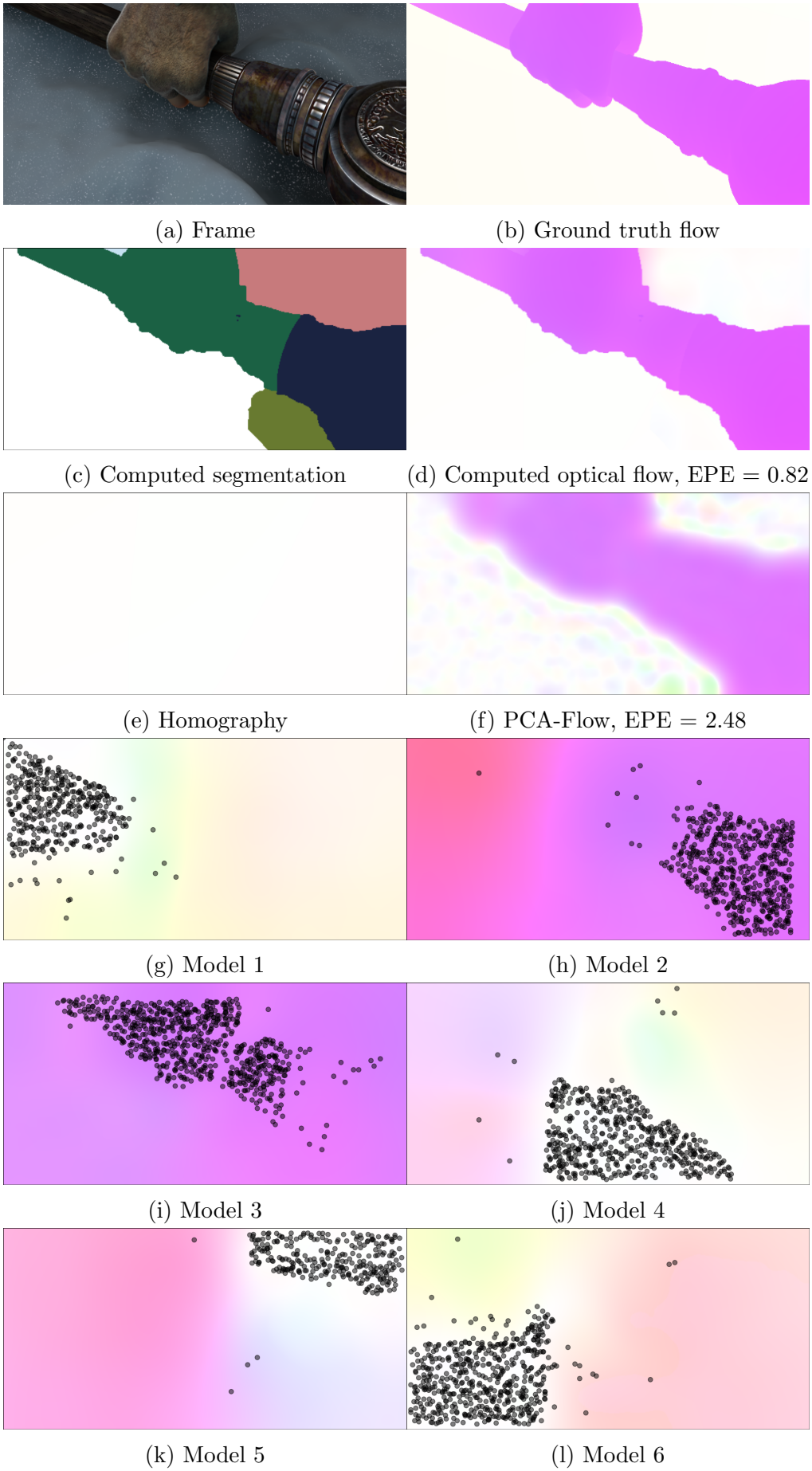
Note that all images were scaled to 512×256 pixel, since this is the resolution that our algorithms use internally.

Figures 4.12 and 4.13 show examples from the clean pass of MPI-Sintel, Figs. 4.14 and 4.15 show examples from the final pass of MPI-Sintel, and Figs. 4.16 and 4.17 show examples from KITTI. Since the motion in KITTI is inherently low dimensional, it is fairly well captured by the pure PCA-Flow approach; using multiple models does not improve the accuracy in terms of endpoint error. It does, however, reduce the number of "wrong" pixels with an error larger than 3 pixel. Additionally, as shown in the examples, it increases the accuracy near motion boundaries.

4.5.4 Timings

On a current CPU (Intel Xeon i7), the PCA-Flow algorithm takes on average 190 ms per frame on the MPI-Sintel dataset. 80 milliseconds are used for the feature matching. One advantage of our algorithm is that, when using longer sequences such as those from MPI-Sintel, the features for each frame have to be computed only once. Fitting the flow basis itself requires approximately 90 milliseconds. PCA-Layers is significantly slower, requiring on average 3.2 seconds per pair of frames. Our implementation uses Python and its OpenCV bindings. The core IRLS algorithm is implemented in C++ using Armadillo [207]. Figure 4.18 plots the best and fastest published methods on Sintel and KITTI in the EPE-runtime plane[5]. Generally, all methods faster than PCA-Flow require a GPU, and achieve a much higher endpoint error. PCA-Layers achieves a much lower error, at the cost of increased runtime.

[5]For Sintel, we used the timings as reported in the respective publications; for KITTI, we use the timings given in the public table.



(a) Frame (b) Ground truth flow

(c) Computed segmentation (d) Computed optical flow, EPE = 0.82

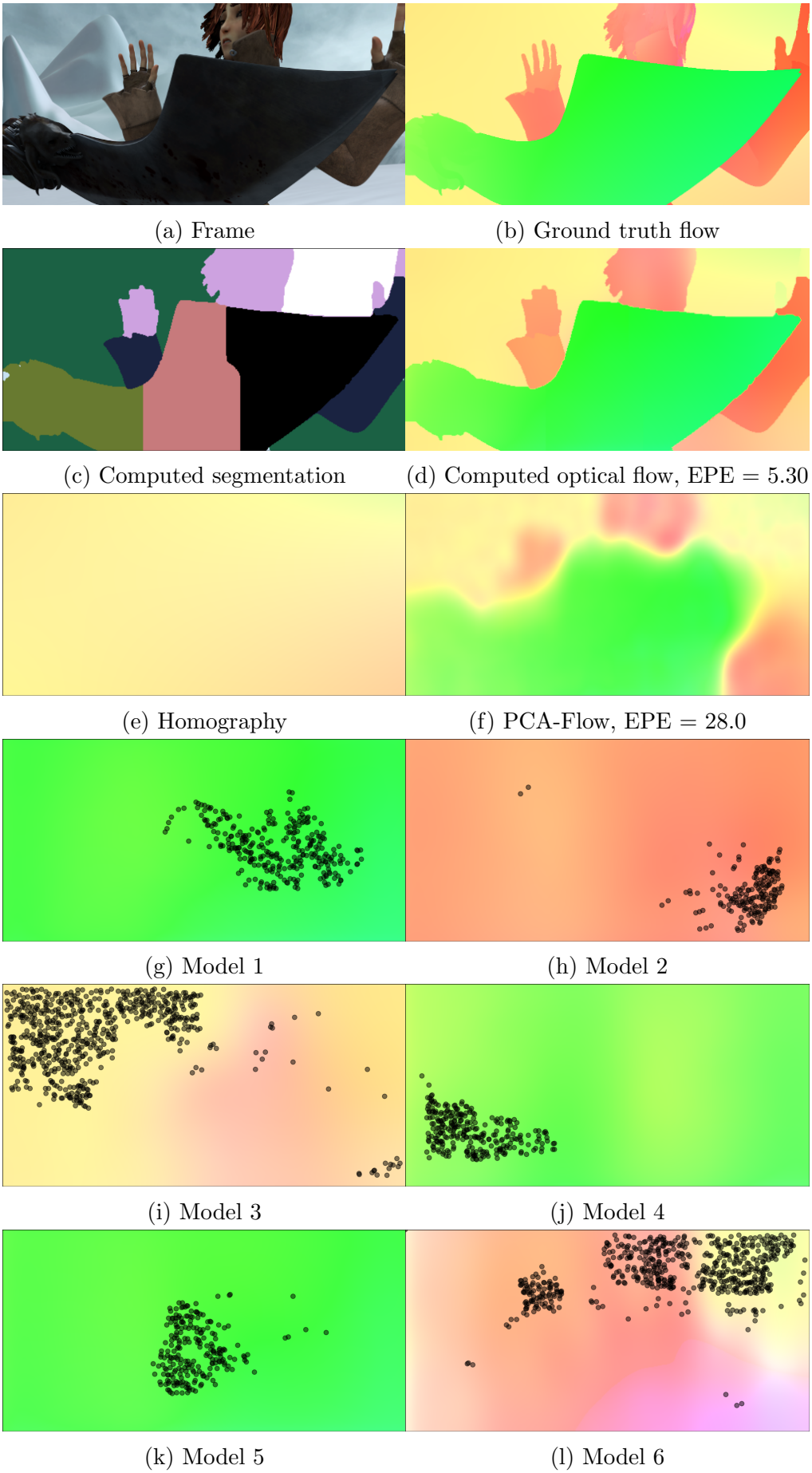
(e) Homography (f) PCA-Flow, EPE = 2.48

(g) Model 1 (h) Model 2

(i) Model 3 (j) Model 4

(k) Model 5 (l) Model 6

Figure 4.12: Example from `sintel-clean`.



(a) Frame (b) Ground truth flow

(c) Computed segmentation (d) Computed optical flow, EPE = 5.30

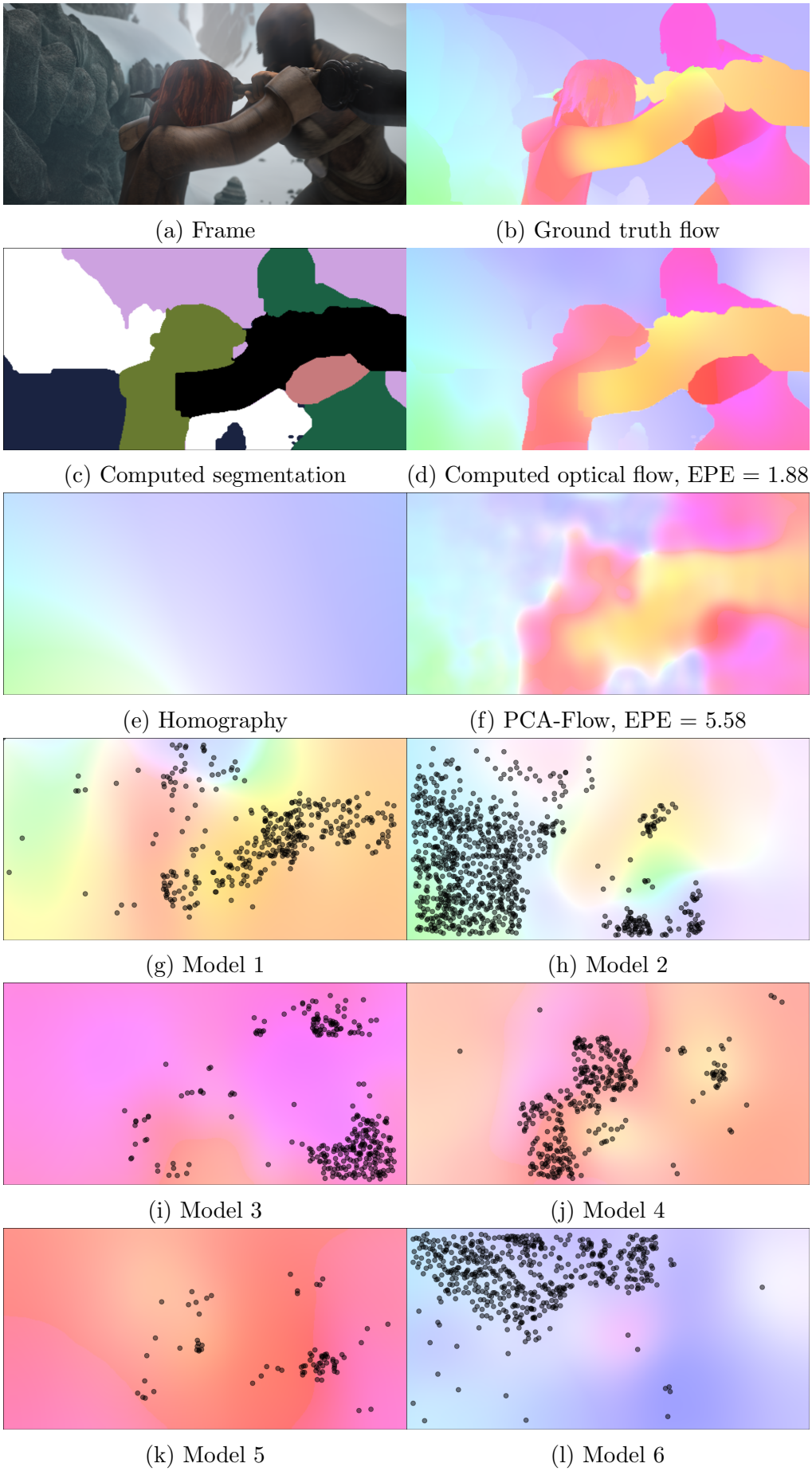
(e) Homography (f) PCA-Flow, EPE = 28.0

(g) Model 1 (h) Model 2

(i) Model 3 (j) Model 4

(k) Model 5 (l) Model 6

Figure 4.13: Example from `sintel-clean`.



(a) Frame (b) Ground truth flow

(c) Computed segmentation (d) Computed optical flow, EPE = 1.88

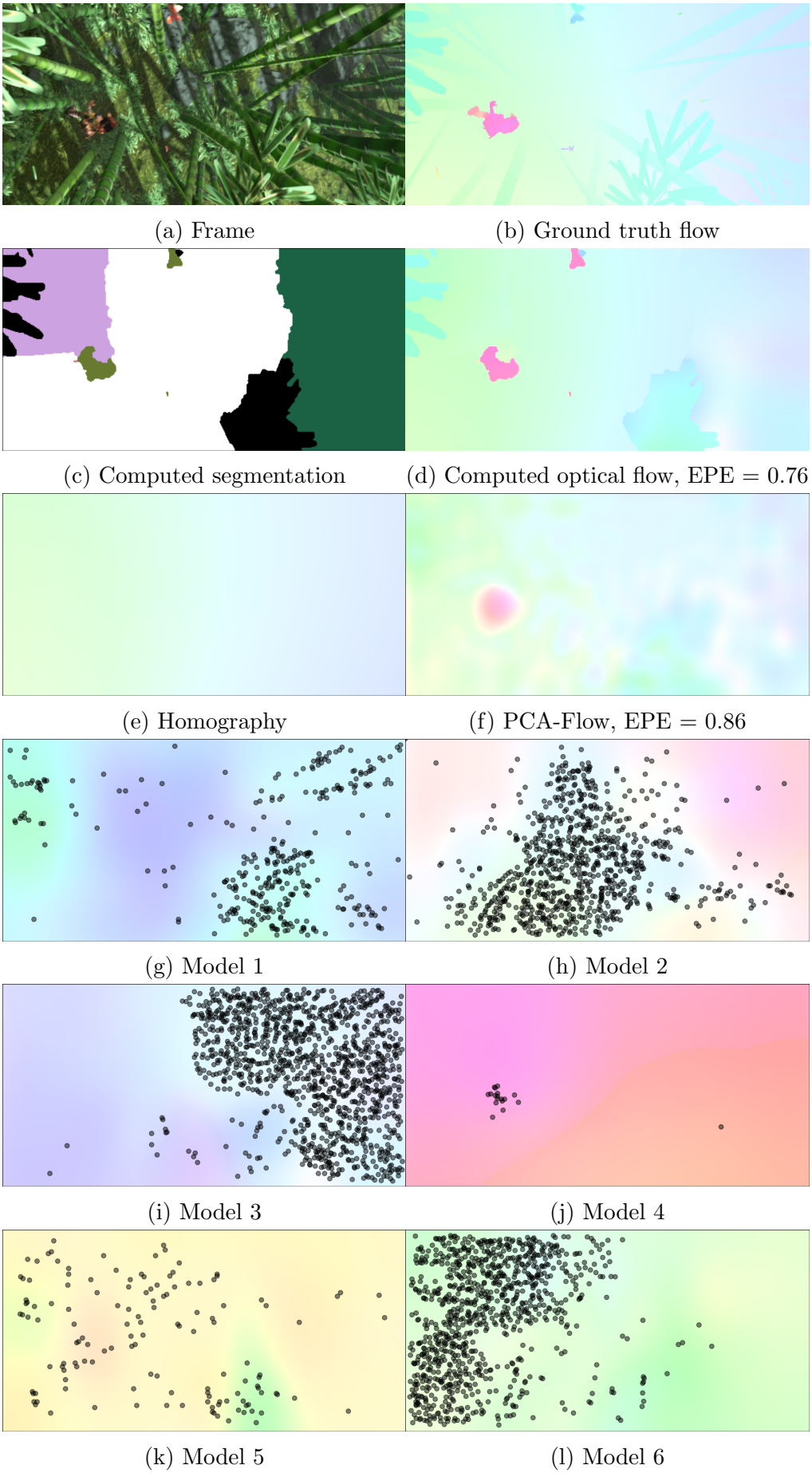
(e) Homography (f) PCA-Flow, EPE = 5.58

(g) Model 1 (h) Model 2

(i) Model 3 (j) Model 4

(k) Model 5 (l) Model 6

Figure 4.14: Example from `sintel-final`.



(a) Frame (b) Ground truth flow

(c) Computed segmentation (d) Computed optical flow, EPE = 0.76

(e) Homography (f) PCA-Flow, EPE = 0.86

(g) Model 1 (h) Model 2

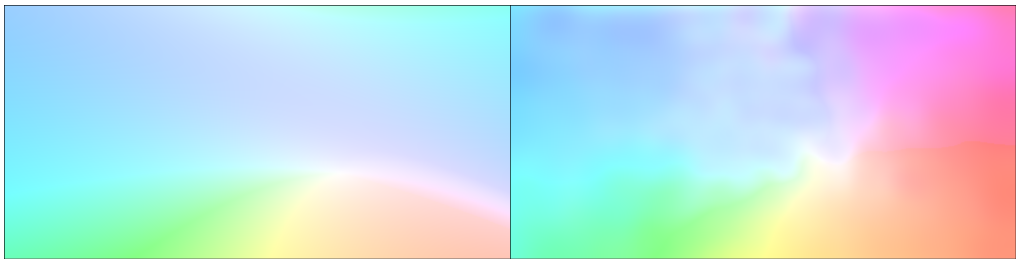
(i) Model 3 (j) Model 4

(k) Model 5 (l) Model 6

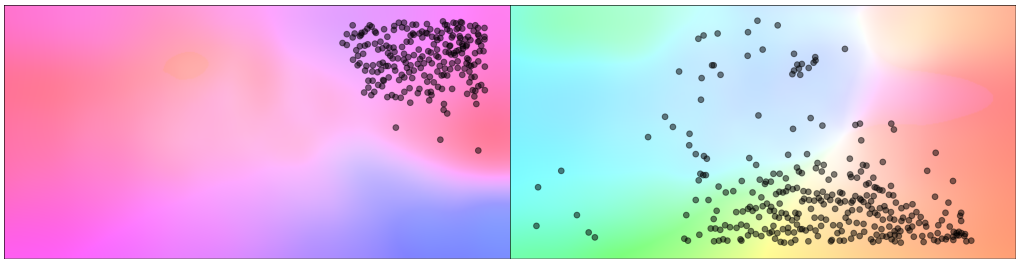
Figure 4.15: Example from `sintel-final`.

(a) Frame (b) Ground truth flow

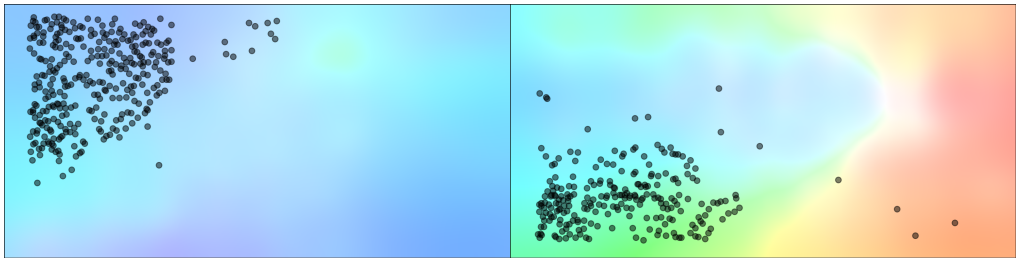
(c) Computed segmentation (d) Computed optical flow, EPE = 1.07



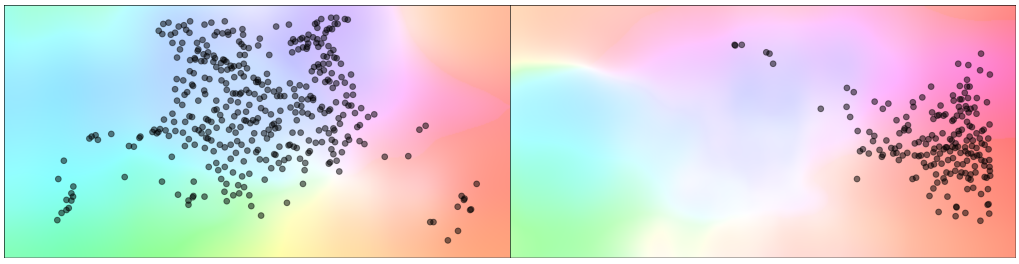
(e) Homography (f) PCA-Flow, EPE = 1.34



(g) Model 1 (h) Model 2

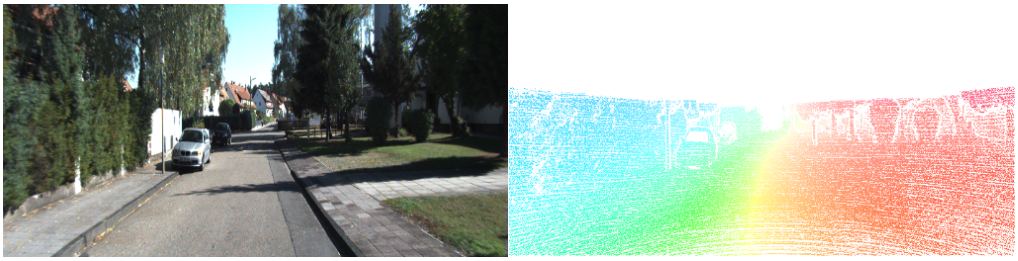


(i) Model 3 (j) Model 4



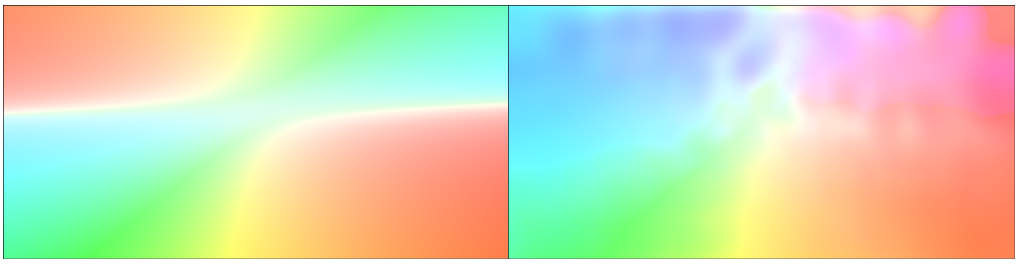
(k) Model 5 (l) Model 6

Figure 4.16: Example from KITTI.

