FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Geometrically Motivated Dense Large Displacement Matching in Continuous Label Spaces

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der technischen Wissenschaften

by

Dipl.-Ing. Michael Hornáček
Registration Number 0848108

to the Faculty of Informatics
at the Vienna University of Technology

Advisors: Ao.Univ.Prof. Dipl.-Ing. Mag. Dr. Margrit Gelautz, Vienna, TU Vienna, Austria
Prof. Ph.D. Carsten Rother, TU Dresden, Dresden, Germany

The dissertation has been reviewed by:

(Prof. Ph.D. Carsten Rother) (doc. Mgr. Ondřej Chum, Ph.D.)

Vienna, 1.12.2014

(Dipl.-Ing. Michael Hornáček)

Erklärung zur Verfassung der Arbeit

Dipl.-Ing. Michael Hornáček
Schönbrunnerstr 81/16, 1050 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Acknowledgements

I owe a debt of gratitude to a number of people, whose feedback, support, or even source code helped me reach this point. Of these people, nobody contributed as much to my Ph.D. as Carsten Rother, who in addition to sharing a wealth of ideas motivated me better than anybody by being on many occasions astonishingly difficult to convince or impress. I gratefully acknowledge the financial support of Microsoft Research Cambridge through its European Ph.D. scholarship programme, which included not only a generous scholarship but also supplemental funding to visit their lab in Cambridge and the conferences I attended during my studies. I would like to thank A Min Tjoa and Christian Breiteneder for their indispensable contribution to allowing me to direct more of my attention to matters of research. I would also like to thank Carsten, Frederic Besse, Andrew Fitzgibbon, and Jan Kautz for the work they put into our joint paper on optical flow, and for giving me a chance to join the project and contribute my ideas. I am immensely grateful to Ondřej Chum for his feedback concerning my chapter on geometry, and to him and Zuzana Kúkelová for their help (in part over a memorable glass or two of exquisite Moravian wine) in getting my head around pulling rigid body motions out of the 5 point algorithm. I also thank Anita Sellent for providing feedback on my scene flow chapter, and to Cäcilie Kovács, whose help in translating my abstract into German was invaluable in keeping my broken German off the printed page.

A number of my close friends and colleagues deserve special mention as well, owing in no small measure to what they put up with in the weeks (and months) leading up to my deadlines. In Vienna, no such list could be complete without mention of Milene Pacheco, Márton János, Thomas Jackson, Emina Alić, Stefan Schlaffer, Nicole Brosch, Tara Akhavan, and Ji Ho Cho; during my 3 month stay at Carsten's lab at TU Dresden, Eric Brachmann, Alexander Krull, and Frank Michel made a formidable effort to make me feel part of the group, and among fellow visitors to the lab I had some great discussions with Hassan Abu Alhaija, Shuai (Kyle) Zheng, and Uwe Schmidt.

I am profoundly grateful to Wolfgang Wagner, professor of satellite remote sensing at TU Vienna, who allowed me to switch from an initial Ph.D. in remote sensing to one in computer vision, and then to carry on in remote sensing upon completion of my Ph.D.

Finally, to my parents Peter and Jana, whose love and support long predates these three years I devoted to my Ph.D., I owe whatever drive or ability to achieve something that I may have the good fortune to possess.

Abstract

The problem of *correspondence search* (or *matching*) remains a central focus of research in a variety of domains of computer vision. To date, the overwhelming tendency in the literature has been to rely on either the assumption of *brightness constancy*, or on the assumption of patchwise *local surface planarity*. However, among aspects of matching that render the task challenging *as displacements become large*, there figure the tendency for these two assumptions to ultimately break down, together with the *combinatorial explosion* of the resulting space of possible motions, or *labels*. In this thesis, we tackle three domains that call for performing matching at large displacements: (i) depth super resolution (SR), (ii) RGB-D scene flow, and (iii) optical flow, whereby we carry out correspondence search (i) within a single depth map, (ii) across a pair of RGB-D frames, and (iii) across a pair of color images, respectively. For optical flow and RGB-D scene flow, we relax reliance on brightness constancy by focusing attention in the available color images on image gradients. Recognizing for all three that 2D motion in the image plane is ultimately a function of 3D motion in the underlying scene, we *overparameterize* the 2D motions that patches of pixels are allowed to undergo in the image plane in terms of 3D rigid body motions applied to patches of 3D *points*: a *geometrically motivated* form of correspondence search. For depth SR and RGB-D scene flow, we apply these 3D rigid body motions directly to patches of 3D points encoded in the available depth maps, thereby *overcoming the assumption of local surface planarity*; for optical flow, we recover 3D points for all pixels of a patch of pixels by intersecting respective pixel viewing rays with a 3D plane, and have the resulting 3D points undergo a 3D rigid body motion. In the former case, we accordingly seek—individually for *each pixel*—a 6 degree of freedom (DoF) 3D rigid body motion describing the motion of the patch of points associated with the pixel; in the latter, we similarly assign to each pixel a 3 DoF 3D plane in addition to a 6 DoF rigid body motion, for a total of 9 DoF. In both cases, even a coarse discretization of these *high-dimensional*, *continuous label spaces* would lead to spaces of a daunting number of labels to consider exhaustively as displacements become large, a problem we succeed in overcoming by instead calling on variants of the simple yet effective PatchMatch correspondence search algorithm to *grow and refine* sparse correspondence seeds in a manner tailored to our 6 DoF and 9 DoF motion models.