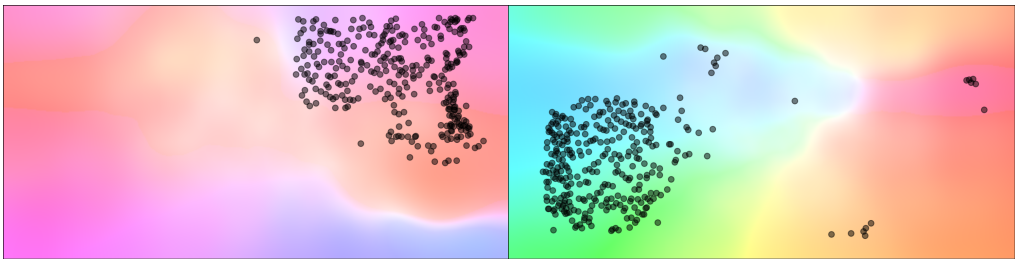


(a) Frame (b) Ground truth flow

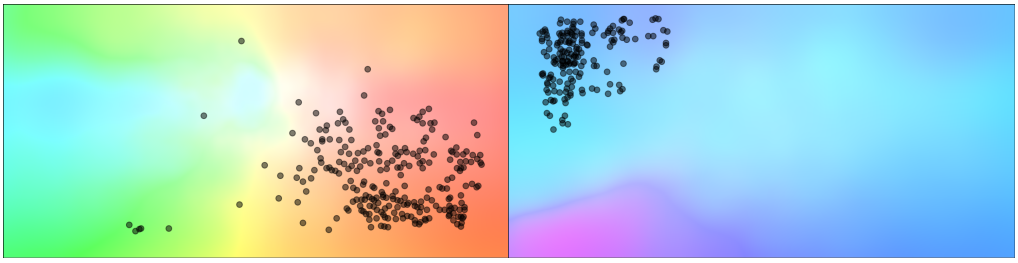
(c) Computed segmentation (d) Computed optical flow, EPE = 2.69



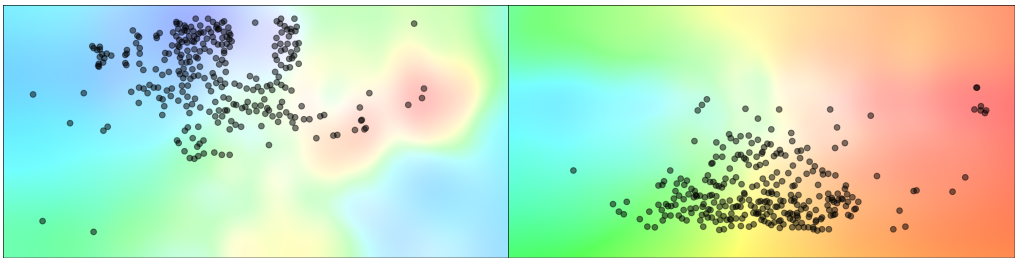
(e) Homography (f) PCA-Flow, EPE = 4.88



(g) Model 1 (h) Model 2

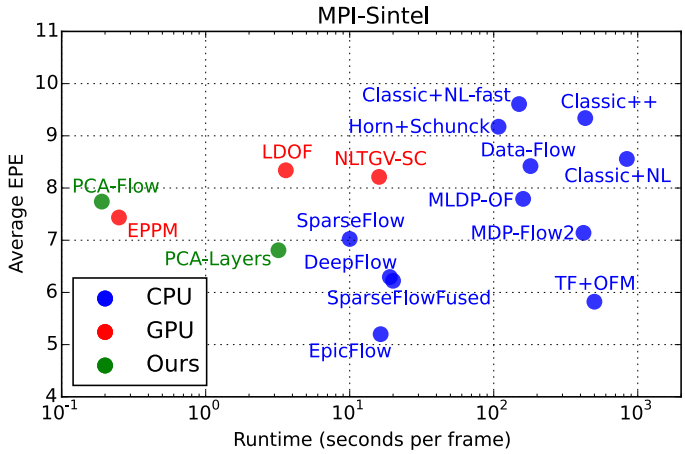


(i) Model 3 (j) Model 4

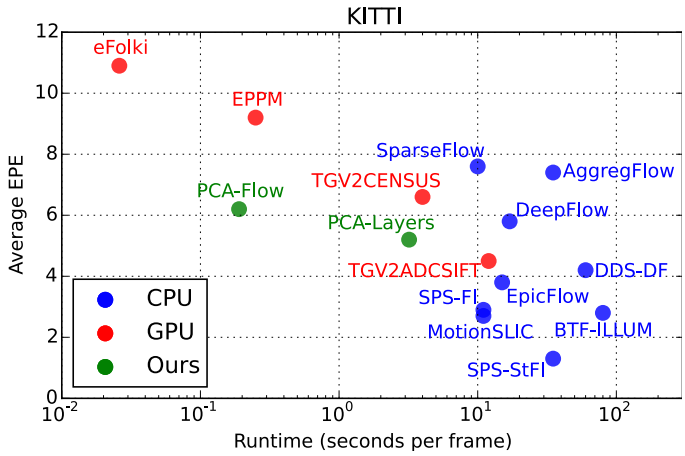


(k) Model 5 (l) Model 6

Figure 4.17: Example from KITTI.



(a) Sintel



(b) KITTI

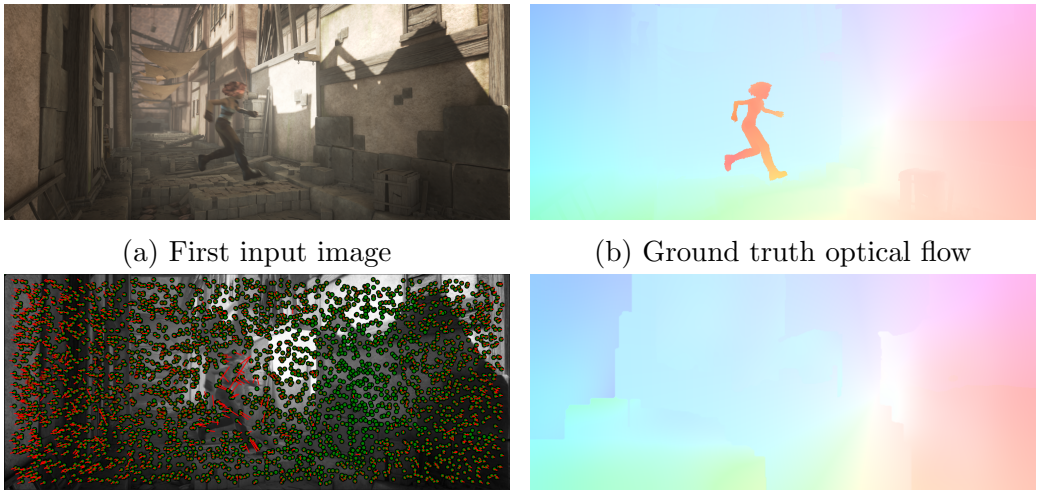
Figure 4.18: Avgerage EPE vs. runtime of PCA-Flow and PCA-Layers on Sintel (top) and KITTI (bottom).

4.5.5 Failure cases

If our algorithm fails, it is primarily for two reasons, missing or wrong features and large unstructured regions. Figs. 4.19 and 4.20 show examples for both cases.

In Fig. 4.19, the girl is absent from the estimated optical flow field. As can be seen from Fig. 4.19c, due to motion blur, not many features are detected on her body, especially on her legs. Even worse, the few features that are detected are very noisy, and are eliminated by the robust estimation. Better features can help improve the performance in cases like this; nevertheless, such features often come at higher computational cost. Whether they should be used or not is therefore dependent on the application; here, we decided against it.

In Fig. 4.20, one can see a wrong, "blocky" structure in the background. While



(a) First input image (b) Ground truth optical flow

(c) Matched features (d) Estimated optical flow

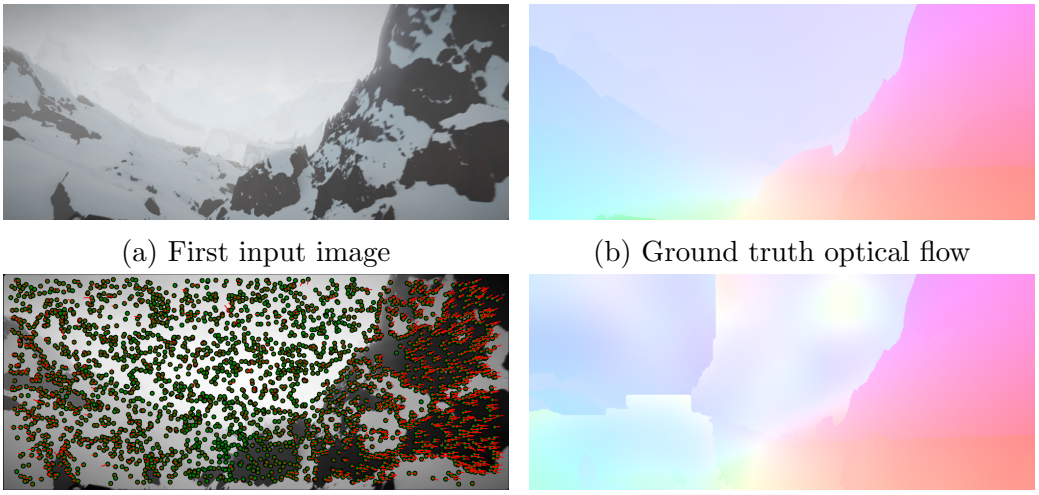
Figure 4.19: Failure case: Missing or incorrect features.

many features are found, they are not all assigned to the same model. In such a case and in the absence of other image cues, the MRF tends to create blocky assignments, causing artifacts at the seams. One way to fix this would be to use a better inference scheme than the simple pairwise MRF we currently use; for example a densely connected CRF [144].

4.6 Summary and conclusion

To summarize, this chapter demonstrated the feasibility of computing a basis for global optical flow fields from a large amount of training data, and showed how this basis can be used with different datasets and scenarios, showing good generalization capabilities. We proposed an algorithm to efficiently estimate approximate optical flow, using sparse feature matches and the learned basis. By integrating several PCA-Flow fields into a layered scene model, we are able to capture model discontinuities, while retaining the high computational efficiency of PCA-Flow. Results on two current, challenging datasets for optical flow estimation show that our method outperforms existing layered optical flow methods, while at the same time reducing computation time by two orders of magnitude, even on a standard CPU.

However, it should also be noted that, in terms of accuracy on current benchmarks, even PCA-Layers does not match state of the art methods, despite outperforming other layered methods. This points to fundamental issues with layered scene models. In the next chapter, we will describe these issues and propose an extension to layered world models that takes into account the three-dimensional structure of the scene. We will show how a method based on this extended world



(a) First input image (b) Ground truth optical flow

(c) Matched features (d) Estimated optical flow

Figure 4.20: Failure case: Artifacts in weakly structured regions.

model not just achieves state-of-the-art performance in optical flow benchmarks, but at the same time allows reasoning about the geometry of the scene using a very small number of input frames.

Chapter 5

From layers to geometry

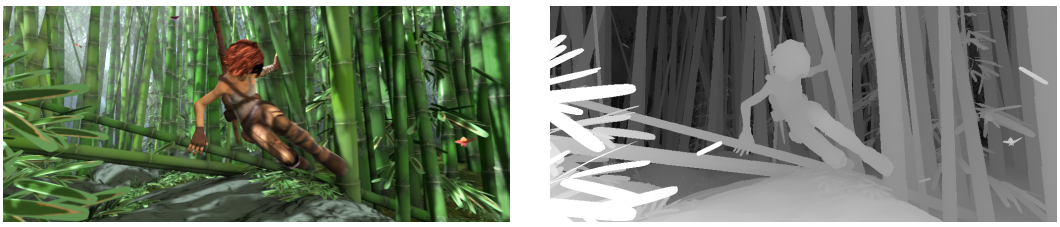
5.1 Introduction

As we have seen in the previous two chapters, layers allow us to approximate the depth structure of a scene and in particular model the effects occuring at depth discontinuities[1]. Yet, in current optical flow benchmarks, layered methods do not occupy the top spots, as can be seen in Fig. 4.18. The reason for this is that scenes with complex structure and non-frontoparallel surfaces are difficult to model using layers. To illustrate, consider Fig. 5.1, showing an image from MPI-Sintel and the corresponding depth map. Even in the context of layers, it is not obvious what would be the right exact representation for this scene: Using a low number of layers would group large parts of the background together, which would require modeling the complex depth and occlusion relationships of the background *within* a layer. Modeling each tree using a separate layer, on the other hand, would require a large number of layers and would quickly become computationally prohibitive, even using an approach such as PCA-Layers.

What, then, would be a better representation that would maintain the advantages of layers while being able to model a scene such as the one shown in Fig. 5.1? To answer this question, we note that the world is composed of things that move and things that do not. The 2D motion field, which is the projection of the 3D scene motion onto the image plane, arises from observer motion relative to the static scene and the independent motion of objects. A large body of work exists on estimating camera motion and scene structure in purely static scenes, generally referred to as Structure-from-Motion (SfM). On the other hand, methods that estimate general 2D image motion, or optical flow, make much weaker assumptions about the scene and can be applied to any image sequence. Neither approach fully exploits the mixed structure of natural scenes. Most[2] of what we see in such scenes is static - houses, roads, desks, etc. Using the usual SfM nomenclature, we refer to these

[1]This chapter is based on [278]. While working on this project, JW and LS were supported by the Max Planck ETH Center for Learning Systems.

[2]In KITTI-2015 and MPI-Sintel, independently moving regions make up only 15% and 28% of the pixels, respectively.



(a) Image (b) Depth

Figure 5.1: Image and depth map for an example sequence from MPI-Sintel. A proper layer segmentation for scenes with complex geometry is not obvious.

static parts of the scene as the *rigid scene*, or *rigid regions*[3]; these regions can have non-zero optical flow, but since they do not move by themselves, their optical flow is always caused by the motion of the observer. At the same time, moving objects like people, cars, and animals make up a small but often important part of natural scenes. Despite the long history of both SfM and optical flow, no state-of-the art optical flow method synthesizes both into an algorithm that works on general scenes like those in the MPI-Sintel dataset [54]. In this chapter, we propose such a method (Fig. 5.2). From three frames, our method computes a segmentation of the scene into rigid and moving regions, the depth structure of the scene, and the optical flow. Using multiple frames and regularizing the flow based on the underlying 3D scene structure, we are able to achieve higher accuracy in challenging image regions, such as occlusions.

For the rigid scene, the camera motion and depth structure fully determine the motion, which forms the basis of SfM methods. Since the goal of most SfM methods is the accurate reconstruction of the rigid scene geometry, moving objects are commonly treated as nuisances, and either explicitly excluded from the computation, or implicitly ignored by using robust error functions. Modern optical flow benchmarks, however, contain moving objects such as cars or bicycles in KITTI, or humans and dragons in Sintel. Assuming a fully static scene or treating these moving objects as outliers is hence not viable for optical flow algorithms; we want to reconstruct flow everywhere.

One possible approach is to segment the scene into regions corresponding to independently moving surfaces, which is commonly done by exploiting 3D motion cues and epipolar motion [5, 32, 188, 242, 263]. Commonly, those methods require a given optical flow field as input and their output is the segmentation; the optical flow itself is not refined. Furthermore, by using an of-the-shelf optical flow approach that does not take segmentation into account, errors in the initial optical flow will

[3]We make no statement whether a moving object is deforming or moving rigidly; here, the term *rigid* always refers to the static parts of the scene.

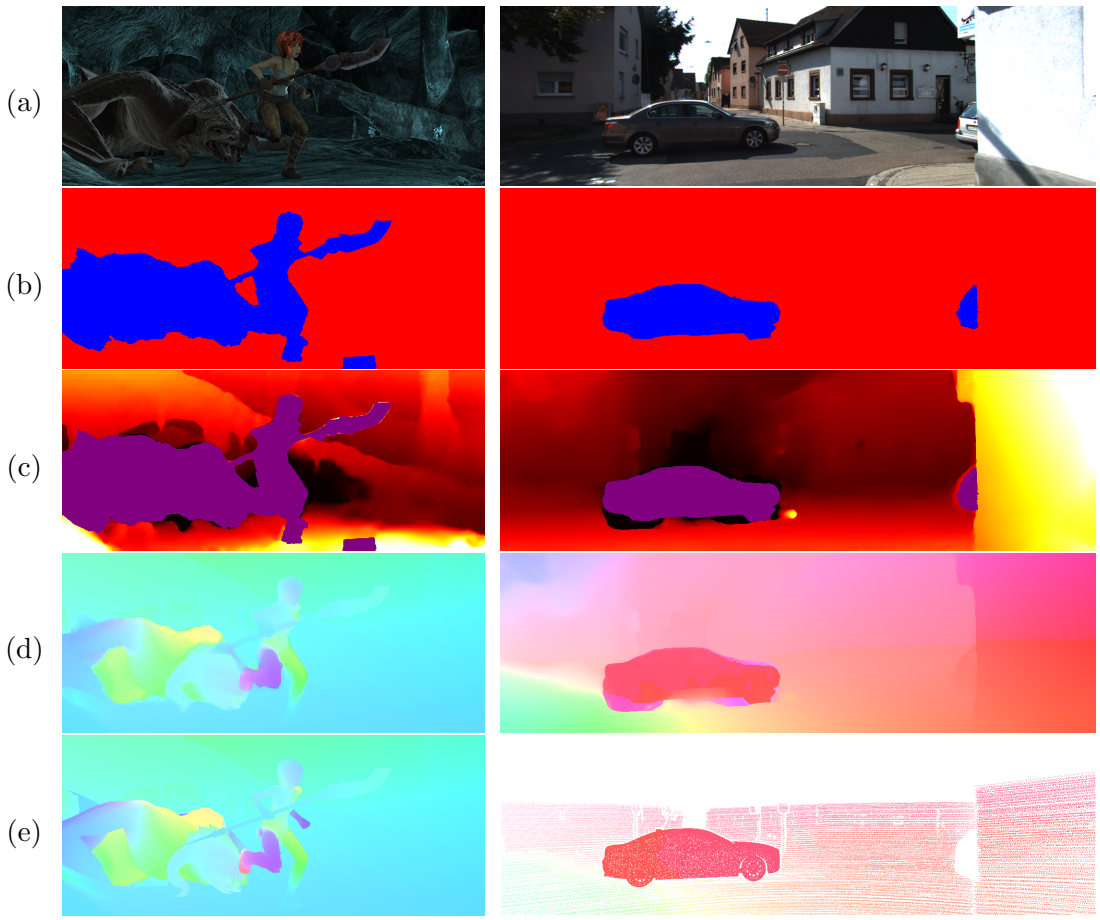


Figure 5.2: From three frames (a) our method computes a segmentation of the scene into rigid (red) and moving (blue) regions (b), the depth structure of the scene (c) with moving regions masked out in purple, and the optical flow (d). (e) shows ground truth flow. Using multiple frames and regularizing the flow based on the underlying 3D scene structure, we are able to achieve higher accuracy in challenging image regions, such as occlusions.

always be present in the segmentation.

Here, we note that independent motion in a scene typically arises from well defined objects with the ability to move. Hence, moving surfaces are not arbitrary, but instead have semantic meaning. This points to a possible solution. Recently, convolutional neural networks (CNNs) have achieved good performance on detecting and segmenting objects in images, and have been successfully incorporated into optical flow methods [17, 216]. Here we take a slightly different approach. We modify a common CNN and train it on novel data to transform the class labels into a score indicating whether an image region is static or belongs to a moving object,

taking into account that some objects (*e.g.* humans) are more likely to move than others (*e.g.* houses). This score is combined with additional motion cues to obtain an estimate of the rigid scene and independently moving regions.

After partitioning the scene into rigid and moving regions, we can deal with each appropriately. Since the motion of moving objects can be almost arbitrary, it is best computed using a classical unconstrained flow method. Taking the semantic object category into account helps [216, 173], but requires sophisticated class-dependent motion models or restricted scenarios. For reasons of simplicity, we use an existing accurate optical flow method [174] in the moving regions. The flow of the rigid scene, on the other hand, is extremely restricted, and only depends on the depth structure and the camera motion and calibration. In theory, one could use an existing SfM algorithm to reconstruct the camera motion and the 3D structure of the scene, and project this structure back to obtain the motion of the rigid scene regions. Two factors make this hard in practice. First, the overlap of different views of the 3D scene between frames in typical optical flow scenarios (*e.g.* taken from a fast car) is often small. Therefore, most optical flow methods only work on two or three consecutive frames. SfM algorithms, on the other hand, require tens or hundreds of frames to work reliably. Second, SfM algorithms require large camera baselines in order to reliably estimate the fundamental matrices. One example in which SfM algorithm fail is a small rotation and a small translation at the same time. In case of a long focal length, these cause virtually the same motion pattern on the image plane [117]. In video sequences, large baselines are rare, since the camera usually translates very little between frames. An exception to this are automotive scenarios such as the KITTI benchmark, where the recording car often moves rapidly and the frame rate is low.

Since full SfM is unreliable in general flow scenarios, we adopt the *Plane+Parallax* (P+P) framework [123, 124, 209]. In this framework, frames are registered to a common plane, which is aligned in all images after the registration. This removes the motion caused by camera rotation and simple intrinsic camera parameter changes, leaving parallax (caused by camera translation and depth variation) as the sole source of motion. Since all parallax is oriented towards or away from a common focus of expansion in the frame, computing the parallax is reduced to a 1D search problem and therefore easier than computing the full optical flow. Given the practical benefits, one may ask why P+P methods are not more prevalent in the leader boards of optical flow benchmarks. The problem is that such methods work only for purely static scenes; the work presented in this chapter addresses this issue.

The P+P framework brings an additional advantage: the parallax can be factored into a structure component, which is independent of the camera motion and constant across time, and a temporally varying camera component, which is a *single number per frame*. While factorization problems are generally hard, the structure of the problem is such that it can be solved easily. The structure component resulting from this factorization is always defined in the current reference frame. Since, by

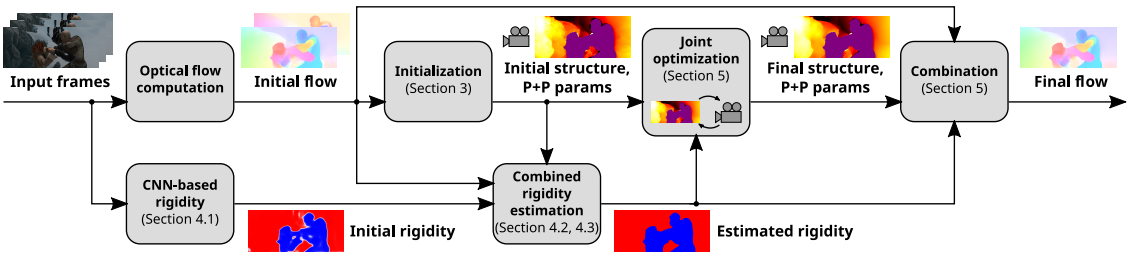


Figure 5.3: Algorithm overview. Given a triplet of frames, we first compute initial flow and an initial segmentation estimate based on a semantic segmentation CNN. The images are then aligned to a common plane, and the initial flow is converted to an estimate of the structure in the rigid scene using the Plane+Parallax framework. Where the P+P constraints are violated, the segmentation is refined, while at the same time the structure is refined using a variational optimization. To obtain the final flow estimate, the initial flow is used in moving regions, while the refined structure induces the flow in the rigid scene.

definition, the structure of the rigid scene does not change, the structure component is a convenient representation to integrate information across time. As described in Sec. 2.1.6, most existing optical flow algorithms do not consider more than two frames, and those that do often rely on heuristics such as constant velocity or constant acceleration. In contrast, using the structure component to reason about the motion of more than two frames at any given time is more principled and better able to model the actual temporal evolution of the images and the flow. By combining the structure information from multiple frames, our algorithm generates a better structure component for all frames, and fills in areas that are unmatched in a single pair of frames due to occlusion.

Additionally, the relationship between the structure component and the parallax (and thus, the optical flow) enables us to regularize the flow in a physically meaningful way, since regularizing the structure implicitly regularizes the flow. We use a robust second-order regularizer, which corresponds to a locally planar prior. We integrate the regularization into a novel objective function measuring the photometric error across three frames as a function of the structure and camera motion. This allows us to optimize the structure and also to recover from poor initializations. We call the method *MR-Flow* for *Mostly-Rigid Flow* and show an overview in Fig. 5.3.

Our algorithm takes as an input a triplet[4] of images $I_{t-1} \dots I_{t+1}$ and computes

[4]We use a triplet of frames since it allow us to treat most of the occlusion problems; our method can be readily extended to more than 3 frames as long as all frames can be registered to a common plane.

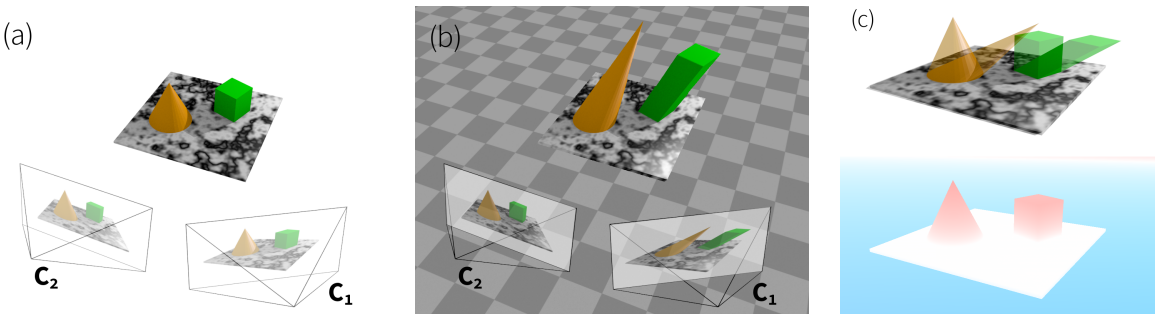


Figure 5.4: Illustration of P+P. (a) A scene is recorded from two cameras C_1 and C_2. (b) The image as seen by C_2 is projected to a plane in the image, and from there reprojected onto C_1. (c) Overlaying the two images (top) seen by C_1 shows that the remaining image motion (shown at the bottom) depends only on the distance from the plane, and always points towards or away from the epipole.

accurate, structure-aware optical flow fields from t to $t-1$ and $t+1$, a segmentation into the rigid scene and moving objects, and an estimation of the 3D scene structure. We test MR-Flow on MPI-Sintel [54], where no current methods employ a model of static scenes, and KITTI 2015 [173] (Fig. 5.2). Among published monocular methods, we achieve state-of-the-art results, ranking first on the clean pass of MPI-Sintel. Most importantly, we achieve good results across datasets unlike other methods which are either explicitly or implicitly tuned to a specific dataset. Our code, the trained CNN, and all data is available at [3].

In summary, we present three main contributions. First, we show how to combine semantic information with an initial motion estimation to segment the scene into rigid regions and independently moving objects. This allows us to use appropriately constrained estimation methods in different parts of the scene. Second, we extend previous plane+parallax methods to express the flow in the rigid regions via its depth structure. This allows us to regularize this structure instead of the flow field and to combine information across more than two frames. Third, we formulate the motion of the rigid regions as a single model. This allows us to iterate between estimating the structure and to recover from unstable initializations.

5.2 Plane + Parallax background

The Plane+Parallax paradigm has been developed for static scene analysis in the mid-1990's [123, 209]. Common use cases were the 3D reconstruction of a scene [123] and the detection of independent motions [122, 210, 288]. However, so far these methods have only been used for fairly simple scenes with dominant planes or motions, such as large traffic signs or the ground plane in aerial imagery.

The core idea of P+P is that stabilizing two frames with a planar motion (homography) removes the camera rotation and simplifies the geometric reasoning about structure [125, 248]. In the stabilized pair, motion is always oriented towards or away from the epipole and corresponds to parallax, which depends on the distance of the point from the plane in the 3D scene.

Estimating a planar homography can be done robustly and with more stability than estimating the fundamental matrix [124, 125]. Additionally, the plane does not need to correspond to a physical plane in the world; it suffices if it is spanned by three static points in the world [124]. While one is not able to estimate metric depth, the planar stabilization simplifies the matching process, turning the 2D optical flow estimation problem into a 1D problem that is equivalent to stereo estimation.

An illustration of the main principle is given in Fig. 5.4. The core idea of P+P is to align two or more images to a common plane $\mathbf{\Pi}$, so that

$$\mathbf{x} = \langle \mathbf{H}\mathbf{x'}_h \rangle \quad \forall(\mathbf{x}, \mathbf{x'}) \text{ corresponding to points on } \mathbf{\Pi} \tag{5.1}$$

where $\mathbf{x}$ and $\mathbf{x}'$ represent a point in the reference frame and the corresponding point in another frame of the sequence, $\mathbf{x}_h = \begin{pmatrix} \mathbf{x}^\top & 1 \end{pmatrix}^\top$ denotes $\mathbf{x}$ in homogeneous coordinates, $\mathbf{H}$ is the homography mapping the image of $\mathbf{\Pi}$ between frames, and $\langle \mathbf{z} \rangle = (z_1/z_3, z_2/z_3)$ is the perspective normalization.

This alignment removes the effects of camera rotation and the effect of camera calibration change (such as a zoom) between the pair of frames [291]. Getting rid of rotation is especially convenient, since the ambiguity between rotation and translation in case of small displacements is a major source of numerical instabilities in the estimation of the structure of the scene.

When computing optical flow between aligned images, the flow of the pixels corresponding to points on the plane is zero[5]. For an image point $\mathbf{x}$ corresponding to a 3D point $\mathbf{x}_w$ off the plane, the residual motion is given as [209]

$$\mathbf{u_p}(\mathbf{x}) = \frac{1}{1 - \frac{\mathbf{\Pi}(C_2)}{\mathtt{depth}(C_2)} \frac{\mathtt{depth}(\mathbf{x}_w)}{\mathbf{\Pi}(\mathbf{x}_w)}} (\mathbf{e} - \mathbf{x}), \tag{5.2}$$

where $\mathbf{\Pi}(C_2)$ is the distance of the second camera center to $\mathbf{\Pi}$, $\mathtt{depth}(\mathbf{x}_w)$ is the distance of point $\mathbf{x}_w$ to the first camera, $\mathtt{depth}(C_2)$ is the depth displacement of the second camera, $\mathbf{\Pi}(\mathbf{x}_w)$ is the distance of point $\mathbf{x}_w$ to $\mathbf{\Pi}$, and $\mathbf{e}$ is the common focus of expansion that coincides with the epipole corresponding to the second camera. This representation has two main advantages. First, instead of an arbitrary 2D vector, each flow is confined to a line; therefore computing the optical flow is reduced to a 1D search problem. Second, when considering the flow of a pixel to different frames

[5]Note that the plane does not have to correspond to a physical surface, but merely to a virtual plane within the static scene.

t which are registered to the same plane, Equation (5.2) can be written as

$$\mathbf{u_p}(\mathbf{x}, t) = \frac{s(\mathbf{x})b_t}{s(\mathbf{x})b_t - 1}(\mathbf{e}_t - \mathbf{x}), \tag{5.3}$$

where $s(\mathbf{x}) = \mathbf{\Pi}(\mathbf{x}_w)/\texttt{depth}(\mathbf{x}_w)$ is the structural component of the flow field. For a given reference frame, $s(\mathbf{x})$ is independent of time t, *i.e.* when computing the parallax between I_t and $I_{t'}$, s has to be equal, regardless of the time t' at which the second frame is recorded. It is hence convenient to accumulate structure over time via s. On the other hand, $b_t = \texttt{depth}(C_2)/\mathbf{\Pi}(C_2)$ encodes the camera motion to frame t', and is a single number per frame. To simplify notation, we express the residual flow in terms of the *parallax field* $h(\mathbf{x}, t)$, so that

$$\mathbf{u_p}(\mathbf{x}) = h(\mathbf{x}, t)\frac{\mathbf{e} - \mathbf{x}}{\|\mathbf{e} - \mathbf{x}\|}, \quad h(\mathbf{x}, t) = \frac{s(\mathbf{x})b_t\|\mathbf{e} - \mathbf{x}\|}{s(\mathbf{x})b_t - 1}. \tag{5.4}$$

Here, h denotes the flow in pixels along the line towards $\mathbf{e}$.

We can thus parametrize the motion across multiple frames as a common structure component s and per-frame parameters $\boldsymbol{\theta}_t = \{\mathbf{H}_t, b_t, \mathbf{e}_t\}$. Since we use the center frame of a triplet of frames as the reference and compute the motion to the two adjacent frames, from here on we denote the two parameter sets as $\boldsymbol{\theta}^+ = \{\mathbf{H}^+, b^+, \mathbf{e}^+\}$ for the forward direction and $\boldsymbol{\theta}^- = \{\mathbf{H}^-, b^-, \mathbf{e}^-\}$ for the backward direction.

To use the P+P framework across multiple frames, two main conditions need to be fulfilled. First, all frames have to be aligned to the *same plane* in 3D. Second, for Eq. (5.2) to hold, the four points defining the homography have to be *coplanar* in 3D. The following section describes our approach to enforce both conditions.

5.3 Initialization

Given a triplet of images and a coarse, image-based rigidity estimation (described in Sec. 5.4.1), the goal of our algorithm is to compute (i) a segmentation into rigid regions and moving objects and (ii) optical flow for the full frame. We start by computing initial motion estimates using an existing optical flow method [174]. For a triplet of images $\{I^-, I, I^+\}$, we compute four initial flow fields, $\mathbf{u_0^+}$ from I to I^+ and $\mathbf{u_0^-}$ from I to I^-, and their respective backwards flows $\bar{\mathbf{u}}_\mathbf{0}^+$ and $\bar{\mathbf{u}}_\mathbf{0}^-$. Due to the non-convex nature of our model (see Sec. 5.5) we need to compute good initial estimates for the P+P parameters $\boldsymbol{\theta}_0^+, \boldsymbol{\theta}_0^-$, visibility maps v^+, v^- denoting which pixels are visible and which are occluded in forward and backward directions, and an initial structure estimate s_0.

5.3.1 Initial alignment and epipole detection

First we compute the planar alignments (homographies) between frames. Since P+P only holds in the rigid scene, in this section we only consider points that are marked as rigid by the initial semantic rigidity estimation. While computing a homography between two frames is usually easy, two factors make it challenging in our case: (i) when aligning multiple frames, the plane to which the frames are aligned has to be equivalent for each frame for P+P to work, and (ii) the 3D points corresponding to the four points used to estimate the homographies have to be coplanar for Eq. (5.3) to hold. To make the last point clearer, consider fitting a basic homography to a pair of frames. When using four features, it is *always* possible to obtain a homography; if they are not coplanar, however, the projected 2D motion of the points will be modeled by the motion of a plane under strong changes of the intrinsic parameters. In this case, the basic assumption of Eq. (5.4) will not hold anymore.

To compute homographies obeying these constraints, we use a two-stage process. Let $\mathbf{x}_i$ be the i-th pixel, and $\mathbf{x}_i^+ = \mathbf{x}_i + \mathbf{u}_\mathbf{0}^+(\mathbf{x}_i), \mathbf{x}_i^- = \mathbf{x}_i + \mathbf{u}_\mathbf{0}^-(\mathbf{x}_i)$ its corresponding points in the forward and backward directions. Furthermore, let $\mathcal{J}$ be a set of 4 points, $\mathcal{J}^+, \mathcal{J}^-$ the sets of its forward and backward correspondences, and $\mathbf{H}_\mathcal{J}^{\{+,-\}}$ the homography fitted to the points in $\mathcal{J}$ and $\mathcal{J}^{\{+,-\}}$. The sets $\mathcal{K}_\mathcal{J}^{\{+,-\}}$ contain the points that are inliers according to $\mathbf{H}_\mathcal{J}^{\{+,-\}}$, for example

$$\mathcal{K}_\mathcal{J}^+ = \left\{ \mathbf{x}_i \middle| \| \mathbf{x}_i - \left\langle \mathbf{H}_\mathcal{J}^+ \mathbf{x}_i^+ \right\rangle \| < \epsilon \right\}. \tag{5.5}$$

An initial set of consistent forward and backward correspondences is then found by solving

$$\hat{\mathcal{J}} = \arg\max_\mathcal{J} \texttt{card}\left(\mathcal{K}_\mathcal{J}^+ \cap \mathcal{K}_\mathcal{J}^-\right) \tag{5.6}$$

using RANSAC, where $\texttt{card}\,(\mathcal{S})$ denotes the cardinality of set $\mathcal{S}$.

First, we compute initial homographies $\tilde{\mathbf{H}}^+, \tilde{\mathbf{H}}^-$ using RANSAC. In each iteration, the *same* random sample is used to fit both $\tilde{\mathbf{H}}^+, \tilde{\mathbf{H}}^-$, and a point is considered an inlier only when its reprojection error is low in both forward *and* backward directions.

Solving Eq. (5.6) selects the points that define the homographies with the largest set of inliers in both forward and backward direction. Assuming that this largest set of inliers correspond to correct correspondences, this ensures that the computed homographies belong to the same plane. If a computed homography displaces the images corners by more than half the image size, it is considered invalid. If no valid homography is found, our method returns the initial flow field. This happens on average in 2% of the frames. $\tilde{\mathbf{H}}^+$ and $\tilde{\mathbf{H}}^-$ are then set to $\mathbf{H}_{\hat{\mathcal{J}}}^+$ and $\mathbf{H}_{\hat{\mathcal{J}}}^-$, respectively.

The second step is to ensure the coplanarity of the points inducing the homographies. For this, we can turn around Eq. (5.3), and simultaneously refine

the homographies and estimate the epipoles $\mathbf{e}^{\{+,-\}}$ so that Eq. (5.3) holds. Let $\mathbf{u_r}(\mathbf{x}) = \langle \mathbf{H}(\mathbf{x} + \mathbf{u_0}(\mathbf{x}))_h \rangle - \mathbf{x}$ be the residual flow after registration with $\mathbf{H}$. Each pair $\mathbf{x}, \mathbf{u_r}(\mathbf{x})$ defines a residual flow line, and in the noise-free case, the epipole $\mathbf{e}$ is simply the intersection of these lines. Since the computed optical flow contains noise, we compute the epipole using the method described in [162], which we found to be sufficiently robust to noise. Therefore, $\mathbf{e} = \mathbf{e}(\mathbf{H}, \mathbf{u})$ is a function of the optical flow and of the computed homography.

Using $\mathbf{u_r}$ and $\mathbf{e}$, we can compute the signed distance of the line at $\mathbf{x}$ to the epipole,

$$d(\mathbf{x}, \mathbf{H}) = (\mathbf{e} - \mathbf{x})^\top \begin{pmatrix} -u_{r,2}(\mathbf{x}) \\ u_{r,1}(\mathbf{x}) \end{pmatrix} \frac{1}{\|\mathbf{u_r}(\mathbf{x})\|}. \tag{5.7}$$

As defined above, both $\mathbf{e}$ and $\mathbf{u_r}$ depend on $\mathbf{H}$. Enforcing coplanarity of the homographies is now equivalent to enforcing that the residual flow lines in both directions each pass through a common point as well as possible, *i.e.* that $d(\mathbf{x}, \mathbf{H})$ is as small as possible for all $\mathbf{x}$. The refined homographies are thus computed as

$$\mathbf{H_0}^+, \mathbf{H_0}^- = \underset{\mathbf{H}^+, \mathbf{H}^-}{\arg\min} \sum_{\mathbf{x}} \rho\left(d(\mathbf{x}, \mathbf{H}^+)\right) + \rho\left(d(\mathbf{x}, \mathbf{H}^-)\right). \tag{5.8}$$

While Eq. (5.8) is highly non-linear, we found that initializing with $\tilde{\mathbf{H}}^{\{+,-\}}$ and using a standard non-linear minimization package (here, we use L-BFGS [184]) produced results that greatly improved the final flow error compared to using the unrefined homographies $\tilde{\mathbf{H}}^{\{+,-\}}$. Throughout this chapter, we use the Lorentzian

$$\rho(z) = \frac{\sigma^2}{2} \log\left(1 + \left(\frac{z}{\sigma}\right)^2\right) \tag{5.9}$$

as the robust function, and compute the scaling parameter σ via the MAD [42]. The initial epipolar estimates $\mathbf{e_0}^{\{+,-\}}$ are computed using $\mathbf{H_0}^{\{+,-\}}$.

To initialize b^+, b^-, we first compute the parallax fields by projecting $\mathbf{u_r}$ onto the parallax flow lines,

$$h = \mathbf{u_r}^\top \frac{\mathbf{e} - \mathbf{x}}{\|\mathbf{e} - \mathbf{x}\|}. \tag{5.10}$$

Inserting (5.10) into (5.4) and solving for s, we get

$$s = \frac{h}{b\left(\|\mathbf{e} - \mathbf{x}\| - h\right)}. \tag{5.11}$$

Note that Eq. (5.3) contains a scale ambiguity between the structure s and the camera motion parameter b. Therefore, we can freely choose one of b^+, b^-, which only affects the scaling of s; we choose b_0^+ so that the initial forward structure s^+ defined by Eq. (5.11) has a MAD of 1. Since s^- is a function of b^- and should be

as close as possible to s^+, we obtain the estimate b_0^- by solving

$$b_0^- = \arg\min_{b^-} \sum_{\mathbf{x}} \rho\left(s_0^+(\mathbf{x}) - s^-(\mathbf{x})\right), \tag{5.12}$$

where $s^-(\mathbf{x})$ is a function of b^-, as defined in (5.11). Using b_0^-, we compute the initial backward structure s_0^- using Eq. (5.11), and set the full sets of P+P parameters to $\boldsymbol{\theta}_0^+ = \{\mathbf{H_0}^+, b_0^+, \mathbf{e_0}^+\}$, and $\boldsymbol{\theta}_0^-$ accordingly.

5.3.2 Occlusion estimation

Pixels can become occluded in both directions. In occluded regions, we expect the flow to be wrong, since it can at best be extrapolated. Given the initial flow fields, $\mathbf{u_0^+}, \bar{\mathbf{u}}_\mathbf{0}^+, \mathbf{u_0^-}, \bar{\mathbf{u}}_\mathbf{0}^-$ we compute the visibility masks $v^+(\mathbf{x})$, $v^-(\mathbf{x})$ using a forward-backward check [137]:

$$v^+(\mathbf{x}) = \delta\left[\|\mathbf{u_0^+}(\mathbf{x}) + \bar{\mathbf{u}}_\mathbf{0}^+(\mathbf{x} + \mathbf{u_0^+}(\mathbf{x}))\| < \tau_{cons}\right] \tag{5.13}$$

$$v^-(\mathbf{x}) = \delta\left[\|\mathbf{u_0^-}(\mathbf{x}) + \bar{\mathbf{u}}_\mathbf{0}^-(\mathbf{x} + \mathbf{u_0^-}(\mathbf{x}))\| < \tau_{cons}\right] \tag{5.14}$$

Since we assume occlusions to be present only in forward or backward direction, but not in both, we check where both $v^+(\mathbf{x})$ and $v^-(\mathbf{x})$ are zero, and at those locations set $v^+(\mathbf{x}) = v^-(\mathbf{x}) = 1$.

5.3.3 Initial structure estimation

Using the computed structure maps $s_0^{\{+,-\}}$ and visibility maps $v^{\{+,-\}}$, the initial estimate for the full structure is

$$s_0(\mathbf{x}) = \frac{1}{v^+(\mathbf{x}) + v^-(\mathbf{x})}\left(v^+(\mathbf{x})s_0^+(\mathbf{x}) + v^-(\mathbf{x})s_0^-(\mathbf{x})\right). \tag{5.15}$$

5.4 Rigidity estimation

Different cues provide different, complementary information about the segmentation of the scene. The semantic category of an object tells us whether it is capable of independent motion, static scene parts have to obey the parallax constraint (5.3), and the 3D structure of static parts cannot change over time. We integrate all of them in a probabilistic framework to estimate a segmentation map of the scene, marking each pixel as belonging to the rigid scene or to a moving object.

Figure 5.5: Results of rigidity estimation on the MPI-Sintel test set. Top: Original frame; Bottom: probability map of rigidity (white is likely rigid, black likely moving independently).

Figure 5.6: Results of rigidity estimation on KITTI 2015. Top: Original frame; Bottom: probability map of rigidity.

5.4.1 Semantic rigidity estimation

We leverage the recent progress of CNNs for semantic segmentation to predict rigid regions and independently moving regions in the scene[6]. In short, we model the relationship between an object's appearance and its ability to move. The semantic category of an object or region is related to whether the object is capable of independent motion, and thus whether it is likely to move relative to the scene.

Obviously object appearance alone does not fully determine whether something is moving independently. A car may be moving, if driving, or static, if parked. However, for the purpose of motion estimation, not all errors are the same. Assuming an object is static when in reality it is not imposes false constraints that hurt the estimation of the global motion, while assuming a rigid region is independently moving does little harm. Thus, when in doubt, we predict a region to be independently moving, and hence do not impose any constraints on the motion of such a region.

The main optical flow benchmarks, KITTI-2015 and MPI-Sintel, provide different training data. While the essence of our model is the same for both, our training process varies to adapt to the available data. In both cases we start with the DeepLab architecture [59], pre-trained on the 21 classes of Pascal VOC [79], substitute all fully connected layers with convolutional layers, and use the atrous algorithm [165] to densify the predictions [216]. Both networks produce a rigidity score between 0 and 1 which we call the semantic rigidity probability p_s.

MPI-Sintel contains many objects that are not contained in Pascal VOC, such

[6]The networks described in this section were implemented and trained by Laura Sevilla.

as dragons. Thus using the CNN to predict a semantic segmentation is not possible. Also, no ground truth semantic segmentation is provided, so training a CNN to recognize these categories is not possible. However, the dataset provides ground truth camera calibration, depth and optical flow for the training set, and this data can be used to generated ground truth rigidity maps.

To obtain this data, we first reproject each point in the reference frame back into the three-dimensional coordinate space

$$\mathbf{x}_w = \texttt{depth}(\mathbf{x})\mathbf{P}_t^{-1}\left(\mathbf{x}_h\right)^\top \tag{5.16}$$

and project the 3D point $\mathbf{x}_w$ into the next frame

$$\mathbf{x'}_{static} =< \mathbf{P}_{t'}\mathbf{x}_w > . \tag{5.17}$$

Here, $\mathbf{P}_t$ is the camera matrix at time t, and $\texttt{depth}(\mathbf{x})$ is the depth value at 2D coordinate $\mathbf{x}$, given by the dataset. If a point would belong to the rigid scene, the optical flow at this location would hence be given as $\mathbf{u}_{static} = \mathbf{x'}_{static} - \mathbf{x}$. Therefore, the ground truth rigidity map can be computed as the difference between the ground truth optical flow, and the hypothetical optical flow in case of a static surface.

$$r_{gt}(\mathbf{x}) = \delta\left[\|\mathbf{x'}_{static} - \mathbf{x} - \mathbf{u}_{gt}(\mathbf{x})\| < \tau_{static}\right]. \tag{5.18}$$

Empirically, we set $\tau_{static} = 0.1$ pixel. The full ground truth segmentations are available at [3].

We modify the last layer of the CNN to predict 2 classes, rigid and independently moving, instead of the original 21. We train using the last 30 frames of each sequence in the training set, and validate on the first 5 frames of each sequence. Sequences shorter than 50 frames are only included in the validation set. For long sequences, this split puts the beginning and end frames of the same sequence into the validation and training sets, respectively, and hence there are common image contents between the training and validation sets. Yet, it is important to note that both training and validation sets only include sequences from the MPI-Sintel training set, so it will not affect the fairness of the results on the test set. At test time, the probability of being rigid is computed at each pixel and then thresholded. Examples of the estimated rigidity maps can be seen in Fig. 5.8.

To train the network, we start with one of the latest released versions of the DeepLab architecture, called DeepLab-Coco-LargeFov, which is pretrained including extra annotations from the MS-COCO dataset[7]. We modified the output layer to classify pixels as rigid or nonrigid. We fine-tuned all layers using the same parameters as before for 1.4K iterations. This small number of iterations was selected to

[7]Further details can be found on their web page `http://ccvl.stat.ucla.edu/software/deeplab/deeplab-coco-largefov/`

avoid overfitting. At test time, we obtain a probability of rigidity, and we compute the final estimate of rigidity by thresholding at 0.5. The accuracy of the estimation on a validation set is 94.2%.