Kurzfassung

Das Problem der *Korrespondenzsuche* (oder des *Matchings*) stellt einen sehr aktiven Forschungsschwerpunkt in mehreren Teilgebieten des Bereichs Computer Vision dar. Bislang hat man sich in der Literatur hauptsächlich auf die Annahme der *Brightness Constancy* oder der *lokalen Oberflächenplanarität* gestützt. *Werden die Abstände größer*, wird die Korrespondenzsuche jedoch zu einer größeren Herausforderung, da man sich nicht mehr auf diese zwei Annahmen verlassen kann, und auch aufgrund der *kombinatorischen Explosion* des Raumes der möglichen Bewegungen, oder *Labels*. In dieser Dissertation beschäftigen wir uns mit drei Bereichen, wo Matching entlang größerer Abstände benötigt wird: (i) Depth Super Resolution (SR), (ii) RGB-D Scene Flow, und (iii) Optical Flow. In diesen Bereichen wird das Matching (i) innerhalb einer einzigen Tiefenkarte, (ii) zwischen zwei RGB-D Bildern, und (iii) zwischen zwei Farbbildern durchgeführt. Für Optical Flow und RGB-D Scene Flow lockern wir die Annahme der Brightness Constancy, indem wir unsere Aufmerksamkeit in den vorhandenen Farbbildern auf die Gradienten des Bildes richten. Nachdem wir für alle drei Bereiche erkennen, dass eine 2D-Bewegung in der Bildebene letztendlich eine Funktion einer 3D-Bewegung in der Szene ist, *überparametrisieren* wir die erlaubten 2D-Bewegungen mittels 3D-Starrkörperbewegungen von Patches von 3D-*Punkten*: eine *geometrisch motivierte* Art der Korrespondenzsuche. Für Depth SR und RGB-D Scene Flow, wenden wir diese 3D-Starrkörperbewegungen direkt auf Patches der 3D-Punkte an, die in den vorhandenen Tiefenkarten kodiert sind, wodurch wir *die Annahme der lokalen Oberflächenplanarität überwinden*; für Optical Flow stellen wir 3D-Punkte für alle Pixel eines Patches wieder her, indem wir die jeweiligen Pixel-Sehstrahlen mit einer 3D-Ebene schneiden und unterziehen die daraus entstandenen 3D-Punkte einer 3D-Starrkörperbewegung. Im ersten Fall suchen wir—einzeln für *jedes Pixel*—eine 3D-Starrkörperbewegung mit 6 Freiheitsgraden (Engl. *degrees of freedom*, oder DoF), die die Bewegung des mit dem jeweiligen Pixel verbundenen Patches von Punkten beschreibt; im zweiten Fall schreiben wir auf ähnliche Weise zusätzlich zu der Starrkörperbewegung mit 6 DoF jedem Pixel eine 3D-Ebene mit 3 DoF zu. In beiden Fällen hätte man bei größeren Abständen sogar durch eine grobe Diskretisierung dieser *hochdimensionalen*, *kontinuierlichen* Labelräume eine Anzahl von Labels, die zu groß ist, um sie vollständig berücksichtigen zu können. Dieses Problem gelingt es uns zu bewältigen, indem wir uns stattdessen des PatchMatch Korrespondenzsuchalgorithmus bedienen, um spärliche Korrespondenzen entsprechend unseren 6 DoF- und 9 DoF-Bewegungsmodellen zu propagieren und zu verfeinern.