In **KITTI 2015**, some independently moving objects (*e.g.* people) are masked out from the depth and flow ground truth. Therefore, the approach we followed for MPI-Sintel cannot be used. The objects in KITTI, however, appear in standard datasets like the enriched Pascal VOC. We modify the last layer of the network to predict the 22 classes that may be present in KITTI (*e.g.* person or road) similar to [216]. This includes the 21 classes present in Pascal VOC, plus a generic background class. We then classify an object as moving if it has the ability to move independently (*e.g.* cars, or buses) and as rigid otherwise (including the background class). See Fig. 5.8 for examples of semantic rigidity estimation.

To train the network for KITTI, we followed the same procedure as previous work on semantic segmentation [216], since it was shown to be successful. We used the DeepLab architecture [59] modifying the output layer to classify 22 classes: aeroplane, bicycle, bird, boat, building, bus, car, cat, cow, dog, floor, grass, horse, motorbike, road, sheep, sidewalk, sky, train, water, person and background, where background includes anything that is not one of the other 21 classes. We initialized the weights with the VGG model trained on Pascal VOC and then fine-tuned it on our categories using a fixed momentum of 0.9, weight decay of 0.0005 and learning rate of 0.0001 for the first 100K iterations, reduced by 0.1 after every 50K steps, during 200K iterations. We used a dense CRF [145] on top, where the unaries are the CNN output at each pixel and the pairwise potentials are position and a bilateral kernel with both position and RGB values. The inference in the dense CRF model is performed using 10 steps of mean-field approximate inference. At test time we obtain a probability over the classes. We estimate rigidity by choosing the class with the highest probability, and classifying the pixel as rigid or non-rigid based on whether an object in the class is capable of moving independently (for example, car) or not (for example, building). The accuracy of rigidity classification on the training set is 96.09%, where rigid parts are correctly classified 96.93% of the time, and independent moving parts are correctly classified 91.51% of the time.

Note that the same approach we use for KITTI can be used for general video sequences by using a generic pre-trained semantic segmentation network together with a definition of which semantic classes can move and which are rigid. This allows our method to directly benefit from advances in semantic segmentation and novel, fine-grained semantic segmentation datasets.

5.4.2 Physical rigidity estimation

While the accuracy of the CNN-based rigidity estimation is surprisingly high, there are still occasions when it fails. Examples are a strong motion blur in the rigid regions which get classified as moving objects, causing a false positive. On the

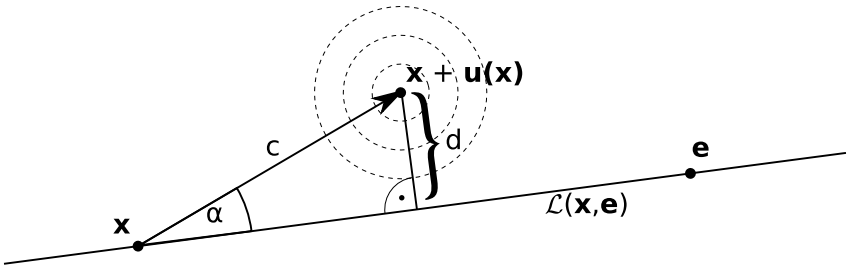


Figure 5.7: Rigidity probability, assuming Gaussian errors. See text for details.

other hand, a noisy or over-saturated appearance of objects can cause the semantic segmentation to fail and result in a false negative. Therefore, we combine the CNN-based rigidity estimation with two additional cues, motion direction and temporal consistency of the structure.

Moving regions from motion direction. A simple approach to classify a pixel as static or independently moving is to test whether its parallax flow points to the epipole [122]. This requires previously computed optical flow, turning it into a chicken-and-egg problem; the precomputed optical flow did not take the split into moving regions and the rigid scene into account, thus might contain incorrectly moving regions, which would be incorrectly classified as non-rigid. Still, they provide a useful additional source of information. Here, we employ a probabilistic framework for this classification.

For a given point $\mathbf{x}$, our model assumes the measured corresponding point $\mathbf{x}' = \mathbf{x}+\mathbf{u}(\mathbf{x})$ to have a Gaussian error distribution around the true correspondence with covariance matrix $\mathbf{\Sigma} = \sigma_m^2\mathbf{I}$. For a given rigid point (denoted by the conditioning on $r(\mathbf{x}) = 1$, in the following abbreviated as $r = 1$) and a given focus of expansion $\mathbf{e}$, the probability of the true correspondence pointing towards $\mathbf{e}$ is then given as

$$p\left(\mathbf{x}'|r=1,\mathbf{e}\right) = \int_{\mathbf{y}\in\mathcal{L}(\mathbf{x},\mathbf{e})} \frac{1}{2\pi\sigma_m^2} \exp\left(-\frac{1}{2}\mathbf{y}^{\mathbf{T}}\Sigma^{-1}\mathbf{y}\right) d\mathbf{y}, \tag{5.19}$$

where $\mathcal{L}(\mathbf{x},\mathbf{e})$ is the line going through $\mathbf{x}$ and $\mathbf{e}$. Figure 5.7 shows an illustration.

Since the error distribution is Gaussian, every marginal is also Gaussian. Therefore, the line integral in (5.19) is given as

$$p\left(\mathbf{x}'|r=1,\mathbf{e}\right) = \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp\left(-\frac{d^2}{2\sigma_m^2}\right), \tag{5.20}$$

where d denotes the distance of $\mathbf{x}'$ to $\mathcal{L}(\mathbf{x},\mathbf{e})$, as shown in Figure 5.7.

In the following, it will be convenient to express the corresponding point $\mathbf{x}'$ in

(5.20) in terms of the angle α and the displacement magnitude $c = \|\mathbf{u}(\mathbf{x})\|$. Then,

$$p\left(\alpha, c | r = 1, \mathbf{e}\right) = \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp\left(-\frac{c^2 \sin^2(\alpha)}{2\sigma_m^2}\right). \tag{5.21}$$

We can thus drop $\mathbf{e}$ from the equation. The likelihood of a point being rigid given the measurements α, c is now

$$\begin{aligned} p(r = 1|\alpha, c) &= \frac{p(\alpha|r = 1, c)p(r = 1)}{p(\alpha|r = 0, c)p(r = 0) + p(\alpha|r = 1, c)p(r = 1)} \\ &= \frac{\frac{1}{Z}p(\alpha, c|r = 1)p(r = 1)}{p(\alpha|r = 0, c)p(r = 0) + \frac{1}{Z}p(\alpha, c|r = 1)p(r = 1)}. \end{aligned} \tag{5.22}$$

Abbreviating the prior for rigidity $p(r = 1)$ as p_1 and setting $p(\alpha|r = 0, c) = \frac{1}{2\pi}$ (the motion of independently moving regions is assumed to be uniformly distributed), we get

$$p(r = 1|\alpha, c) = \frac{p_1 p(\alpha, c|r = 1)}{\frac{Z(1-p_1)}{2\pi} + p_1 p(\alpha, c|r = 1)} \tag{5.23}$$

and for uninformative priors $p_1 = 0.5$

$$p(r = 1|\alpha, c) = \frac{p(\alpha, c|r = 1)}{\frac{Z}{2\pi} + p(\alpha, c|r = 1)}. \tag{5.24}$$

What remains is to compute the normalization Z. Using Eq. (5.21), it can be computed as

$$\begin{aligned} Z = p(c|r = 1) &= \int_0^{2\pi} p(\alpha, c|r = 1)\mathrm{d}\alpha \\ &= \frac{1}{\sqrt{2\pi\sigma_m^2}} \int_0^{2\pi} \exp\left(-\frac{c^2 \sin^2(\alpha)}{2\sigma_m^2}\right) \mathrm{d}\alpha \\ &= \frac{1}{\sqrt{2\pi\sigma_m^2}} \int_0^{2\pi} \exp\left(-z(1 - \cos(2\alpha))\right) \mathrm{d}\alpha \qquad \left[z = \tfrac{c^2}{4\sigma_m^2}, \sin^2(x) = \tfrac{1}{2}(1 - \cos(2x))\right] \\ &= \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp\left(-z\right) \int_0^{2\pi} \exp\left(z\cos(2\alpha)\right) \mathrm{d}\alpha \\ &= \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp\left(-z\right) \frac{1}{2} \int_0^{4\pi} \exp\left(z\cos(\beta)\right) \mathrm{d}\beta \qquad [\beta = 2\alpha] \\ &= \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp\left(-z\right) \int_0^{2\pi} \exp\left(z\cos(\beta)\right) \mathrm{d}\beta \end{aligned}$$

$$= \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp(-z)\, 2\pi \mathbb{I}_0(z)$$
$$= \frac{\sqrt{2\pi}}{\sigma_m} \exp(-z)\, \mathbb{I}_0(z), \tag{5.25}$$

with $\mathbb{I}_0(x)$ the modified Bessel function of the first kind. Inserting Eq. (5.25) and Eq. (5.21) into Eq. (5.24) yields the likelihood of a point being rigid as

$$\begin{aligned} p(\mathbf{x} \text{ is rigid}) &= p(r = 1 | \alpha, c) \\ &= \frac{\exp\left(-2z \sin^2(\alpha)\right)}{\exp(-z)\mathbb{I}_0(z) + \exp\left(-2z \sin^2(\alpha)\right)}. \end{aligned} \tag{5.26}$$

with $z = c^2/(4\sigma_d^2)$. For numerical stability, mathematics packages often provide a function to compute $\exp(-x)\mathbb{I}_0(x)$; in SciPy, this function is available as `special.ive()`. Solving for both forward and backward directions yields the direction-based rigidity probabilities p_d^+ and p_d^-. These are then combined into the final direction-based rigidity probability using the visibility maps

$$p_d(\mathbf{x}) = \begin{cases} \frac{1}{v^+(\mathbf{x}) + v^-(\mathbf{x})} \left(v^+(\mathbf{x}) p_d^+(\mathbf{x}) + v^-(\mathbf{x}) p_d^-(\mathbf{x}) \right) & \text{if } v^-(\mathbf{x}) + v^+(\mathbf{x}) > 0 \\ 1/2 & \text{otherwise.} \end{cases} \tag{5.27}$$

Moving regions from structure consistency. While the previous approaches generally identify moving regions, there are still occasions where they fail. For example, the CNN-based rigidity estimation often fails in regions with low textural structure, while the motion-based rigidity estimation fails when an object moves towards the focus of expansion, such as a car driving parallel to the observer's motion. To address such cases, we compute a rigidity probability based on the temporal consistency of the structure.

Recall that according to the P+P framework the structure of the rigid scene is independent of time. In rigid regions that are visible in all frames, we assume the forward and backward structure s^+ and s^- to be close to each other. A structure based rigidity estimate p_s can thus be computed as

$$p_s(\mathbf{x}) = \begin{cases} \exp\left(-\left(s^+(\mathbf{x}) - s^-(\mathbf{x}) \right)^2 / \sigma_s^2 \right) & \text{if } v^-(\mathbf{x}) v^+(\mathbf{x}) = 1 \\ 1/2 & \text{otherwise.} \end{cases} \tag{5.28}$$

The reason for the fallback condition is that, if an occlusion is estimated in either the forward or backward direction, the structure estimation from the respective direction is assumed to be wrong; hence we set this probability to be uninformative.

Combined rigidity probability from motion. The motion-based probabilities p_d, p_s can be seen as orthogonal. Surfaces that move independently along the

parallax direction are considered to be rigid according to p_d, while surfaces that move by small amounts orthogonal to the parallax direction are considered to be rigid according to p_s. Hence, for a region to be considered *actually* rigid, we require both p_d and p_s to be high. The final motion-based rigidity probability p_m is

$$p_m(\mathbf{x}) = \begin{cases} p_d(\mathbf{x})p_s(\mathbf{x}) & \text{if } v^+(\mathbf{x})v^-(\mathbf{x}) = 1 \\ (p_d(\mathbf{x}) + p_s(\mathbf{x}))/2 & \text{otherwise.} \end{cases} \tag{5.29}$$

5.4.3 Combining rigidity estimates

The previously computed rigidity probabilities p_c, p_m yield per-pixel rigidity probabilities. To combine those into a coherent estimate, we first compute a rigidity unary

$$p_r(\mathbf{x}) = \lambda_{r,c} p_c(\mathbf{x}) + (1 - \lambda_{r,c})\, p_m(\mathbf{x}) \tag{5.30}$$

and the corresponding energy

$$E_r(r(\mathbf{x}), \mathbf{x}) = -r(\mathbf{x}) \log p_r(\mathbf{x}) - (1 - r(\mathbf{x})) \log (1 - p_r(\mathbf{x})) \tag{5.31}$$

with $r(\mathbf{x}) = 1$ if $\mathbf{x}$ is rigid, and 0 otherwise. Since we expect the segmentation to be spatially coherent, we estimate the full labeling by solving

$$\hat{r} = \arg\min_r \sum_{\mathbf{x}} E_r\left(r(\mathbf{x}), \mathbf{x}\right) + \lambda_{r,p} \sum_{\mathbf{y} \in \mathcal{N}(\mathbf{x})} m(I(\mathbf{x}), I(\mathbf{y}))\, [r(\mathbf{x}) \neq r(\mathbf{y})] \tag{5.32}$$

where $m(I_t(\mathbf{x}), I_t(\mathbf{y}))$ is the image-based Potts modulation from [202] and $\mathcal{N}(\mathbf{x})$ is the 8-connected neighborhood of $\mathbf{x}$. Eq. (5.32) is solved using TRWS [7].

Figure 5.8 (top) shows the importance of combining different cues to recover from errors and accurately estimate the segmentation. The semantic estimation (b) misses a large part of the dragon's head, while both the direction-based (b) and structure-based estimations misclassify different segments of the scene. Combining cues yields a good estimate (e).

5.5 Model and optimization

Model. The final structure should fulfill a number of criteria. First, as in the classical flow approach, warping the images using the flow induced by the structure should result in a low photometric error. Second, we assume that our initial flow fields are reasonable, hence, the final structure should be similar to the structures defined by the initial forward and backward flow. Third, the structure directly corresponds to the surface structure of the world, and thus we can regularize it using a locally planar model. This implicitly regularizes the flow in a more geometrically

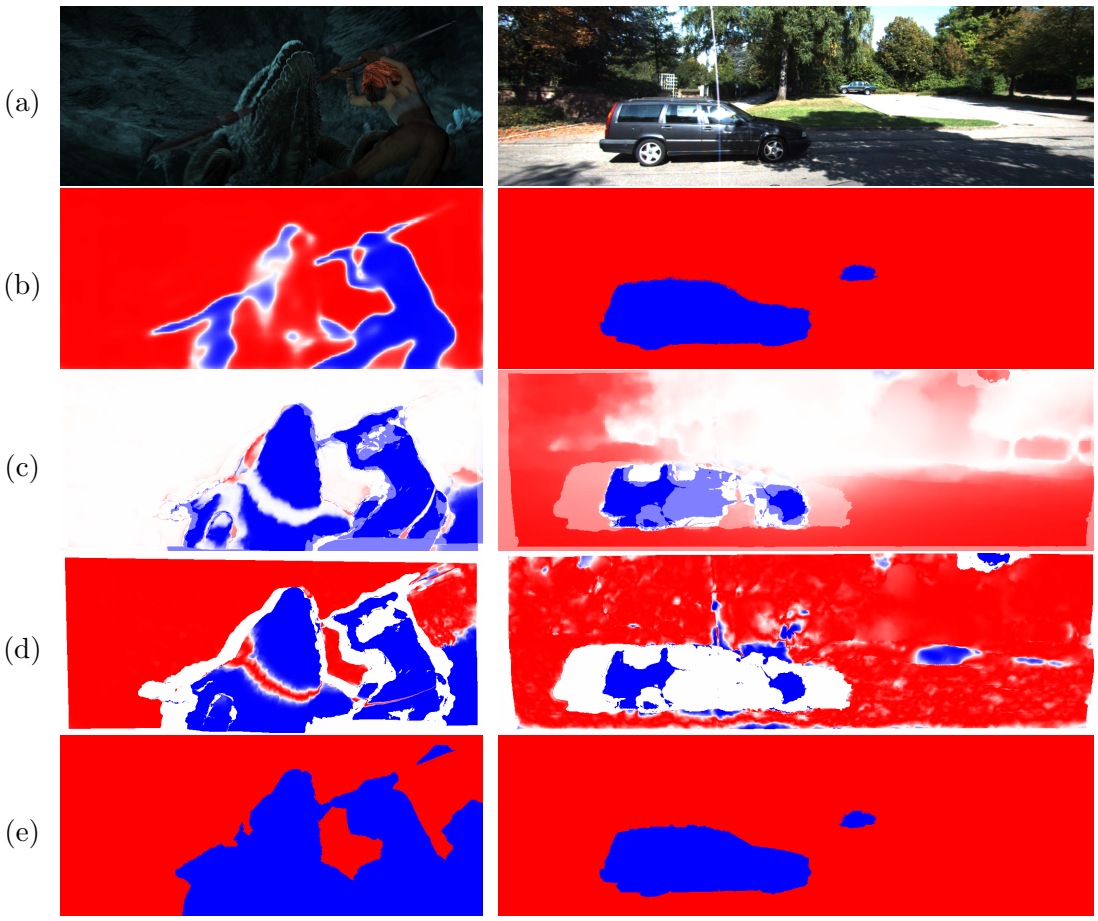


Figure 5.8: Results of rigidity estimation on examples from the test sets of MPI-Sintel and KITTI-2015. From an image (a), we estimate a segmentation into the rigid scene and moving objects based on the semantic segmentation (b) and combine it with the direction-based segmentation (c) and the structure-based segmentation (d) to obtain the final estimate (e). Likely rigid regions are red, likely moving regions are blue.

meaningful way than traditional priors on the flow.

Under these considerations, the full model for the motion of the rigid parts of the scene is defined as $E(s, \boldsymbol{\theta}^+, \boldsymbol{\theta}^-) =$

$$\sum_{\mathbf{x}} \hat{r}(\mathbf{x}) \left(E_d + \lambda_c E_c + \lambda_{1st} E_{1st} + \lambda_{2nd} E_{2nd}\right). \tag{5.33}$$

E_d is the photometric error, modulated by the estimated visibilities in forward and backward directions,

$$E_d = v^+(\mathbf{x}) \rho \left(\| I_a^+ \left(\boldsymbol{w} \left(\mathbf{x}, s(\mathbf{x}), \boldsymbol{\theta}^+ \right) \right) - I_a(\mathbf{x}) \| \right)$$

$$+ v^{-}(\mathbf{x})\rho\left(\|I_a^{-}\left(\boldsymbol{w}\left(\mathbf{x}, s(\mathbf{x}), \boldsymbol{\theta}^{-}\right)\right) - I_a(\mathbf{x})\|\right), \tag{5.34}$$

where I_a^{-}, I_a, I_a^{+} are augmented versions of I^{-}, I, I^{+}, *i.e.* stacked images containing the respective gray scale images and gradients,

$$I_a\left(\mathbf{x}\right) = \begin{pmatrix} \gamma I\left(\mathbf{x}\right) \\ \nabla_{x_1} I\left(\mathbf{x}\right) \\ \nabla_{x_2} I\left(\mathbf{x}\right) \end{pmatrix}, \tag{5.35}$$

and I_a^{-}, I_a^{+} set correspondingly. The weight γ is used to balance the influence of gray scale intensities and gradients; we empirically set it to $\gamma = 0.01$. The warping function $\boldsymbol{w}(\mathbf{x}, s(\mathbf{x}), \boldsymbol{\theta})$ defines the correspondence of $\mathbf{x}$ according to the structure s and the P+P parameters $\boldsymbol{\theta}$,

$$\boldsymbol{w}(\mathbf{x}, s(\mathbf{x}), \boldsymbol{\theta}) = \left\langle \boldsymbol{H}^{-1}\left(\mathbf{x} + \frac{s(\mathbf{x})b}{s(\mathbf{x})b - 1}\left(\mathbf{e} - \mathbf{x}\right)\right)_h \right\rangle. \tag{5.36}$$

The consistency term E_c encourages similarity between s and $s^{\{+,-\}}$.

$$E_c = v^{+}(\mathbf{x})\rho_c\left(s(\mathbf{x}) - s^{+}(\mathbf{x})\right) + v^{-}(\mathbf{x})\rho_c\left(s(\mathbf{x}) - s^{-}(\mathbf{x})\right). \tag{5.37}$$

Here, we use the Charbonnier function as the robust penalty ρ_c. This ensures that all values for $s(\mathbf{x})$ that lie inside the interval $[s^{-}(\mathbf{x}), s^{+}(\mathbf{x})]$ are penalized equally, and the energy only increases for values outside the interval.

The locally-planar regularization uses a 2nd order prior,

$$\begin{aligned} E_{2nd} = {} & m_{x_1}\rho\left(\nabla_{x_1x_1} s(\mathbf{x})\right) \\ & + m_{x_1}m_{x_2}\rho\left(\nabla_{x_1x_2} s(\mathbf{x})\right) + m_{x_2}\rho\left(\nabla_{x_2x_2} s(\mathbf{x})\right), \end{aligned} \tag{5.38}$$

with, using a slight abuse of notation, $\nabla_{x_1x_1}, \nabla_{x_1x_2}, \nabla_{x_2x_2}$ as the second derivative operators and again the modulation terms from [202]

$$\begin{aligned} m_{x_1} = {} & m\left(I(\mathbf{x}), I(\mathbf{x} + (1,0)^{\top})\right) \\ m_{x_2} = {} & m\left(I(\mathbf{x}), I(\mathbf{x} + (0,1)^{\top})\right) \\ m\left(I(\mathbf{x}), I(\mathbf{y})\right) = {} & \exp\left(-\frac{\left(I\left(\mathbf{x}\right) - I\left(\mathbf{y}\right)\right)^2}{2\mathbb{E}\left[\|\nabla I_1\|_2\right]}\right). \end{aligned} \tag{5.39}$$

Since the second order prior by itself is highly sensitive to noise, we add a first order prior

$$E_{1st} = m_{x_1}\rho\left(\nabla_{x_1} s(\mathbf{x})\right) + m_{x_2}\rho\left(\nabla_{x_2} s(\mathbf{x})\right), \tag{5.40}$$

where $\nabla_{x_1}, \nabla_{x_2}$ are the first derivative operators in the horizontal and vertical

direction respectively.

Optimization. To minimize the energy (5.33) we employ an iterative scheme, and alternate between optimizing for s with $\boldsymbol{\theta}^{\{+,-\}}$ fixed, and for $\boldsymbol{\theta}^{\{+,-\}}$ with s fixed. When optimizing s, we use a standard warping-based variational optimization [50] with 1 inner and 5 outer iterations and no downscaling. To optimize for $\boldsymbol{\theta}$, we first optimize for $\boldsymbol{H}, b$ using L-BFGS and then recompute $\mathbf{e}$ as described in Sec. 5.3. We use two iterations, since we found that more do not decrease the error significantly. This yields the final estimates $\bar{s}, \bar{\boldsymbol{\theta}}^{+}, \bar{\boldsymbol{\theta}}^{-}$ for the structure and the P+P parameters.

Due to the non-convex nature of (5.33), a global optimum is not guaranteed. However, in practice we found that our initializations are close to a good optimum, and hence our optimization procedure works well.

Final flow estimation. Finally, we convert the estimated structure $\bar{A}$ into an optical flow field

$$\mathbf{u_s}(\mathbf{x}) = \mathbf{w}\left(\mathbf{x}, \bar{s}, \bar{\boldsymbol{\theta}}^{+}\right) - \mathbf{x}. \tag{5.41}$$

In the moving regions, we use the initial forward flow $\mathbf{u_0^+}$, and compose the full flow field as

$$\mathbf{u}\left(\mathbf{x}\right) = \hat{r}(\mathbf{x})\mathbf{u_s}(\mathbf{x}) + \left(1 - \hat{r}(\mathbf{x})\right)\mathbf{u_0^+}(\mathbf{x}). \tag{5.42}$$

5.6 Experiments

5.6.1 Ablation study

We show how different subcomponents of our algorithm impact the end result. The results in this section were obtained using a reduced version of the MPI-Sintel training set containing every 4th frame, and only the final pass. To assess the impact in different regions of the frame, we provide the errors both on the full frames and only in the ground truth rigid regions.

For the ablation study, we successively switch on four steps: *occlusion reasoning*, *coplanarity refinement*, *nonlinear initialization of* b_0^-, and *spatial priors*.

- **Occlusion reasoning** refers to the estimation of the visibility maps v^+, v^- using the forward-backward consistency, as described in Sec. 5.3.2. If this step is switched off, we set $v^- = v^+ = 1$ everywhere, and therefore do not explicitly exclude occluded pixels from subsequent computations.

- **Coplanarity refinement** refers to the second part of the initial alignment, described in Sec. 5.3.1. This step refines the initial homographies $\bar{\mathbf{H}}^-, \bar{\mathbf{H}}^+$ to ensure that after registration all residual flow vectors meet in the two epipoles $\mathbf{e}^-, \mathbf{e}^+$. The optimization yields $\mathbf{H_0}^-, \mathbf{H_0}^+$. If this step is switched off, we simply set $\mathbf{H_0}^+ = \tilde{\mathbf{H}}^+, \mathbf{H_0}^- = \tilde{\mathbf{H}}^-$.

- **Nonlinear initialization of** b_0^- refers to initializing b_0^- using Eq. (5.12), *i.e.* choosing b_0^- so that the resulting backward s_0^- structure is as similar as possible under a robust error norm to the initial forward structure s_0^+. Note that, without loss of generality, b_0^+ is always chosen so that the MAD of s_0^+ is 1.

  If this step is switched off, we use Eq. (5.11) to express s_0^- as a function of b_0^-. Equating s_0^+ and s_0^- produces an estimate of b_0^- per pixel, of which we take the median to arrive at the global estimate of b_0^-.

$$b_0^- = \underset{\mathbf{x}}{\texttt{median}} \frac{1}{s^+(\mathbf{x})} \frac{h^-(\mathbf{x})}{\|\mathbf{e}^- - \mathbf{x}\| - h^-(\mathbf{x})} \tag{5.43}$$

- **Spatial priors** refer to the 1st- and 2nd-order spatial smoothness regularizers in our objective function (17). To disable those, we set $\lambda_{1st} = \lambda_{2nd} = 0$.

Table 5.1: Errors when successively switching on parts of the algorithm

	occlusion reasoning	coplanarity refinement	nonlinear b^- initialization	spatial priors	EPE (rigid)	EPE (all)
Baseline	☐	☐	☐	☐	1.859	3.798
+occlusions	■	☐	☐	☐	1.733	3.705
+coplanarity	■	■	☐	☐	1.695	3.671
+nonlin-init	■	■	■	☐	1.619	3.628
Full	■	■	■	■	1.602	3.614

Table 5.1 shows the improvement when successively switching on more parts of the algorithm. The occlusion reasoning has the largest positive impact on the error, since it allows the algorithm to properly merge the flow in both directions from the reference frame. Following this, the most important parts are the nonlinear initialization and ensuring the coplanarity. The spatial priors serve mostly to remove flow noise near boundaries. This improves the result visually, but has a small numerical impact.

Table 5.2 shows the impact when turning off individual components, but leaving all others intact. Again, we can observe that disabling the occlusion reasoning has the largest negative impact, followed by the nonlinear initialization and ensuring the coplanarity.

Table 5.3 shows the impact of the different terms of the variational refinement (Eq. (5.33)). The cases are as described above. In addition, Table 5.3 includes a complete omission of the variational refinement (*no-opt*), which uses only the

Table 5.2: Errors when disabling individual parts of the algorithm.

	occlusion reasoning	coplanarity refinement	nonlinear b^- initialization	spatial priors	EPE (rigid)	EPE (all)
no-occlusions	□	■	■	■	1.809	3.759
no-coplanarity	■	□	■	■	1.642	3.645
no-nonlin-init	■	■	□	■	1.677	3.656
no-spatial-priors	■	■	■	□	1.619	3.628
Full	■	■	■	■	**1.602**	**3.614**

Table 5.3: Influence of regularization terms.

	Data term	1st order regularization	2nd order regularization	EPE (rigid)	EPE (all)
No-opt	□	□	□	1.6124	3.6214
No-spatial-priors	■	□	□	1.6194	3.6285
No-1st	■	□	■	1.6025	**3.6138**
No-2nd	■	■	□	1.6183	3.6274
Full	■	■	■	**1.6024**	**3.6138**

initial structure estimate as described by Eq. (5.15), and selective disabling of the individual regularizers ($\lambda_{1st} = 0$ for *no-1st* and $\lambda_{2nd} = 0$ for *no-2nd*).

When using the data term only (*no-spatial-priors*), the error is higher than when not using any optimization. Due to effects in the Sintel final pass such as motion blur, fog, vignetting etc. this is to be expected. Using the 2nd order regularization improves the results; interestingly, however, the impact of the 1st order regularization is negligible.

5.6.2 Qualitative results

This section shows visual results on the clean (Fig. 5.9) and final passes of Sintel (Fig. 5.10) and KITTI (Fig. 5.11). In all figures, we show the three unaligned input frames as overlays, the ground truth optical flow, the estimated segmentation, where red corresponds to the rigid scene and blue corresponds to moving regions, the structure with the estimated moving regions masked in purple, the final optical flow field and the change compared to the initial optical flow. In this comparison, we display in green regions where MR-Flow improves upon the initial flow method, while in red regions, MR-Flow makes the results worse.

Note that for KITTI the ground truth flow is only given for a sparse sampling of pixels and does not include some objects such as moving people. Hence, the flow

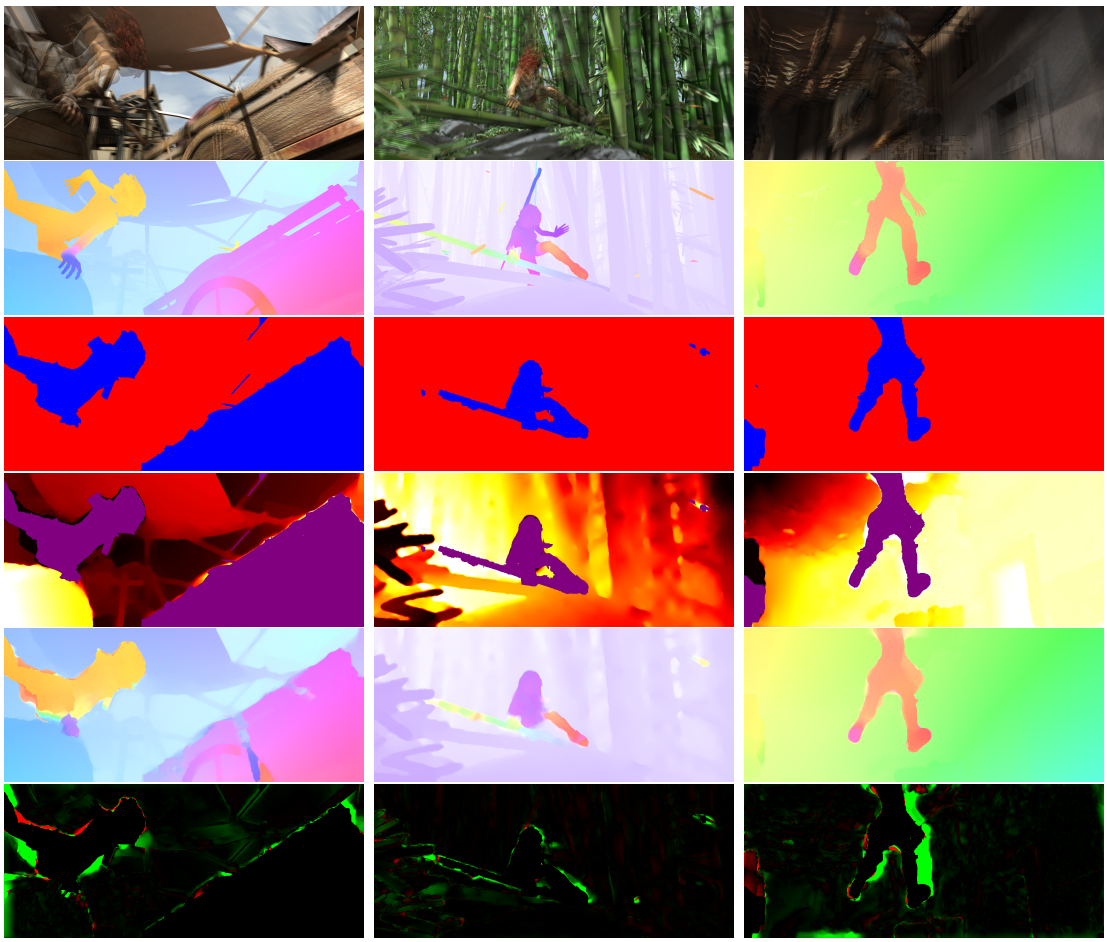
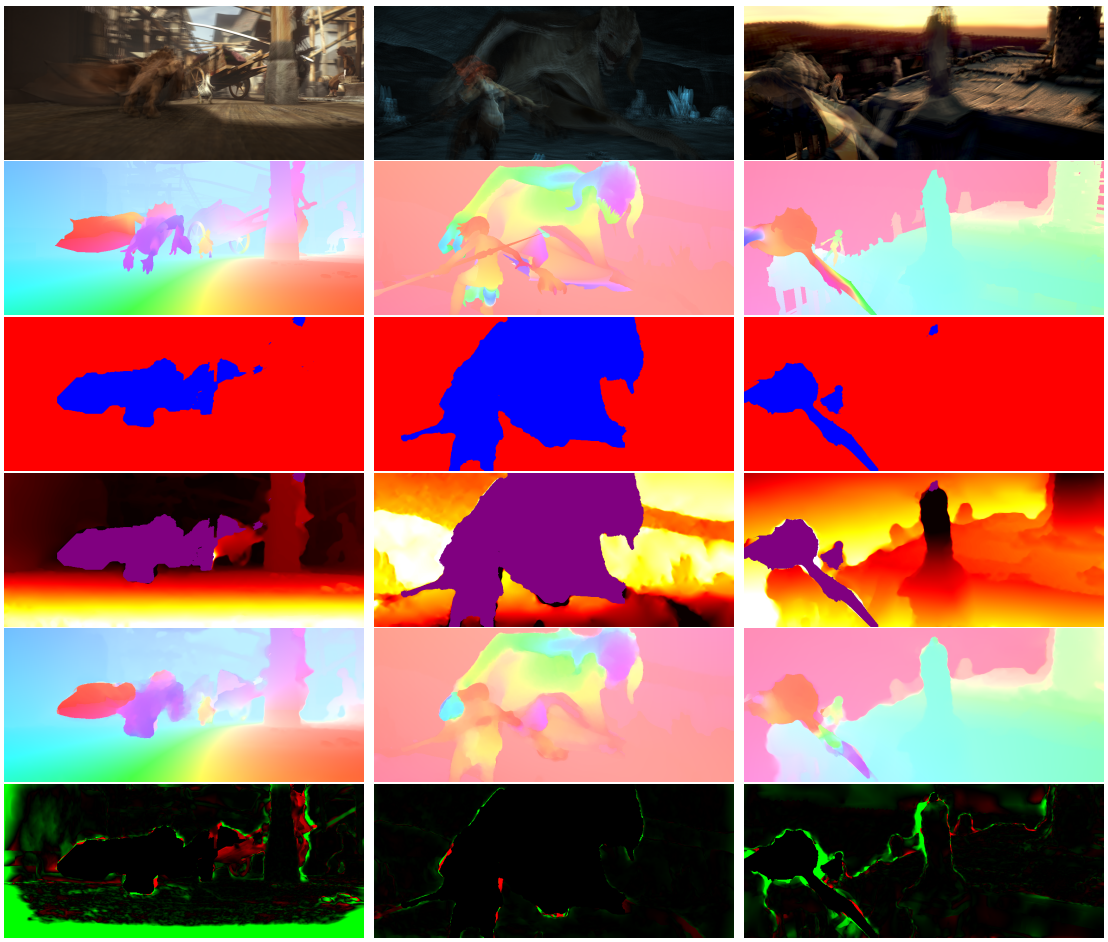


Figure 5.9: Results on MPI-Sintel, clean pass. From top to bottom: Input images; Ground truth optical flow; estimated segmentation; estimated depth structure with moving objects masked in purple; estimated optical flow; comparison to DiscreteFlow [174].

estimated using our method (fifth row) looks very different from the ground truth flow (second row). However, this is merely an artifact of the visualization of the flow which shows unlabeled pixels in white.

5.6.3 Quantitative results

To quantify our method, we evaluate on the MPI-Sintel and KITTI-2015 flow benchmarks. The parameters are chosen to minimize errors on the training sets, and are set to $\{\sigma_d, \sigma_s, \lambda_{r,c}, \lambda_{r,p}, \lambda_c, \lambda_{1st}, \lambda_{2nd}\} = \{0.75, 2.5, 0.1, 1.1, 0, 0.1, 5e3\}$ for Sintel and $\{1.0, 0.25, 0.5, 1.1, 0.01, 1, 5e4\}$ for KITTI. As dense flow initialization and fallback, we use the publicly available DiscreteFlow [174]. Table 5.4 shows the errors for



(a) Sintel, clean (b) Sintel, final (c) KITTI

Figure 5.10: Results on MPI-Sintel, final pass.

our method, our initialization (DiscreteFlow), and for top performing methods on MPI-Sintel (FlowFields+) [18] and KITTI-2015 (SDF) [17]. Both evaluate only on one dataset; in contrast, our method achieves high accuracy on both datasets. Figures 5.9, 5.10, and 5.11 visualize results and a comparison with our initialization.

On **MPI-Sintel**, our method gives state-of-the-art results, ranking first on the clean pass and third behind DCFlow [281] and FlowFieldsCNN[19] on final. In particular, the structure estimation gives flow in occluded regions, producing the *lowest errors in the unmatched regions* of any published or unpublished work. On a 2.2 GHz i7 CPU, our method takes on average 2 minutes per triplet of frames without the initial flow computation, 74s for the initialization and rigidity estimation, and 46s for the optimization.

In **KITTI-2015** the scenes are simpler and contain only automotive situations; however, the images suffer from artifacts such as noise and overexposures. Among

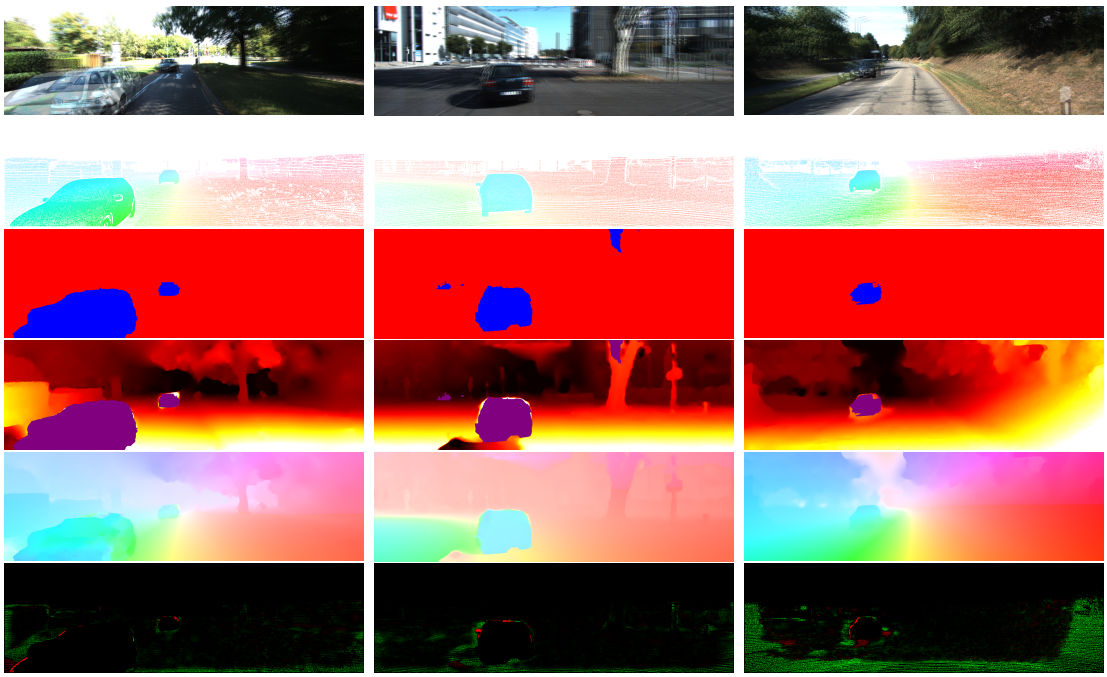


Figure 5.11: Results on KITTI.

published monocular methods, MR-Flow is third after Flownet2 [120] and SDF [17], the later of which is designed for automotive scenarios and not tested on Sintel.

The fact that different competing methods outperform MR-Flow on different datasets indicate that most top performing methods to date are tuned to either Sintel or KITTI, and do not perform very well on the respective other dataset. Table 5.5 supports this finding; here, we show the methods that outperform MR-Flow on any of the datasets, and give the rank on the test set across *all* datasets[8]. As can be seen, the top performing methods do not generalize well. We believe that this is because these methods use learning as a core component, and are hence prone to overfitting. MR-Flow, on the other hand, is based on geometric reasoning, and geometric constraints hold, regardless of the specific dataset.

5.6.4 Failure cases

While our algorithm gives usually good result, there are still some cases where it fails. Here, we define a failure case as one in which the flow computed by our algorithm is worse than the initial flow [174]. This section shows such failure cases, and gives on example for each the clean and final passes of Sintel and KITTI. All examples are among the worst overall in the respective training sets. In our training

[8]These results were obtained at the time of writing, in October 2017, and include published, monocular methods.

Table 5.4: Errors on Sintel (EPE) and KITTI (%incorrect).

	Sintel				KITTI 2015	
	clean		final			
	Train	Test	Train	Test	Train	Test
DF [174]	1.96	3.57	3.80	6.08	23.09%	21.57%
FF+ [18]	-	3.10	-	5.71	-	-
SDF [17]	-	-	-	-	**12.14%**	**11.01%**
MR-Flow	**1.83**	**2.53**	**3.59**	**5.38**	14.09%	12.19%

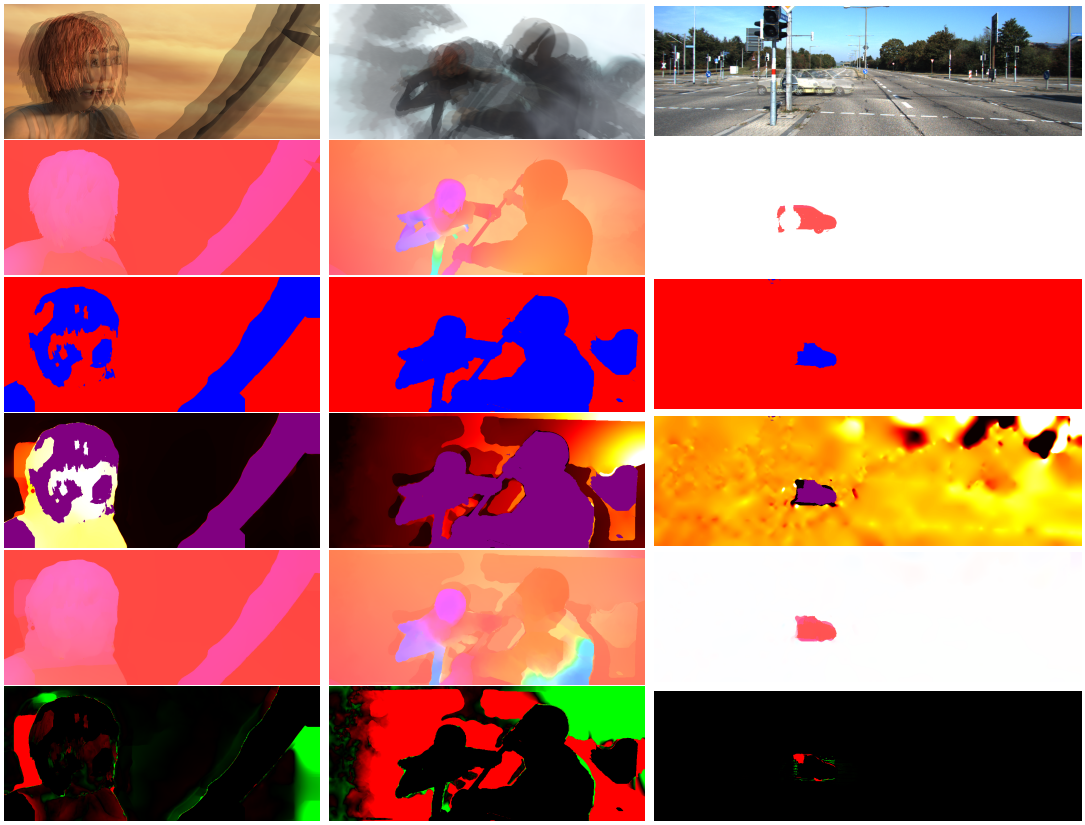
Table 5.5: Ranks of top performing methods on the test sets of Sintel and KITTI. While MR-Flow is the best only on the clean pass of Sintel, it achieves good performance across all datasets, unlike all other methods, as reflected in the average rank.

	Sintel (clean)	Sintel (final)	KITTI 2015	*Average*
DCFlow [281]	6	**1**	4	*3.7*
FlowFieldsCNN [19]	14	2	10	*8.7*
FlowNet2 [120]	19	6	**1**	*8.7*
SDF [17]	n/a	n/a	2	*n/a*
MR-Flow	**1**	3	3	***2.3***

set, we observe two primary sources of error, segmentation failures and alignment failures.

Figures 5.12c and Figure 5.12a show examples for the first type of error, segmentation failures. In these cases, moving regions are mistaken as parts of the static background, such as the car in Fig. 5.12c or the girl's head in Fig. 5.12a. These failures occur if the CNN does not pick up a region strongly enough and if, at the same time, the motion of the object is consistent with the motion of the rigid scene. In Fig. 5.12c, the CNN picks up only the frontal part of the car. Since in this example the camera is not moving, the focus of expansion is mistakenly determined by the few parts of the frame that move (*i.e.* the car), and the motion-based segmentation cannot correct the mistake made by the CNN.

In Fig. 5.12a, the camera pans to the left while at the same time the head moves to the right. Both directions are close to parallel, and hence the parallax of the head points towards the center of expansion. Therefore, the head is considered to be static. Note how in this case the estimated flow has a very similar hue to the ground truth flow, even in most of the regions that are misclassified. This



(a) Sintel, clean.
EPE initial: 0.93 px.
EPE MR-Flow: 1.59 px.

(b) Sintel, final.
EPE initial: 11.17 px.
EPE MR-Flow: 12.21 px.

(c) KITTI.
Perc. wrong initial: 3.62%.
Perc. wrong MR-Flow: 4.48%.

Figure 5.12: Examples for failure cases. From top to bottom: Input image overlay; ground truth optical flow; estimated rigidity segmentation; estimated structure; estimated optical flow; comparison to initial flow.

confirms that the direction of the flow is approximately consistent with the motion of the rigid regions of the scene. However, the head motion still slightly violates the P+P constraints. Since the misclassification results in the wrong constraints being applied to the region of the head, our method increases the error over the initialization.

Figure 5.12b shows the second type of error, a failure to align the images. As can be seen in the top row of Fig. 5.12b, the background in this sequence contains heavy motion blur and a slight vignetting. Together, these two effects cause a high uncertainty of the initial optical flow in the background regions, which in turn causes our initial alignment procedure to fail.