List of Figures

List of Tables

Acronyms

AAE	Average Angular Error
ANN	Approximate Nearest Neighbor
ASIFT	Affine-SIFT
CCD	Charge-Coupled Device
CRF	Conditional Random Field
DoF	Degree(s) of Freedom
DoG	Difference of Gaussians
EPE	End Point Error
FP	Fronto-Parallel
FPS	Frames Per Second
GT	Ground Truth
ICP	Iterative Closest Point
MAP	Maximum A Posteriori
MCMC	Markov Chain Monte Carlo
MRF	Markov Random Field
NN	Nearest Neighbor
PCA	Principal Component Analysis
PBP	Particle Belief Propagation
PMBP	PatchMatch Belief Propagation
RMSE	Root Mean Squared Error

RGB-D Red, Green, Blue, Depth

SIFT Scale Invariant Feature Transform

SR Super Resolution, Super Resolved

SSD Sum of Squared Differences

SURF Speeded Up Robust Features

SVD Singular Value Decomposition

ToF Time of Flight

Contents

CHAPTER 1

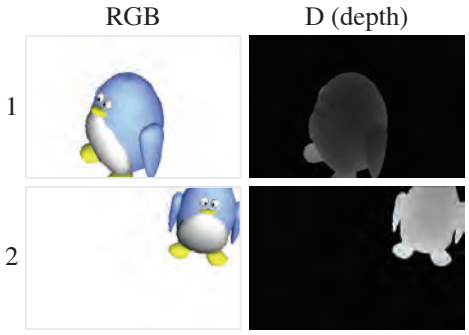
Introduction

1.1 Motivation

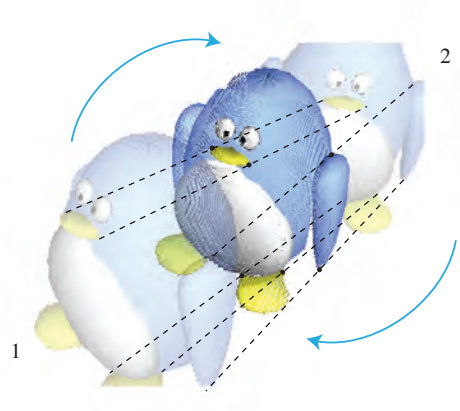
In his rather memorable foreword to Hartley and Zisserman [HZ06], Olivier Faugeras notes that "making computers see" was thought in the 1960s by leading experts in the field of artificial intelligence to be at the level of difficulty of a summer student's project, a point of view Faugeras proceeds to attribute to the problem of naïve introspection. Admittedly, matching a *patch* of pixels in one image of a sequence to its *correspondence* in a second image—in essence, the very subject of this thesis—itself perhaps seems, at least in principle, a task fit for a small child. Appearances aside, the problem of *correspondence search* (or *matching*) remains a central focus of research in several domains of the field of computer vision; examples we address ourselves in the pages of this thesis involve (i) depth super resolution (SR), (ii) RGB-D[1] scene flow, and (iii) optical flow in Chapters 5, 6, and 7, respectively. An illustrative example application of dense matching is provided in Figure 1.1, where an *intermediate frame* is interpolated using the dense matches obtained between a pair of RGB-D frames of a moving scene using our RGB-D scene flow approach from Chapter 6.

The overwhelming tendency in the literature on dense matching has been to rely on either the assumption of *brightness constancy* (cf. Figure 1.2)—which is to say the assumption that corresponding pixels share the same brightness—or on the assumption of patchwise *local surface planarity* (cf. Figure 1.4a) implicit in all traditional patch motion models. In addition to the perennial difficulty of seeking correspondences for inadequately discriminative patches—as is the case, e.g., for patches over untextured surfaces—aspects of matching that render the task challenging *as displacements between correspondences become large* include the ultimate breakdown of the classical assumptions of brightness constancy and local surface planarity, and the explosion of the *space of possible motions*. In the remainder of this section, we outline the

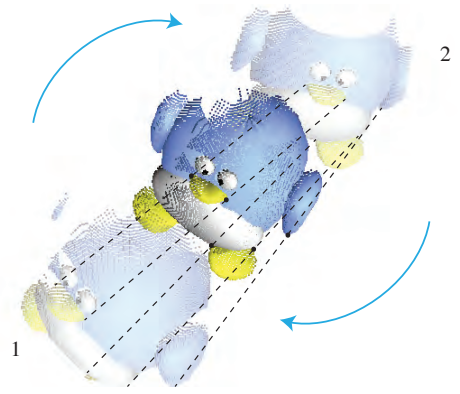
[1] An *RGB-D* image is merely a conventional color image provided in conjunction with a corresponding depth map, as illustrated in Figure 1.1a. The role of this *depth map* is to map each pixel $\mathbf{x} = (x, y)^\top$ in the image to the depth Z of the 3D point $\mathbf{X} = (X, Y, Z)^\top$ that projects to $\mathbf{x}$, respectively.



(a) Input pair of RGB-D frames.



(b) Original viewpoint.



(c) Novel viewpoint.