5.7 Conclusion

The focus of this chapter was to extend the layered scene approximation that we described in the previous two chapters. Instead of using a set of general layers without any semantic meaning, here we have demonstrated an optical flow method that segments the scene and improves accuracy by exploiting strong constraints in the rigid parts of the scene. Two main contributions made this possible. First, we note that moving surfaces always belong to objects that can be detected using semantic segmentation methods. This provides a useful initialization and the basis for a segmentation of a natural, moving scene into the rigid background and moving objects using both semantic and motion information. For the moving objects, we use an existing general optical flow method to compute the motion, while in the rigid background the motion is strongly constrained by the 3D structure of the world.

The second contribution is to show how we can use the Plane-Plus-Parallax methodology to reason about this 3D structure using only three frames, a scenario in which classical SfM methods often become unstable. Apart from constraining the solution space, expressing the flow of the rigid scene via its three-dimensional structure gives us two main advantages. First, it allows us better integrate information from more than two frames, which is crucial to compute the optical flow for regions that become occluded or leave the visible frame. Second, it is now possible to regularize the depth structure using a piecewise-planar model as opposed to regularizing the projected motion on the image plane. This better capture the actual structure of the scene, as opposed to the structure of the projected motion.

Our algorithm, MR-Flow, uses the appropriate constraints in different regions of the image and produces accurate flow in challenging situations and competitive results on Sintel and KITTI. Most importantly, across datasets MR-Flow generalizes much better than comparable state-of-the-art algorithms. Due to their strong focus on learning, these algorithms are implicitly tuned for specific datasets. MR-Flow, on the other hand, is based on geometric reasoning, which can be applied to any dataset coming from the natural world, irrespective of the specific content.

Chapter 6

Conclusion

Optical flow has long been an active area of research in computer vision. However, most methods for the computation of optical flow focus on motion in the image plane. They only determine "where a pixel moves", and thus do not take into account the image formation process, the three-dimensional geometry of the scene, or other information that heavily constrains the possible optical flow fields – in short, these methods compute optical flow without considering how this optical flow might have arisen in the first place. Various problems with optical flow methods are a direct result of this. Examples include the severe degradation of accuracy in case of motion blur, the difficulty of resolving occluded pixels, and the inability to consider more than two frames at a time and to properly integrate information from multiple frames.

Layered models of the scene are an attempt to overcome these difficulties. In these models, the scene is approximated as a set of overlapping layers which move independently from each other, with the implicit assumption that the subdivision into layers roughly corresponds to a segmentation of the scene into objects. Layered models are therefore well suited to capture effects at object boundaries such as occlusions and naturally yield anisotropic regularization, smoothing the flow only among pixels belonging to the same object (*i.e.* layer).

Simple layered methods, however, still suffer from a number of issues. First, similar to existing flow methods, they do not work well in the presence of motion blur, since in this case the boundaries between objects become smeared out, and simple color-based layer assignment hence becomes difficult. Second, existing layer methods represent the optical flow of each layer using a full, dense optical flow field. While this retains maximum expressiveness, it does not make use of the fact that in the ideal case the motion *within* a layer corresponds to the motion of a coherent surface of the world, and is hence highly constrained. Methods neglecting this have to solve for full flow fields many times during the computation, making them computationally inefficient. Third, due to the basic assumption that a scene can be composed of overlapping patches, layered methods often have problems with complex geometries. Consider a forest - in such a scene, it is not immediately clear what the "right" layer segmentation would be. Should each tree constitute

a separate layer? This would lead to a huge number of layers, and hence a combinatorial explosion of possible depth orderings among them. Approximating the whole forest using a low number of layers, on the other hand, would not do justice to the complex three-dimensional structure of the scene, which contains numerous depth discontinuities and self-occlusions. Taken together, these issues contributed to layered methods being not particularly popular in practice.

This thesis lays out several approaches addressing the issues mentioned above. First, Chapter 3 shows how the effect of motion blur can be conveniently addressed in a layered framework. The two primary problems when addressing motion blur are that a dynamic scenes contains blur originating from multiple motions (for example, the camera motion and individual object motion), and that the blur at object boundaries is a mixture of the blurs of the occluding and the occluded surfaces. Within a layered framework, both problems can be addressed, since layers cleanly separate the scene into individual overlapping objects. The total motion blur can thus be formulated as a overlapping set of individual motion blurs, defined on each layer, allowing us to compute the blur for each layer separately. This enables us to treat motion blur in multiple directions at the same time and provides a clean way to model the interplay of motion blurs at object boundaries. To this end, we propose a generative model of a layered scene under the influence of motion blur, and parameterize the scene by a set of overlapping layers and their motion. Hence, we can use our model to solve for both the sharp appearances of individual layers in a scene, a sharp segmentation into layers, and the optical flow.

Chapter 4 then turns to the issue of the structure of optical flow, in particular *within* a layer. This chapter shows that a low-dimensional model of optical flow can be learned from a large number of training data using simple Principle Component Analysis. This model encodes global properties of the optical flow field, and can therefore be used to effectively interpolate between sparse feature matches, yielding a full optical flow field with high computational efficiency. While the resulting optical flow field is blurry and lacks high frequency information, it can be integrated into a layered model. This combines the best of both worlds, in that the layered model is used to model important discontinuities of the optical flow field, while the fast, low-dimensional model is used to represent the optical flow *within* each layer, where the flow should ideally be smooth and not exhibit any discontinuities. The result is a fast, layered model that simultaneously computes optical flow and a segmentation of the scene into layers.

Lastly, Chapter 5 addresses the difficulty of modeling complex geometries with layers. To this end, it proposes an alternative formulation to layers, namely, not to segment the scene into a set of overlapping, qualitatively equal segments, but to use simple semantic categories to distinguish between moving objects and the static scene. While we do not consider full semantics (to treat the motion computation of *e.g.* a horse differently from that of a car), the distinction between the rigid scene and moving objects nevertheless can help us in three important ways. First, in the

static scene the optical flow at a given point is only determined by the distance of this point to the camera and the motion of the camera. The problem of estimating the flow hence reduces to solving for a single unknown number per pixel in addition to a low number of global camera parameters, *i.e.* only about half of the $2 \times W \times H$ unknowns of a generic optical flow problem. Here, we show how a Plane+Parallax formulation allows us to reason about the camera motion and the depth structure, achieving high accuracies even in scenes that would be challenging for geometric reasoning based on standard epipolar geometry. Second, by definition the static scene does not move, and hence has to have the same geometric structure at different points in time. This provides a clean way to accumulate information from more than two frames, in contrast to previous approaches which usually use heuristics such as constant velocity or constant acceleration, which do not work well in practice. Third, since we treat the optical flow as implicitly parameterized by the depth structure of the scene, we can impose a piecewise-planar prior directly on this depth structure instead of on the optical flow, which is a more direct way of introducing additional structure and assumptions about the world. The segmentation of a scene into moving objects and static parts hence allows us to reason about general scenes and still make use of these three insights – we can impose them *where it is appropriate*, and in the remaining regions (*i.e.* the moving regions) compute optical flow using a generic optical flow method. The resulting algorithm computes state-of-the-art results on standard benchmarks, and generalizes significantly better than learning based optical flow methods, which overfit to specific benchmarks.

6.1 Limitations and potential extensions

The goal of this thesis is to show that combining layered methods with more explicit reasoning about the scene can have a substantial impact on the quality of the computed optical flow. As such, the individual results should be considered as feasibility studies, and not complete algorithms suitable and robust enough to process arbitrary inputs. Here, we will briefly restate the current limitations of the work presented in this thesis, and give a few pointers on how to overcome those.

Chapter 3 shows how modeling a motion blurred video using a layered scene model can help with both deblurring and optical flow estimation. We make a number of simplifying assumptions. First, we model the optical flow of each layer as affine, which currently restricts our method to scenes with suitable motion. While this provides a good approximation for a number of scenarios, it fails for complex deformations such as a walking person. However, this is just an implementation restriction. The actual model, presented in Section 3.2, uses a formulation based on the filter flow framework [213]. Since filter flow can also model dense, smoothly varying optical flow, our formulation is just as valid for those general flow scenarios. Furthermore, our model assumes a two-layer scene. While such a model is frequently

used in comparable work [189, 233], it is again somewhat restrictive. Extending layered methods to a larger number of layers is possible using largely the same underlying theory [231], but comes with additional computational cost. Lastly, our model assumes constant layer appearance over the sequence, as well as a constant layer segmentation. This causes problems in case of changing appearances, such as the highlight on the windshield in Fig. 3.14, where a changing appearance results in ringing artifacts. One possible remedy for this would be to estimate a single, latent appearance and segmentation for the whole sequence, but allow small deviations at each frame. Finally, we only consider a classical shutter, and do not address the issues arising with rolling shutters. Modeling such shutters is possible in our framework but remains future work (cf. [99]). Beyond motion blur, images contain further artifacts such as focal blur, discretization, and camera noise. These have been modeled before and could be incorporated into our model, enabling joint motion estimation, denoising, and super-resolution (cf. [21]).

In Chapter 4, we turn to the problem of parameterization of the flow on each layer, and show that a linear basis learned using PCA can be used for this purpose, resulting in a fast, layered optical flow algorithm. The main issues with our approach are threefold. First, using a PCA implicitly assumes that the underlying data lives on a linear subspace and is Gaussian distributed. While good approximations, both assumptions do not hold exactly in case of general optical flow. The first extension to this work would hence be to estimate a subspace for full-frame optical flow using more advanced, non-linear subspace estimation methods, while retaining as much of the computational efficiency of a PCA-based method as possible. Second, the MRF we use to estimate the segmentation of the scene into layers is fairly simplistic and does not take non-local connections into account. This could be solved by replacing our MRF with a method such as [144], or by integrating higher level scene classification and segmentation [216]. Third, our segmentation as well as our low-dimensional flow representation are not temporally consistent. In reality, both the motion characteristics (*i.e.* the low-dimensional representation of the motion of a layer) as well as the support of a layer only change slowly over time due to physical constraints. Again, one could address this issue by either estimating a latent representation over the whole scene and penalizing deviations from this; an alternative would be to condition the estimates of the motion estimation and the layer segmentation on the previous estimates, effectively encouraging both to vary slowly.

In Chapter 5, we extend the classical layered approach and propose to treat the regions belonging to the static parts of the scene separately; this allows us to use strong constraints on the motion where appropriate and to better accumulate information across multiple frames. Currently, the main failure cases of the proposed method are a wrong segmentation into the static scene and moving objects, and wrong motion estimation of moving objects due to bad initialization. Both could be addressed by a more integrated formulation of our approach. Currently, our al-

gorithm contains several modules (*e.g.* segmentation and the estimation of motion in the static regions), which work largely independently from each other. In the future, these components should be better integrated, so that the algorithm would simultaneously reason about (a) the segmentation, (b) the motion in the static parts of the scene, and (c), using a traditional layered flow method, about the motion of the moving objects. Integrating these components into a joint objective function would lead to less reliance on the initialization, thereby reducing potential negative impact of a bad initialization. Still, on the clean pass of MPI-Sintel, our algorithm outperforms all others, while the performance slightly degrades in the final pass. This suggests that our simple photometric data term is not sufficiently robust; using a learned data term such as the one used in [281] would help here. Furthermore, semantics could be used to constrain the motion of moving objects [216]. Currently, we estimate the semantic class of an object, but do not fully exploit this information, *e.g.* by constraining the motion of a car differently from that of a person. Also, similar to above, beyond the three frame triplet the structure estimation and the segmentation are not forced to be temporally constrained. Again, the segmentation should change only slowly in time, while the structure should be constant. Lastly, if calibration data of the camera is available, it would be possible to upgrade the structure to a metric depth reconstruction, allowing higher level reasoning about the ground plane, gravity, and the resulting constraints on moving objects.

6.2 Future work

The suggestions mentioned in the previous section constitute steps to push each of the projects described in this thesis further towards robustness and general applicability. What, however, are the larger implications of the work shown here, both for the field of optical flow computation in particular and for the broader field of computer vision? There are two main directions: the combination of models and knowledge on one hand and the integration into learning frameworks on the other.

6.2.1 Combined models

This thesis presented a number of ways in which constraints from the real world can be integrated into the estimation of motion, especially in a layered framework. All these ways, however, have been largely distinct from each other. As an example, while we address motion blur in Chapter 3, we do not integrate it into Chapters 4 and 5, since these chapters focus on other properties of optical flow in layered scenes. To solve the motion estimation problem, however, it would be necessary to integrate all these individual components into one joint framework. Such a model would reason about the image formation process, the geometry of the scene, a proper segmentation of the scene into meaningful units, including their semantic

meaning, and the right motion representation for each segment.

Attempting to solve for all these unknown would pose two main questions. First, since a stream of images is everything that is observed in a video, this would be a highly ill-posed problem, in that we have much more variables than data to estimate these variables from. A large part of the reasoning would hence have to be shifted to a previous learning process, which would teach the system about the world, its parts, and their dynamics. Having learned about all this, the system could infer the parts of the scene, their motion, and their geometry from just a video, similar to how we humans are able to interpret a scene and its 3D layout just from watching a 2D video.

The second issue would be computational efficiency. In this thesis, we commonly use iterative algorithms, either in the form of gradient descent as in Chapter 3 and the variational refinement in Chapter 5, or IRLS in Chapter 4. Those algorithms, however, are not among the most computationally efficient, and become slower as the model complexity increases, as seen in Chapter 3. Thus, for an all-encompassing model to be of any practical relevance, a focus needs to be on how such a complicated model can be optimized efficiently.

6.2.2 Integration into learning systems

The second implication is the integration of models into learning-based systems. A convenient property of algorithms based on convolutional neural networks is that they usually achieve top performance when they are trained in a so-called "end-to-end" fashion, *i.e.* they learn to directly map some inputs (such as images) to one or multiple outputs (such as optical flow), without explicit requirements on intermediate representations. Such requirements would be defined by a user, and might hence reflect the intuition of this user more than the actual properties of the world. End-to-end based methods, in contrast, learn purely from the given training data. The flipside of this is that such algorithms are extremely prone to overfitting, since most datasets used for training contain more or less subtle biases [246]. We saw the impact of this when comparing the results of MR-Flow (described in Chapter 5) to other, predominantly CNN-based methods. While the latter often outperform MR-Flow on one of the two main optical flow benchmarks, they all perform significantly worse on the other, showing strong overfitting to a specific dataset and fairly bad generalization capabilities. The geometry-based MR-Flow, however, generalizes to different datasets and settings.

Furthermore, vision systems for many different applications do share common representations and intermediate steps; it is reasonable to assume that the detection of edges and corners is one, and some "filling-in" of unstructured regions is another. Systems that are based on end-to-end learning have to relearn these basic representations every time, making large parts of the training procedures redundant and ultimately a waste of time and energy.

The solution for both of these issues would be to take a step back, and re-consider end-to-end learning. Instead, one could explicitly integrate models and constraints into learning-based methods; in the context of optical flow computation, these models could for example capture occlusions, physics, and the constraints of geometry. Forcing an algorithm to reason along these lines, to take such representations and their effects into account, could make learning more efficient and reduce the impact of unwanted biases in datasets.

How could such constraints be integrated into learning-based algorithms? Three main possibilities come to mind. The first way is to use an information-processing lens, and view a computer vision algorithm as a pipeline of steps, each of which transform one representation into another [167]. Each of these transformation steps can now either take the form of a trained CNN, or of an explicit geometric processing step. This approach is often used in stereo [139] or 3D reconstruction algorithms [138], since reprojecting one image to another camera using an estimated 3D geometry is a fairly easy step, yet encodes a lot of useful information about the geometric relationship between cameras.

A second, largely unexplored way is to integrate geometry into the architecture itself, for example by forcing an intermediate representation to be decodable to the camera motion or 3D geometry of a scene [85, 292], or by forcing a bottleneck to correspond to a low-resolution optical flow field [76].

The third way, which has not been explored as of yet, is to constrain the learning process itself. In a CNN, the learning involves backpropagating a loss through the network and updating the network parameters to minimize this loss, thereby effectively taking a small step in the negative gradient direction. After computing this gradient, an algorithm could evaluate whether going along its direction would make the network return a physically possible result; if not, the gradient could be adjusted accordingly. At each learning iteration, this would ensure that the network always returns a physically plausible result; in effect, the network would be optimized along the manifold of physically plausible output values. While GANs [97] are a step into the direction of "learning on plausible manifolds", explicitly enforcing physics and geometry in this stage is still unexplored.

6.3 What is motion for?

To conclude this thesis, we shall briefly address the question "What is motion for?". Why do we invest time, energy, and thoughts to compute better optical flow? What is the point of motion estimation, and what are the advantages of a sequence of images bound together by temporal correspondences as compared to a sequence without any temporal linkage? In short, what do we hope to gain by computing motion?

I think the usefulness of motion lies primarily in three areas. The first is orien-

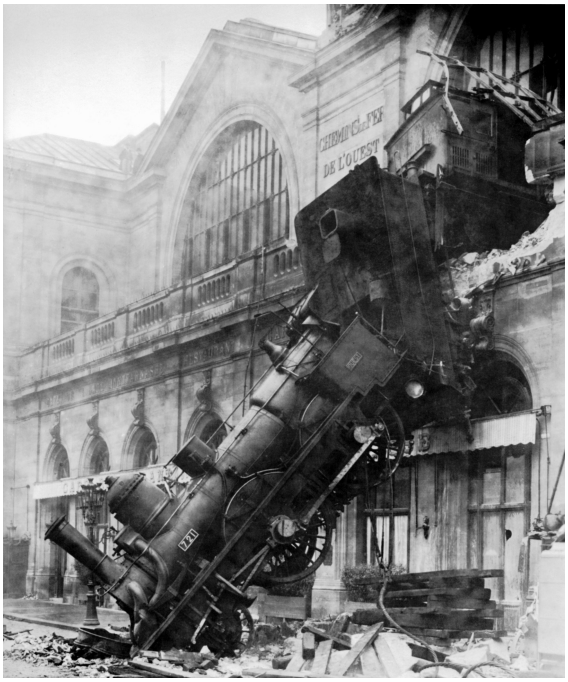


Figure 6.1: Train wreck. From this two-dimensional image, we can infer a tremendous amount of information about the scene. Furthermore, we can well imagine what happened immediately before this image was taken, as well as what will happen afterwards.

tation and the perception of space. As mentioned in Chapter 1, parallax resulting from motion information is an important cue for the three-dimensional shape and context of a scene and for the egomotion of the observer; even very primitive biological systems such as Drosophila utilize this cue, showing its prevalence and basic, low-level importance. Motion tells us about the scene, about our surroundings, and about our place in this world - in short, it helps answer the question *"Where am I?"*.

Second, it helps to answers the question *"What is happening?"*. We are not lonely observers of a three-dimensional world; instead, we interact with and change the environment, we observe how other agents do the same, and we collaborate with or work against such agents. We see a changing world, and furthermore, at least on a small scale, we perceive the reasons for these changes. Hence, motion helps us understand temporal evolution, subjects and objects of actions, capabilities of agents, and causal effects.

Lastly, we use both the perception of the environment and the perception of change to learn. Consider a single, two-dimensional photograph, such as the one shown in Fig. 6.1. The actual information contained in such an image is extremely limited; yet, we are able to determine (or at least make reasonable, educated guesses

about) a vast number of effects that would be hard to infer from the photo alone. The simplest, of course, is the 3D shape of the world in which the picture was taken; despite only seeing a printed 2D surface, we effortlessly perceive the geometry and depth relationships in the picture, which in itself is quite a remarkable feat. But our perception of a photograph goes further - we have a fairly reasonable sense of what must have happened before the photo was taken and what is going to happen immediately after. Furthermore, we can easily imagine the soundscape that accompanied the photo (*i.e.* what the photographer heard while taking the photo), and roughly how the scene looks like and what happens behind the photographer. We can call all these parts our *dynamic perception of an image*. Since it is static, the image itself contains only little evidence for this dynamic perception, if at all. Instead, from a single, static image, we are able to recall a wide range of matching dynamic stimuli, visually as well as in other modalities. Our understanding and interpretation of visual data is hence shaped to a large degree not by what we see in the moment, but by the years of experience we had, experience of us interacting with a world *in motion*.

We can thus consider motion as important for orientation in the world, for understanding the dynamics of a scene, and for allowing us to use both to learn to interpret visual stimuli in the absence of direct evidence. These are important capabilities not only for humans and other animals, but for any entity that needs to interpret visual data, whether the purpose is to query a large image database, understand the activities of an elderly person in a household, or autonomously drive an exploration vehicle on Mars. My hope is that this thesis contributed a small step towards motion understanding, and hence the scope of capabilities of such systems.

Notation

Unless otherwise stated, these are the notations used throughout this thesis.

General conventions

Scalars and scalar functions	Standard lower case letters	$a, b, u_1(\mathbf{x})$
	Exceptions: Energy terms,	E
	robust functions	$\rho(z)$
Vectors and vector-valued functions	Bold lower case letters	$\mathbf{x}, \mathbf{q}, \mathbf{u}(\mathbf{x})$
	Exception: Images	$I(\mathbf{x})$
Matrices	Bold upper case letters	$\mathbf{A}$
Scalar constants	Greek lower case letters	λ, θ
Sets	Calligraphic notation	$\mathcal{N}, \mathcal{M}$

Subscripts denote individual vector elements (*e.g.* $\mathbf{u}(\mathbf{x}) = (u_1(\mathbf{x}), u_2(\mathbf{x}))$), points in time (*e.g.* I_t), or modifiers (*e.g.* E_{data}, λ_u). Superscripts denote the layer index (*e.g.* $\mathbf{u}^l$ for the optical flow of layer l). Bracketed superscripts indicate the value of a variable at a specific iteration

$$\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + (\Delta\mathbf{u})^{(i)}.$$

Hatted quantities are estimated values of a variable

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} f(\mathbf{x}).$$

All other modifiers are described in their respective contexts.

Specific symbols

The following are specific symbols that are used throughout this thesis. Other symbols will be introduced as required.

General symbols	

$\mathbf{x}$	2D location on the image plane
$\mathbf{x}'$	2D location of point corresponding to x on a different frame
$I(\mathbf{x})$	2D Image
H, W	Height and width of image.
$\mathbf{u}(\mathbf{x})$	Optical flow
$\rho(z)$	Robust error function
$\mathbf{F}$	Fundamental matrix
$\boldsymbol{\theta}$	Motion parameters
$\mathbf{w}(\mathbf{x}, \boldsymbol{\theta})$	Warped location of $\mathbf{x}$ according to motion parameters $\boldsymbol{\theta}$.
$\{\mathbf{b}_1(\mathbf{x}) \dots \mathbf{b}_B(\mathbf{x})\}$	Optical flow basis containing B basis flow fields.
$\mathcal{M} = \{(\mathbf{q}, \mathbf{q}')\}$	Set of correspondences
M	Total number of matches
$\mathbf{v}_m$	Displacement induced by feature match m
t	Point in time
$l \in \{1 \dots L\}$	Layer index

Chapter 3: Modeling blurred video with layers

$\mathbf{a}^l$	Unblurred appearance of layer l.
$\mathbf{g}^l$	Layer support (segmentation mask) of layer l.
$\mathbf{m}$	Binary mask to exclude pixels outside of the visible frame
$k(x)$	Heavy-side function

Chapter 4: Learning the structure of layered optical flow

$\bar{\mathbf{v}}$	Stacked displacements of all matched features
$\mathbf{A}$	Matrix for least squares problem
$\boldsymbol{\Gamma}$	Tikhonov regularizer
c_m	Assignment variable, denoting layer assigned to feature m.
$c(\mathbf{x})$	Dense assignment field

Chapter 5: From layers to geometry

C_1, C_2	Cameras
$\bar{\mathbf{u}}(\mathbf{x})$	Optical flow in backward direction
$\mathbf{x}_h$	$\mathbf{x}$ in homogeneous coordinates
$\mathbf{x}_w$	World coordinates of point projecting to $\mathbf{x}$
$\mathbf{H}$	Homography
$\boldsymbol{\Pi}$	Registration plane

$\mathbf{e}$	Epipole
$\mathbf{u}_p(\mathbf{x}) \in \mathbb{R}^2$	Parallax motion
$h(\mathbf{x}, t)$	Parallax field to frame t
$s(\mathbf{x})$	Structure
b_t	Camera motion after registration
$v^+(\mathbf{x}), v^-(\mathbf{x})$	Visibility maps
$r_{gt}(\mathbf{x})$	Ground truth rigidity map
$r(\mathbf{x})$	Rigidity map
$\mathcal{L}(\mathbf{x}, \mathbf{y})$	Line through $\mathbf{x}$ and $\mathbf{y}$
$m\left(I(\mathbf{x}), I(\mathbf{y})\right)$	Image-based neighborhood weight

Contributions

Parts of Chapter 5 were created in collaboration with Laura Sevilla-Lara. Specifically, Laura Sevilla-Lara implemented and trained the CNNs, and used those CNNs to compute the semantic rigidity estimation, as described in Sec. 5.4.1.

Apart from this, the work presented in this thesis is the sole work of Jonas Wulff.

Bibliography

[1] `http://imsc.uni-graz.at/optcon/projects/bredies/tgv.html`. *Accessed on 2017-07-22.*, 2010. 31

[2] `http://pcaflow.is.tue.mpg.de`, 2015. 84, 87

[3] `http://mrflow.is.tue.mpg.de`, 2017. 118, 125

[4] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984. 25

[5] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(4):384–401, July 1985. 50, 114

[6] O. Ait-Aider, N. Andreff, J. M. Lavest, and P. Martinet. Exploiting rolling shutter distortions for simultaneous object pose and velocity computation using a single view. In *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, pages 35–35, Jan 2006. 46

[7] K. Alahari, P. Kohli, and P. H. S. Torr. Reduce, reuse & recycle: Efficiently solving multi-label mrfs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2008. 130

[8] L. Alvarez, R. Deriche, T. Papadopoulo, and J. Sánchez. Symmetrical dense optical flow estimation with occlusions detection. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *European Conference on Computer Vision (ECCV)*, pages 721–735, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. 28

[9] L. Alvarez, R. Deriche, T. Papadopoulo, and J. Sánchez. Symmetrical dense optical flow estimation with occlusions detection. *International Journal of Computer Vision*, 75(3):371–385, 2007. 28

[10] L. Alvarez, J. Sánchez, and J. Weickert. A scale-space approach to nonlocal optical flow calculations. In M. Nielsen, P. Johansen, O. F. Olsen, and J. Weickert, editors, *Scale-Space Theories in Computer Vision: Second International Conference, Scale-Space'99 Corfu, Greece, September 26–27, 1999*

Proceedings, pages 235–246, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. 26, 30

[11] T. Amiaz and N. Kiryati. Dense discontinuous optical flow via contour-based segmentation. In *IEEE International Conference on Image Processing 2005*, volume 3, pages III–1264–7, Sept 2005. 35

[12] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310, 1989. 5, 25, 84

[13] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, May 2011. 3

[14] S. Ayer and H. S. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and mdl encoding. In *IEEE International Conference on Computer Vision (ICCV)*, pages 777–784, Jun 1995. 38

[15] A. Ayvaci, M. Raptis, and S. Soatto. Occlusion detection and motion estimation with convex optimization. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, pages 100–108. Curran Associates, Inc., 2010. 28

[16] A. Bab-Hadiashar and D. Suter. Robust total least squares based optic flow computation. In R. Chin and T.-C. Pong, editors, *Asian Conference on Computer Vision (ACCV)*, pages 566–573, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. 21

[17] M. Bai, W. Luo, K. Kundu, and R. Urtasun. Exploiting semantic information and deep matching for optical flow. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *European Conference on Computer Vision (ECCV)*, pages 154–170, Cham, 2016. Springer International Publishing. 34, 52, 115, 137, 138, 139

[18] C. Bailer, B. Taetz, and D. Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4015–4023, Dec 2015. 26, 28, 137, 139

[19] C. Bailer, K. Varanasi, and D. Stricker. Cnn-based patch matching for optical flow with thresholded hinge embedding loss. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 137, 139

[20] S. Baker, E. Bennett, S. B. Kang, and R. Szeliski. Removing rolling shutter wobble. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2392–2399, June 2010. 45

[21] S. Baker and T. Kanade. Super-resolution optical flow. Technical Report CMU-RI-TR-99-36, Carnegie Mellon University, The Robotics Institute, 1999. 146

[22] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004. 21, 98

[23] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. 17, 19, 25, 43

[24] L. Bao, Q. Yang, and H. Jin. Fast edge-preserving PatchMatch for large displacement optical flow. *IEEE Transactions on Image Processing*, 23(12):4996–5006, Dec 2014. 26, 83, 85

[25] L. Bar, B. Berkels, M. Rumpf, and G. Sapiro. A variational framework for simultaneous motion estimation and restoration of motion-blurred video. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2007. 48, 56, 60, 62, 63

[26] D. I. Barnea and H. Silverman. A class of algorithms for fast digital image registration. *Computers, IEEE Transactions on*, C-21(2):179–186, Feb 1972. 84

[27] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, Feb 1994. 17

[28] B. Bascle, A. Blake, and A. Zisserman. Motion deblurring and super-resolution from an image sequence. In *European Conference on Computer Vision (ECCV)*, volume II, pages 573–582. Springer-Verlag, 1996. 48

[29] M. Ben-Ezra and S. Nayar. Motion-based motion deblurring. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):689–698, 2004. 48

[30] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *European Conference on Computer Vision (ECCV)*, volume LNCS 588, pages 237–252. Springer, 1992. 22, 51

[31] J. R. Bergen, P. J. Burt, R. Hingorani, and S. Peleg. A three-frame algorithm for estimating two-component image motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):886–896, Sep 1992. 41

[32] P. Bideau and E. Learned-Miller. It's moving! a probabilistic model for causal motion segmentation in moving camera videos. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *European Conference on Computer Vision (ECCV)*, pages 433–449, Cham, 2016. Springer International Publishing. 114

[33] M. Black and A. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):972–986, Oct 1996. 33

[34] M. Black, Y. Yacoob, A. Jepson, and D. Fleet. Learning parameterized models of image motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 561–567, Jun 1997. 98

[35] M. J. Black. *Robust Incremental Optical Flow*. PhD thesis, Yale University, Department of Computer Science, New Haven, CT, 1992. Research Report YALEU-DCS-RR-923. 21, 42

[36] M. J. Black. Recursive non-linear estimation of discontinuous flow fields. In J.-O. Eklundh, editor, *European Conference on Computer Vision (ECCV)*, pages 138–145, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. 42

[37] M. J. Black and P. Anandan. A model for the detection of motion over time. In *IEEE International Conference on Computer Vision (ICCV)*, pages 33–37, Dec 1990. 28, 41

[38] M. J. Black and P. Anandan. Robust dynamic motion estimation over time. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 296–302, Jun 1991. 42

[39] M. J. Black and P. Anandan. A framework for the robust estimation of optical flow. In *IEEE International Conference on Computer Vision (ICCV)*, pages 231–236, May 1993. 5, 27, 29

[40] M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75 – 104, 1996. 7, 91

[41] M. J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International Journal of Computer Vision*, 19(1):57–91, 1996. 29

[42] M. J. Black and G. Sapiro. Edges as outliers: Anisotropic smoothing using local image statistics. In M. Nielsen, P. Johansen, O. F. Olsen, and J. Weickert, editors, *Scale-Space Theories in Computer Vision: Second International Conference, Scale-Space'99 Corfu, Greece, September 26–27, 1999 Proceedings*, pages 259–270, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. 122

[43] M. J. Black and Y. Yacoob. Recognizing facial expressions in image sequences using local parameterized models of image motion. *International Journal of Computer Vision*, 25(1):23–48, 1997. 22, 82

[44] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, MA, USA, 1987. 30

[45] M. Bleyer, C. Rhemann, and M. Gelautz. Segmentation-based motion with occlusions using graph-cut optimization. In K. Franke, K.-R. Müller, B. Nickolay, and R. Schäfer, editors, *Pattern Recognition: 28th DAGM Symposium, Berlin, Germany, September 12-14, 2006. Proceedings*, pages 465–474, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 38

[46] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, Nov 2001. 94

[47] J. Braux-Zin, R. Dupont, and A. Bartoli. A general dense image matching framework combining direct and feature-based costs. In *IEEE International Conference on Computer Vision (ICCV)*, pages 185–192, Dec 2013. 31

[48] K. Bredies, K. Kunisch, and T. Pock. Total generalized variation. *SIAM Journal on Imaging Sciences*, 3(3):492–526, 2010. 31

[49] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 690–696 vol.2, 2000. 53

[50] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In T. Pajdla and J. Matas, editors, *European Conference on Computer Vision (ECCV)*, pages 25–36, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. 25, 26, 27, 133

[51] T. Brox, A. Bruhn, and J. Weickert. Variational motion segmentation with level sets. In A. Leonardis, H. Bischof, and A. Pinz, editors, *European Conference on Computer Vision (ECCV)*, pages 471–483, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 35, 43

[52] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, March 2011. 26, 40, 83, 85, 88

[53] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005. 5

[54] D. Butler, J. Wulff, G. Stanley, and M. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *European Conference on Computer Vision (ECCV)*, volume 7577 of *Lecture Notes in Computer Science*, pages 611–625. Springer Berlin Heidelberg, 2012. 55, 83, 114, 118

[55] A. Chakrabarti, T. Zickler, and W. Freeman. Analyzing spatially-varying blur. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2512 –2519, June 2010. 49, 62

[56] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems withapplications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011. 26

[57] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *IEEE Int. Conf. Image Proc. (ICIP)*, volume 2, pages 168–172, 1994. 62, 63

[58] K. Chaudhury and R. Mehrotra. A trajectory-based computational model for optical flow estimation. *IEEE Transactions on Robotics and Automation*, 11(5):733–741, Oct 1995. 41

[59] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR*, abs/1412.7062, 2014. 124, 126

[60] Q. Chen and V. Koltun. Full flow: Optical flow estimation by global optimization over regular grids. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4706–4714, June 2016. 26

[61] W.-G. Chen, N. Nandhakumar, and W. Martin. Image motion estimation from motion smear-a new computational model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):412 –425, Apr. 1996. 49

[62] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu. Large displacement optical flow from nearest neighbor fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2443–2450, June 2013. 35

[63] M. Chessa, F. Solari, S. P. Sabatini, and G. M. Bisio. Motion interpretation using adjustable linear models. In *BMVC*, pages 1–10, 2008. 23

[64] T. M. Chin, W. C. Karl, and A. S. Willsky. Probabilistic and sequential computation of optical flow using temporal coherence. *IEEE Transactions on Image Processing*, 3(6):773–788, Nov 1994. 41, 42

[65] S. Cho and S. Lee. Fast motion deblurring. In *ACM Trans. Graphics (TOG) – Proc. SIGGRAPH Asia*, pages 145:1–145:8, New York, NY, USA, 2009. ACM. 47, 76, 77

[66] S. Cho, J. Wang, and S. Lee. Video deblurring for hand-held cameras using patch-based synthesis. *ACM Trans. Graph.*, 31(4):64:1–64:9, July 2012. 48

[67] S. Chunhe, Z. Hai, and J. Wei. Motion deblurring from a single image using multi-layer statistics priors. In *Consumer Electronics (ICCE), 2011 IEEE International Conference on*, pages 481–482, Jan 2011. 47, 56

[68] N. Cornelius and T. Kanade. Adapting optical-flow to measure object motion in reflectance and x-ray image sequences. Technical report, Carnegie-Mellon University, 1983. 30

[69] H. B. Cott. *Concealment in defence, mainly as illustrated by birds*, chapter 7, pages 117–139. Methuen, 1940. 3

[70] D. Cremers and S. Soatto. Motion competition: A variational approach to piecewise parametric motion segmentation. *International Journal of Computer Vision*, 62(3):249–265, May 2005. 35

[71] S. Dai and Y. Wu. Motion from blur. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2008. 49

[72] S. Dai and Y. Wu. Removing partial blur in a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2544–2551, 2009. 49

[73] T. Darrell and A. Pentland. Robust estimation of a multi-layered motion representation. In *Visual Motion, 1991., Proceedings of the IEEE Workshop on*, pages 173–178, Oct 1991. 35

[74] T. Darrell and A. P. Pentland. Cooperative robust estimation using layers of support. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):474–487, May 1995. 38

[75] R. Deriche, P. Kornprobst, and G. Aubert. Optical-flow estimation while preserving its discontinuities: A variational approach. In S. Z. Li, D. P. Mital, E. K. Teoh, and H. Wang, editors, *Asian Conference on Computer Vision (ACCV)*, pages 69–80, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. 29

[76] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, December 2015. 43, 84, 85, 149

[77] M. Egelhaaf and A. Borst. A look into the cockpit of the fly: visual orientation, algorithms, and identified neurons. *Journal of Neuroscience*, 13(11):4563–4574, 1993. 2

[78] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *European Conference on Computer Vision (ECCV)*, pages 834–849, Cham, 2014. Springer International Publishing. 51

[79] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, jun 2010. 124

[80] R. Fablet and M. Black. Automatic detection and tracking of human motion with a view-based representation. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *European Conference on Computer Vision (ECCV)*, volume 2350 of *Lecture Notes in Computer Science*, pages 476–491. Springer Berlin Heidelberg, 2002. 22

[81] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. *ACM Trans. Graphics (TOG) – Proc. SIGGRAPH*, 25(3):787–794, July 2006. 47, 56

[82] D. A. Fish, A. M. Brinicombe, E. R. Pike, and J. G. Walker. Blind deconvolution by means of the richardson–lucy algorithm. *J. Opt. Soc. Am. A*, 12(1):58–65, Jan 1995. 67

[83] D. J. Fleet, M. J. Black, Y. Yacoob, and A. D. Jepson. Design and use of linear models for image motion analysis. *International Journal of Computer Vision*, 36(3):171–193, 2000. 22, 23, 82, 87, 91

[84] D. Fortun, P. Bouthemy, and C. Kervrann. Optical flow modeling and computation: A survey. *Computer Vision and Image Understanding*, 134:1 – 21,

2015. Image Understanding for Real-world Distributed Video Networks. 17, 25

[85] A. Fragkiadaki, B. Seybold, R. Sukthankar, S. Vijayanarasimhan, and S. Ricco. Self-supervised learning of structure and motion from video. In *arxiv (2017)*, 2017. 149

[86] K. Fragkiadaki, M. Salas, P. Arbeláez, and J. Malik. Grouping-based low-rank trajectory completion and 3d reconstruction. In *Advances in Neural Information Processing Systems*, NIPS'14, pages 55–63, Cambridge, MA, USA, 2014. MIT Press. 53

[87] C. S. Fuh and P. Maragos. Region-based optical flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 130–135, Jun 1989. 33

[88] R. Garg, L. Pizarro, D. Rueckert, and L. Agapito. Dense multi-frame optic flow for non-rigid objects using subspace constraints. In R. Kimmel, R. Klette, and A. Sugimoto, editors, *Asian Conference on Computer Vision (ACCV)*, pages 460–473, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. 53

[89] R. Garg, A. Roussos, and L. Agapito. A variational approach to video registration with subspace constraints. *International Journal of Computer Vision*, 104(3):286–314, Sep 2013. 53

[90] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research*, 32(11):1231–1237, Sept. 2013. 43, 83, 101

[91] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, June 2012. 52

[92] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3D reconstruction in real-time. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 963–968, June 2011. 88

[93] D. Gibson and M. Spann. Robust optical flow estimation based on a sparse motion trajectory set. *IEEE Transactions on Image Processing*, 12(4):431–445, April 2003. 84

[94] J. Gibson and O. Marques. Sparse regularization of TV-L1 optical flow. In A. Elmoataz, O. Lezoray, F. Nouboud, and D. Mammass, editors, *Image and Signal Processing*, volume 8509 of *Lecture Notes in Computer Science*, pages 460–467. Springer International Publishing, 2014. 32

[95] J. J. Gibson. *The perception of the visual world*. Houghton Mifflin, 1950. 2

[96] B. Glocker, T. H. Heibel, N. Navab, P. Kohli, and C. Rother. Triangleflow: Optical flow with triangulation-based higher-order likelihoods. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *European Conference on Computer Vision (ECCV)*, pages 272–285, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. 38

[97] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, pages 2672–2680. Curran Associates, Inc., 2014. 149

[98] K. G. Götz. Optomotorische Untersuchung des visuellen systems einiger Augenmutanten der Fruchtfliege Drosophila. *Kybernetik*, 2(2):77–92, 1964. 2

[99] M. Grundmann, V. Kwatra, D. Castro, and I. Essa. Calibration-free rolling shutter removal. In *IEEE International Conference Computational Photography (ICCP)*, pages 1–8, 2012. 45, 146

[100] J. Gu, Y. Hitomi, T. Mitsunaga, and S. Nayar. Coded rolling shutter photography: Flexible space-time sampling. In *IEEE International Conference on Computational Photography (ICCP)*, pages 1–8, March 2010. 46

[101] F. Güney and A. Geiger. Deep discrete flow. In *Asian Conference on Computer Vision (ACCV)*, 2016. 5, 27

[102] T. Guthier, J. Eggert, and V. Willert. Unsupervised learning of motion patterns. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, volume 20, pages 323–328, Bruges, April 2012. 22

[103] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, Oct 1998. 21

[104] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, March 2009. 84

[105] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. John Wiley & Sons, Inc., 1986. 27

[106] K. J. Hanna. Direct multi-resolution estimation of ego-motion and structure from motion. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 156–162, Oct 1991. 51

[107] S. Hauberg, A. Feragen, and M. Black. Grassmann averages for scalable robust PCA. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3810–3817, June 2014. 83, 87

[108] K. He and J. Sun. Computing nearest-neighbor fields via propagation-assisted kd-trees. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 111–118, June 2012. 26

[109] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1397–1409, June 2013. 92

[110] X. He, T. Luo, S. C. Yuk, K. Chow, K. Wong, and R. H. Y. Chung. Motion estimation method for blurred videos and application of deblurring with spatially varying blur kernels. In *Int. Conf. on Computer Sciences and Convergence Information Technology (ICCIT)*, pages 355–359, 2010. 48

[111] J. Hedborg, E. Ringaby, P. E. Forssn, and M. Felsberg. Structure and motion estimation from rolling shutter video. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 17–23, Nov 2011. 45

[112] D. Heeger and A. Jepson. Subspace methods for recovering rigid motion I: Algorithm and implementation. *International Journal of Computer Vision*, 7(2):95–117, 1992. 50

[113] D. J. Heeger. Model for the extraction of image flow. *J. Opt. Soc. Am. A*, 4(8):1455–1471, Aug 1987. 40

[114] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, Feb 2008. 26

[115] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. In *ACM SIGGRAPH*, SIGGRAPH '05, pages 577–584, New York, NY, USA, 2005. ACM. 36

[116] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1–3):185–203, 1981. 6, 24, 27, 84

[117] B. K. P. Horn and E. J. Weldon. Direct methods for recovering motion. *International Journal of Computer Vision*, 2(1):51–76, June 1988. 51, 116

[118] M. Hornáček, F. Besse, J. Kautz, A. Fitzgibbon, and C. Rother. Highly overparameterized optical flow using patchmatch belief propagation. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *European Conference on Computer Vision (ECCV)*, pages 220–234, Cham, 2014. Springer International Publishing. 32

[119] J. Hur and S. Roth. Joint optical flow and temporally consistent semantic segmentation. In G. Hua and H. Jégou, editors, *European Conference on Computer Vision Workshops (ECCV Workshops)*, pages 163–177, Cham, 2016. Springer International Publishing. 34, 52

[120] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. 85, 138, 139

[121] M. Irani. Multi-frame correspondence estimation using subspace constraints. *International Journal of Computer Vision*, 48(3):173–194, Jul 2002. 42, 52

[122] M. Irani and P. Anandan. A unified approach to moving object detection in 2d and 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):577–589, Jun 1998. 118, 127

[123] M. Irani, P. Anandan, and M. Cohen. Direct recovery of planar-parallax from multiple frames. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11):1528–1534, Nov 2002. 116, 118

[124] M. Irani, P. Anandan, and D. Weinshall. From reference frames to reference planes: Multi-view parallax geometry and applications. In H. Burkhardt and B. Neumann, editors, *European Conference on Computer Vision (ECCV)*, volume 1407 of *Lecture Notes in Computer Science*, pages 829–845. Springer Berlin Heidelberg, 1998. 116, 119

[125] M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using region alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):268–272, Mar. 1997. 119

[126] J. Janai, F. Güney, J. Wulff, M. Black, and A. Geiger. Slow flow: Exploiting high-speed cameras for accurate and diverse optical flow reference data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 41, 84

[127] A. Jepson and M. Black. Mixture models for optical flow computation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 760–761, Jun 1993. 22, 83

[128] A. Jepson and M. Black. Mixture models for optical flow computation. In *Partitioning Data Sets, DIMACS Workshop*, pages 271–286. AMS Pub, Providence, RI., Apr. 1993. 37

[129] A. Jepson and M. Black. Mixture models for image representation. Technical report, University of Toronto, 1996. 38

[130] A. D. Jepson, D. J. Fleet, and M. J. Black. A layered motion representation with occlusion and compact spatial support. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *European Conference on Computer Vision (ECCV)*, volume 2350 of *Lecture Notes in Computer Science*, pages 692–706. Springer Berlin Heidelberg, 2002. 38

[131] K. Jia, X. Wang, and X. Tang. Optical flow estimation using learned sparse model. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2391–2398, Nov 2011. 32

[132] H. Jin, P. Favaro, and R. Cipolla. Visual tracking in the presence of motion blur. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 18–25 vol. 2, 2005. 48

[133] N. Jojic and B. Frey. Learning flexible sprites in video layers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–199–I–206 vol.1, 2001. 37

[134] C. Jordan. Sur la série de Fourier. *C. R. Acad. Sci., Paris*, 92:228–230, 1881. 30

[135] S. X. Ju, M. J. Black, and A. D. Jepson. Skin and bones: multi-layer, locally affine, optical flow and regularization with transparency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 307–314, June 1996. 19, 32, 39

[136] F. Jurie and M. Dhome. Hyperplane approximation for template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):996–1000, Jul 2002. 21

[137] Z. Kalal, K. Mikolajczyk, and J. Matastun. Forward-backward error: Automatic detection of tracking failures. In *International Conference on Pattern Recognition*, pages 2756–2759, Aug 2010. 123

[138] A. Kar, J. Malik, and C. Häne. Learning a multi-view stereo machine. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 364–375. Curran Associates, Inc., 2017. 149

[139] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-end learning of geometry and context for deep stereo regression. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 149

[140] R. Kennedy and C. Taylor. Optical flow with geometric occlusion estimation and fusion of multiple frames. In X.-C. Tai, E. Bae, T. Chan, and M. Lysaker, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, volume 8932 of *Lecture Notes in Computer Science*, pages 364–377. Springer International Publishing, 2015. 34, 42

[141] R. Kennedy and C. J. Taylor. Hierarchically-constrained optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3340–3348, June 2015. 34, 42

[142] T. H. Kim and K. M. Lee. Generalized video deblurring for dynamic scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5426–5434, June 2015. 49

[143] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004. 67

[144] P. Krähenbühl and V. Koltun. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In *Advances in Neural Information Processing Systems*, pages 109–117, 2011. 39, 111, 146

[145] P. Krähenbühl and V. Koltun. Efficient nonlocal regularization for optical flow. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *European Conference on Computer Vision (ECCV)*, volume 7572 of *Lecture Notes in Computer Science*, pages 356–369. Springer Berlin Heidelberg, 2012. 30, 126

[146] M. Lang, O. Wang, T. Aydin, A. Smolic, and M. Gross. Practical temporal consistency for image-based graphics applications. *ACM Trans. Graph.*, 31(4):34:1–34:8, July 2012. 40, 43

[147] D. T. Lawton. Processing translational motion sequences. *Computer Vision, Graphics, and Image Processing*, 22:116–144, 1983. 50

[148] H. V. Le, G. Seetharaman, and B. Zavidovique. A multiscale technique for optical flow computation using piecewise affine approximation. In *IEEE International Workshop on Computer Architectures for Machine Perception*, pages 10 pp.–230, May 2003. 34

[149] S. H. Lee, N. S. Moon, and C.-W. Lee. Recovery of blurred video signals using iterative image restoration combined with motion estimation. In *Int. Conf. on Image Processing (ICIP)*, volume 1, pages 755–758, 1997. 48

[150] C. Lei and Y. H. Yang. Optical flow estimation on coarse-to-fine region-trees using discrete optimization. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1562–1569, Sept 2009. 34

[151] V. Lempitsky, S. Roth, and C. Rother. Fusionflow: Discrete-continuous optimization for optical flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2008. 35, 93

[152] M. Leordeanu, A. Zanfir, and C. Sminchisescu. Locally affine sparse-to-dense matching for motion and occlusion estimation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1721–1728, Dec 2013. 32

[153] A. Levin. Blind motion deblurring using image statistics. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, pages 841–848. MIT Press, Cambridge, MA, 2007. 47, 56, 63

[154] W. Li, Y. Chen, J. Lee, G. Ren, and D. Cosker. Blur robust optical flow using motion channel. *Neurocomputing*, 220:170 – 180, 2017. Recent Research in Medical Technology Based on Multimedia and Pattern Recognition. 48

[155] Y. Li and D. P. Huttenlocher. Sparse long-range random field and its application to image denoising. In D. Forsyth, P. Torr, and A. Zisserman, editors, *European Conference on Computer Vision (ECCV)*, volume 5304 of *Lecture Notes in Computer Science*, pages 344–357. Springer Berlin Heidelberg, 2008. 33

[156] Y. Li, S. B. Kang, N. Joshi, S. Seitz, and D. Huttenlocher. Generating sharp panoramas from motion-blurred videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2424–2431, 2010. 48

[157] C. K. Liang, L. W. Chang, and H. H. Chen. Analysis and compensation of rolling shutter effect. *IEEE Transactions on Image Processing*, 17(8):1323–1330, Aug 2008. 45

[158] H. T. Lin, Y.-W. Tai, and M. Brown. Motion regularization for matting motion blurred objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2329 –2336, Nov. 2011. 49

[159] J. Lu, H. Yang, D. Min, and M. Do. Patch Match Filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1854–1861, June 2013. 26

[160] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981. 20, 21, 91

[161] O. Mac Aodha, A. Humayun, M. Pollefeys, and G. J. Brostow. Learning a confidence measure for optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5):1107–1120, May 2013. 35, 93

[162] W. J. MacLean. Removal of translation bias when using subspace methods. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 753–758 vol.2, 1999. 122

[163] W. J. MacLean, A. D. Jepson, and R. C. Frecker. Recovery of egomotion and segmentation of independent object motion using the em algorithm. In *Proceedings of the British Machine Vision Conference*, pages 17.1–17.10. BMVA Press, 1994. doi:10.5244/C.8.17. 34

[164] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 689–696, New York, NY, USA, 2009. ACM. 88

[165] S. Mallat. *A wavelet tour of signal processing: the sparse way*. Academic press, 2008. 124

[166] R. Mandelbaum, G. Salgian, and H. Sawhney. Correlation-based estimation of ego-motion and structure from motion and stereo. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 544–550 vol.1, 1999. 51

[167] D. Marr. *Vision – A Computational Investigation into the Human Representation and Processing of Visual Information*. MIT Press, 1982. 149

[168] N. Mayer, E. Ilg, P. Husser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, June 2016. 84

[169] P. Meer, D. Mintz, A. Rosenfeld, and D. Y. Kim. Robust regression methods for computer vision: A review. *International Journal of Computer Vision*, 6(1):59–70, 1991. 29

[170] M. Meilland, T. Drummond, and A. I. Comport. A unified rolling shutter and motion blur model for 3d visual registration. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2016–2023, Dec 2013. 45

[171] M. Meingast, C. Geyer, and S. Sastry. Geometric models of rolling-shutter cameras. Technical report, arXiv:cs/0503076v1, 2005. 45

[172] É. Mémin and P. Pérez. Joint estimation-segmentation of optic flow. In H. Burkhardt and B. Neumann, editors, *European Conference on Computer Vision (ECCV)*, pages 563–577, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. 36

[173] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070. IEEE, June 2015. 52, 116, 118

[174] M. Menze, C. Heipke, and A. Geiger. Discrete optimization for optical flow. In *German Conference on Pattern Recognition (GCPR)*, volume 9358, pages 16–28. Springer International Publishing, 2015. 27, 116, 120, 136, 138, 139

[175] H. Mobahi, C. L. Zitnick, and Y. Ma. Seeing through the blur. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1736–1743, June 2012. 25, 26

[176] M. Mohamed, H. Rashwan, B. Mertsching, M. Garcia, and D. Puig. Illumination-robust optical flow using a local directional pattern. *Circuits and Systems for Video Technology, IEEE Transactions on*, 24(9):1499–1508, Sept 2014. 83

[177] D. W. Murray and B. F. Buxton. Scene segmentation from visual motion using global optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(2):220–228, March 1987. 41

[178] H.-H. Nagel. Extending the 'oriented smoothness constraint' into the temporal domain and the estimation of derivatives of optical flow. In O. Faugeras, editor, *European Conference on Computer Vision (ECCV)*, pages 139–148, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg. 41

[179] H. H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(5):565–593, Sept 1986. 30

[180] S. Negahdaripour and B. K. P. Horn. Direct passive navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(1):168–176, Jan 1987. 51

[181] M. Nicolescu and G. Medioni. Layered 4D representation and voting for grouping from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):492–501, April 2003. 84

[182] T. Nir, A. M. Bruckstein, and R. Kimmel. Over-parameterized variational optical flow. *International Journal of Computer Vision*, 76(2):205–216, 2008. 19, 32, 52

[183] T. Nir, R. Kimel, and A. Bruckstein. Variational approach for joint optic-flow computation and video restoration. Technical Report CIS-2005-03, Technion Israel Institute of Technology, 2005. 48, 66

[184] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980. 68, 122

[185] L. Oisel, E. Memin, L. Morin, and C. Labit. Epipolar constrained motion estimation for reconstruction from video sequences. *Proc. SPIE*, 3309:460–468, 1998. 52

[186] S. Oron, A. Bar-Hille, and S. Avidan. Extended lucas-kanade tracking. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *European Conference on Computer Vision (ECCV)*, pages 142–156, Cham, 2014. Springer International Publishing. 35

[187] Y. Ostrovsky, E. Meyers, S. Ganesh, U. Mathur, and P. Sinha. Visual parsing after recovery from blindness. *Psychological Science*, 20(12):1484–1491, 2009. 3

[188] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1777–1784, Dec 2013. 114

[189] C. Paramanand and A. Rajagopalan. Non-uniform motion deblurring for bilayer scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1115–1122, 2013. 49, 78, 146

[190] D. Pathak, R. B. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. Technical report, arXiv, 2016. 3

[191] M. Pawan Kumar, P. Torr, and A. Zisserman. Learning layered motion segmentations of video. *International Journal of Computer Vision*, 76(3):301–319, 2008. 38, 56

[192] T. Portz, L. Zhang, and H. Jiang. Optical flow in the presence of spatially-varying motion blur. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1752–1759, 2012. 48, 74, 76, 77

[193] R. Ranftl, K. Bredies, and T. Pock. Non-local total generalized variation for optical flow estimation. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars,

editors, *European Conference on Computer Vision (ECCV)*, pages 439–454, Cham, 2014. Springer International Publishing. 32

[194] J. Rannacher. Realtime 3D motion estimation on graphics hardware. Undergraduate Thesis, Heidelberg University, 2009. 85

[195] A. Rav-Acha, P. Kohli, C. Rother, and A. Fitzgibbon. Unwrap mosaics: A new representation for video editing. *ACM Transactions on Graphics (TOG) - Proc. SIGGRAPH*, 27(3):17:1–17:11, Aug. 2008. 47

[196] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1164–1172, June 2015. 7, 14, 30

[197] R. Roberts and F. Dellaert. Direct superpixel labeling for mobile robot navigation using learned general optical flow templates. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 1032–1037, Sept 2014. 23

[198] R. Roberts, C. Potthast, and F. Dellaert. Learning general optical flow subspaces for egomotion estimation and detection of motion anomalies. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 57–64, June 2009. 23

[199] D. Rosenbaum, D. Zoran, and Y. Weiss. Learning the local statistics of optical flow. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, pages 2373–2381. MIT Press, 2013. 33

[200] S. Roth and M. J. Black. On the spatial statistics of optical flow. *International Journal of Computer Vision*, 74(1):33–50, 2007. 32

[201] S. Roth and M. J. Black. Fields of experts. *International Journal of Computer Vision*, 82(2):205–29, Apr. 2009. 32

[202] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, Aug. 2004. 95, 130, 132

[203] A. Roussos, C. Russell, R. Garg, and L. Agapito. Dense multibody motion estimation and reconstruction from a handheld camera. In *IEEE Intl Symposium on Mixed and Augmented Reality (ISMAR 2012)*, 2012. 52

[204] C. Russell, R. Yu, and L. Agapito. Video pop-up: Monocular 3d reconstruction of dynamic scenes. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *European Conference on Computer Vision (ECCV)*, pages 583–598, Cham, 2014. Springer International Publishing. 54

[205] A. Salgado and J. Sánchez. Temporal constraints in large optical flow estimation. In R. Moreno Díaz, F. Pichler, and A. Quesada Arencibia, editors, *Computer Aided Systems Theory – EUROCAST 2007: 11th International Conference on Computer Aided Systems Theory, Las Palmas de Gran Canaria, Spain, February 12-16, 2007, Revised Selected Papers*, pages 709–716, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. 42

[206] P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. *International Journal of Computer Vision*, 80(1):72, 2008. 28, 40

[207] C. Sanderson. Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments. Technical report, NICTA, 2010. 103

[208] O. Saurer, K. Kser, J. Y. Bouguet, and M. Pollefeys. Rolling shutter stereo. In *IEEE International Conference on Computer Vision (ICCV)*, pages 465–472, Dec 2013. 45

[209] H. Sawhney. 3d geometry from planar parallax. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 929–934, Jun 1994. 116, 118, 119

[210] H. Sawhney, Y. Gao, and R. Kumar. Independent motion detection in 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1191–1199, Oct. 1999. 118

[211] T. Schoenemann and D. Cremers. A coding-cost framework for super-resolution motion layer decomposition. *IEEE Transactions on Image Processing*, 21(3):1097–1110, March 2012. 56

[212] Y. Schoueri, M. Scaccia, and I. Rekleitis. Optical flow from motion blurred color images. In *Computer and Robot Vision, 2009. CRV '09. Canadian Conference on*, pages 1–7, 2009. 49

[213] S. Seitz and S. Baker. Filter flow. In *IEEE International Conference on Computer Vision (ICCV)*, pages 143–150, 2009. 60, 62, 145

[214] H. Sekkati and A. Mitiche. Joint optical flow estimation, segmentation, and 3d interpretation with level sets. *Computer Vision and Image Understanding*, 103(2):89 – 100, 2006. 35

[215] A. Sellent, M. Eisemann, B. Goldlucke, D. Cremers, and M. Magnor. Motion field estimation from alternate exposure images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1577–1589, 2011. 48

[216] L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black. Optical flow with semantic segmentation and localized layers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 34, 115, 116, 124, 126, 146, 147

[217] L. Sevilla-Lara, D. Sun, E. G. Learned-Miller, and M. J. Black. Optical flow estimation with channel constancy. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *European Conference on Computer Vision (ECCV)*, pages 423–438, Cham, 2014. Springer International Publishing. 26

[218] Q. Shan, W. Xiong, and J. Jia. Rotational motion deblurring of a rigid object from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007. 49

[219] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1154–1160, Jan 1998. 42

[220] M. Shizawa and K. Maze. Unified computational theory for motion transparency and motion boundaries based on eigenenergy analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 289–295, Jun 1991. 37

[221] D. Shulman and J. Y. Herve. Regularization of discontinuous flow fields. In *Proceedings. Workshop on Visual Motion*, pages 81–86, Mar 1989. 29

[222] H.-Y. Shum and R. Szeliski. Systems and experiment paper: Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision*, 36(2):101–130, 2000. 21

[223] C. Silva and J. Santos-Victor. Motion from occlusions. *Robotics and Autonomous Systems*, 35(3):153 – 162, 2001. Seventh Symposium on Intelligent Robotic Systems - SIRS'99. 28

[224] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, Nov 2008. 50

[225] F. Steinbrücker, T. Pock, and D. Cremers. Large displacement optical flow computation withoutwarping. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1609–1614, Sept 2009. 26

[226] C. Strecha, R. Fransens, and L. Van Gool. A probabilistic approach to large displacement optical flow and occlusion detection. In D. Comaniciu, R. Mester, K. Kanatani, and D. Suter, editors, *Statistical Methods in Video Processing: ECCV 2004 Workshop SMVP 2004, Prague, Czech Republic, May 16, 2004, Revised Selected Papers*, pages 71–82, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. 27

[227] S. Su and W. Heidrich. Rolling shutter motion deblurring. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1529–1537, June 2015. 46

[228] D. Sun, C. Liu, and H. Pfister. Local layering for joint motion estimation and occlusion detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1098–1105, June 2014. 7, 9, 39

[229] D. Sun, S. Roth, and M. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137, 2014. 5, 7, 26, 27, 30, 81, 83

[230] D. Sun, S. Roth, J. Lewis, and M. J. Black. Learning optical flow. In *European Conference on Computer Vision (ECCV)*, volume 5304 of *LNCS*, pages 83–97. Springer-Verlag, Oct. 2008. 33

[231] D. Sun, E. Sudderth, and M. J. Black. Layered segmentation and optical flow estimation over time. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1768–1775, June 2012. 39, 43, 56, 58, 79, 81, 83, 146

[232] D. Sun, E. B. Sudderth, and M. J. Black. Layered image motion with explicit occlusions, temporal consistency, and depth ordering. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, pages 2226–2234. Curran Associates, Inc., 2010. 7, 39

[233] D. Sun, J. Wulff, E. Sudderth, H. Pfister, and M. Black. A fully-connected layered model of foreground and background flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2451–2458, June 2013. 39, 74, 76, 77, 78, 81, 83, 146

[234] Y. Sun and G. Liu. Rolling shutter distortion removal based on curve interpolation. *IEEE Transactions on Consumer Electronics*, 58(3):1045–1050, August 2012. 45

[235] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. Technical Report UCB/EECS-2010-104, EECS Department, University of California, Berkeley, Jul 2010. 40, 85

[236] S. Süsstrunk, P. Fua, A. Shaji, A. Lucchi, K. Smith, and R. Achanta. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, Nov 2012. 39

[237] R. Szeliski. *Image Alignment and Stitching: A Tutorial*, volume 2. Now Publishers Inc., Hanover, MA, USA, Jan. 2006. 21

[238] M. A. Taalebinezhaad. Direct recovery of motion and shape in the general case by fixation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 451–455, Dec 1990. 51

[239] H. Takeda and P. Milanfar. Removing motion blur with space-time processing. *IEEE Transactions on Image Processing*, 20(10):2990–3000, 2011. 48

[240] M. Tao, J. Bai, P. Kohli, and S. Paris. Simpleflow: A non-iterative, sublinear optical flow algorithm. *Computer Graphics Forum*, 31(2pt1):345–353, 2012. 85

[241] W. B. Thompson. Exploiting discontinuities in optical flow. *International Journal of Computer Vision*, 30(3):163–173, 1998. 27

[242] W. B. Thompson and T.-C. Pong. Detecting moving objects. *International Journal of Computer Vision*, 4:39–57, 1990. 34, 114

[243] E. Tola, V. Lepetit, and P. Fua. DAISY: An Efficient Dense Descriptor Applied to Wide Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830, May 2010. 27

[244] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, Nov 1992. 51, 53

[245] P. H. S. Torr, R. Szeliski, and P. Anandan. An integrated bayesian approach to layer extraction from image sequences. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 983–990 vol.2, 1999. 7

[246] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1521–1528, June 2011. 148

[247] L. Torresani and C. Bregler. Space-time tracking. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *European Conference on Computer Vision (ECCV)*, pages 801–812, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. 53

[248] B. Triggs. Plane + parallax, tensors and factorization. In *European Conference on Computer Vision (ECCV)*, volume LNCS 1842, pages 522–538. Springer, 2000. 119

[249] W. Trobin, T. Pock, D. Cremers, and H. Bischof. An unbiased second-order prior for high-accuracy motion estimation. In G. Rigoll, editor, *Pattern Recognition: 30th DAGM Symposium Munich, Germany, June 10-13, 2008 Proceedings*, pages 396–405, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. 31, 32

[250] Z. Tu, R. Poppe, and R. Veltkamp. Estimating accurate optical flow in the presence of motion blur. *Journal of Electronic Imaging*, 24(5):053018, 2015. 48

[251] S. Ullman. *The interpretation of visual motion*. Massachusetts Institute of Technology Press, 1979. 17

[252] L. Valgaerts, A. Bruhn, and J.Weickert. A variational model for the joint recovery of the fundamental matrix and the optical flow. In *DAGM*, 2008. 52

[253] C. Vazquez, A. Mitiche, and R. Laganiere. Joint multiregion segmentation and parametric estimation of image motion by basis function representation and level set evolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(5):782–793, May 2006. 36

[254] L. A. Vese and T. F. Chan. A multiphase level set framework for image segmentation using the mumford and shah model. *International Journal of Computer Vision*, 50(3):271–293, 2002. 35

[255] C. Vogel, S. Roth, and K. Schindler. An evaluation of data costs for optical flow. In J. Weickert, M. Hein, and B. Schiele, editors, *Pattern Recognition: 35th German Conference, GCPR 2013, Saarbrücken, Germany, September 3-6, 2013. Proceedings*, pages 343–353, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 32

[256] C. Vogel, K. Schindler, and S. Roth. Piecewise rigid scene flow. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1377–1384, Dec 2013. 35

[257] S. Volz, A. Bruhn, L. Valgaerts, and H. Zimmer. Modeling temporal coherence for optical flow. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1116–1123, Nov 2011. 42

[258] H. Von Helmholtz. *Handbuch der physiologischen Optik*, volume 9. Voss, 1867. 17

[259] C.-M. Wang, K.-C. Fan, and C.-T. Wang. Estimating optical flow by integrating multi-frame information. *Journal of Information Science and Engineering*, 24(6):1719–1731, Nov 2008. 41

[260] J. Wang and E. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, Sep 1994. 7, 37, 58, 67

[261] J. Y. A. Wang and E. H. Adelson. System for encoding image data into multiple layers representing regions of coherent motion and associated motion parameters. US Pat. 5557684, 1996. 48

[262] A. M. Waxman and S. Ullman. Surface structure and three-dimensional motion from image flow kinematics. *The International Journal of Robotics Research*, 4(3):72–94, 1985. 50

[263] J. Weber and J. Malik. Rigid body segmentation and shape description from dense optical flow under weak perspective. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):139–143, Feb. 1997. 34, 114

[264] A. Wedel, D. Cremers, T. Pock, and H. Bischof. Structure- and motion-adaptive regularization for high accuracy optic flow. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1663–1668, Sept 2009. 30, 52

[265] A. Wedel, T. Pock, J. Braun, U. Franke, and D. Cremers. Duality tv-l1 flow with fundamental matrix prior. In *International Conference Image and Vision Computing New Zealand*, pages 1–6, Nov 2008. 52

[266] J. Weickert and C. Schnörr. Variational optic flow computation with a spatio-temporal smoothness constraint. *Journal of Mathematical Imaging and Vision*, 14(3):245–255, May 2001. 41

[267] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1385–1392, Dec 2013. 27, 84, 86, 88

[268] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 520–526, Jun 1997. 9, 39, 58

[269] Y. Weiss and E. H. Adelson. Motion estimation and segmentation using a recurrent mixture of experts architecture. In *Proceedings of 1995 IEEE Workshop on Neural Networks for Signal Processing*, pages 293–302, Aug 1995. 37

[270] M. Werlberger, T. Pock, and H. Bischof. Motion estimation with non-local total variation regularization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2464–2471, June 2010. 30

[271] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 optical flow. In *BMVC*, London, UK, September 2009. 41, 83, 85, 86

[272] M. Wertheimer. Experimentelle studien über das sehen von bewegung. *Zeitschrift für Psychologie*, 61(1):161 – 265, 1912. 17

[273] M. Wertheimer. Untersuchungen zur lehre von der gestalt. ii. *Psychologische forschung*, 4(1):301–350, 1923. 17

[274] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. Non-uniform deblurring for shaken images. *International Journal of Computer Vision*, 98(2):168–186, 2012. 47

[275] J. Wills, S. Agarwal, and S. Belongie. A feature-based approach for dense segmentation and estimation of large disparity motion. *International Journal of Computer Vision*, 68(2):125–143, Jun 2006. 35

[276] J. Wulff and M. Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 120–130, June 2015. 81

[277] J. Wulff and M. J. Black. Modeling blurred video with layers. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *European Conference on Computer Vision (ECCV)*, volume 8694 of *Lecture Notes in Computer Science*, pages 236–252. Springer International Publishing, 2014. 55

[278] J. Wulff, L. Sevilla-Lara, and M. J. Black. Optical flow in mostly rigid scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 113

[279] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi. Bilateral filtering-based optical flow estimation with occlusion detection. In A. Leonardis, H. Bischof, and A. Pinz, editors, *European Conference on Computer Vision (ECCV)*, pages 211–224, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 27, 28

[280] J. Xiao and M. Shah. Motion layer extraction in the presence of occlusion using graph cut. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–972–II–979 Vol.2, June 2004. 38, 43

[281] J. Xu, R. Ranftl, and V. Koltun. Accurate optical flow via direct cost volume processing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 26, 137, 139, 147

[282] L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1744–1757, 2012. 35, 56, 66, 67, 74, 76, 77, 83, 88

[283] H. Yalcin, M. J. Black, and R. Fablet. The dense estimation of motion and appearance in layers. In *IEEE Workshop on Image and Video Registration*, June 2004. 39, 43

[284] K. Yamaguchi, D. McAllester, and R. Urtasun. Robust monocular epipolar flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1862–1869, June 2013. 52

[285] T. Yamaguchi, H. Fukuda, R. Furukawa, H. Kawasaki, and P. Sturm. Video deblurring and super-resolution technique for multiple moving objects. In *Asian Conference on Computer Vision (ACCV)*, volume IV, pages 127–140, Berlin, Heidelberg, 2011. Springer-Verlag. 48

[286] J. Yang and H. Li. Dense, accurate optical flow estimation with piecewise parametric model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1019–1027, June 2015. 36

[287] F. Yu and D. Gallup. 3d reconstruction from accidental motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3986–3993, June 2014. 53

[288] C. Yuan, G. Medioni, J. Kang, and I. Cohen. Detecting motion regions in the presence of a strong parallax from a moving camera by multiview geometric constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9):1627–1641, Sept. 2007. 118

[289] J. Yuan, C. Schrr, and G. Steidl. Simultaneous higher-order optical flow estimation and decomposition. *SIAM Journal on Scientific Computing*, 29(6):2283–2304, 2007. 31

[290] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In F. Hamprecht, C. Schnrr, and B. Jhne, editors, *Pattern Recognition*, volume 4713 of *Lecture Notes in Computer Science*, pages 214–223. Springer Berlin Heidelberg, 2007. 29, 30, 84, 100

[291] L. Zelnik-Manor and M. Irani. Multi-frame estimation of planar motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1105–1116, Oct 2000. 51, 119

[292] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 149

[293] Y. Zhou and H. Tao. A background layer model for object tracking through occlusion. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1079–1085 vol.2, 2003. 38

[294] C. W. Zitnick, N. Jojic, and S. B. Kang. Consistent segmentation for optical flow estimation. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1308–1315 Vol. 2, Oct 2005. 36

[295] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006. 88

[296] K. Zuiderveld. Contrast limited adaptive histogram equalization. In P. S. Heckbert, editor, *Graphics Gems IV*, pages 474–485. Academic Press Professional, Inc., San Diego, CA, USA, 1994. 88