Figure 1.1: Large displacement view interpolation from dense matches between two RGB-D frames—visualized at original and novel viewpoint, in order to emphasize the availability of 3D *points*—recovered in both directions simultaneously (frame 1 to frame 2, frame 2 to frame 1) using our RGB-D scene flow approach (cf. Chapter 6). Points of the two input RGB-D frames are shown with transparency for reference. Original penguin mesh model from `http://www.3dxtras.com/3dxtras-free-3d-models-details.asp?prodid=5568`.

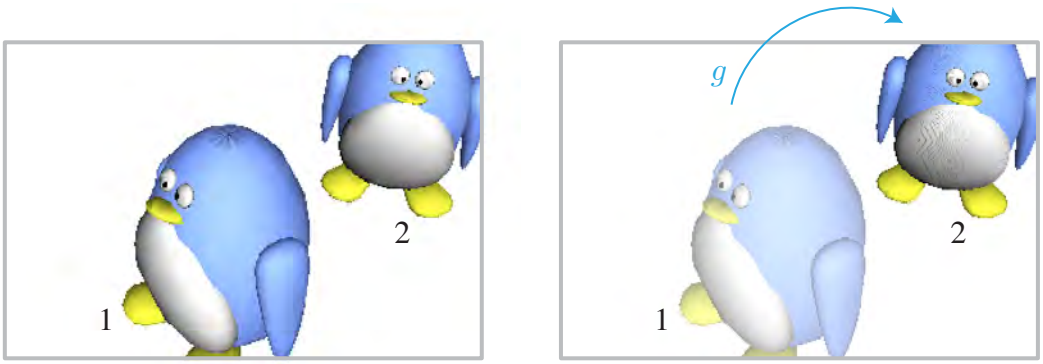
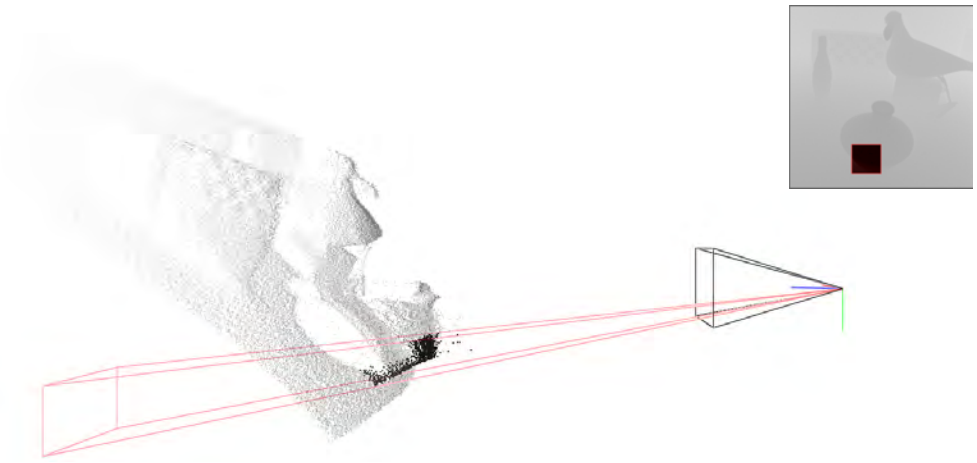


Figure 1.2: Brightness constancy assumption. The tendency of the assumption of brightness constancy to break down as displacements become large is especially easy to see in the presence of rotation, whether due to rotational motion of the scene relative to the camera or—unless the scene is genuinely *Lambertian* [Ang06], whereby shading is invariant to the pose of the camera—to that of the camera relative to the scene. Intuitively, differences in shading are due to the further penguin being illuminated frontally, while the closer one is illuminated from the side (figure best viewed in the electronic version of this thesis).

elements of a typical dense matching pipeline—encompassing the definition of what is meant by *patches*, the *model* according to which a patch may be understood to undergo motion, the *energy* that is being minimized, and the *optimization* according to which the minimization is performed—and how we in this thesis address the problems of dense matching that arise as displacements become large.

Patches. The traditional form of a patch associated with a pixel $\mathbf{x} = (x, y)^\top$ is that of an $n \times n$ square of pixels in image space centered on $\mathbf{x}$. An immediate advantage of such a notion of a patch is the ease of its implementation; a glaring disadvantage is that the pixels of such a patch may well straddle an object boundary, as illustrated in Figure 1.3a, thereby calling for some form of special handling (e.g., adaptive support weighting [YK05]). While smaller patches are less prone to this problem of straddling object boundaries (indeed, a patch that consists only of the pixel $\mathbf{x}$ itself is free of this problem altogether), smaller patches may be less discriminative (more ambiguous) than larger ones. We suggest that in the presence of dense depth data (i.e., depth maps), a fruitful alternative to traditional square patches of *pixels* is to reason instead in terms of the *points* encoded in the depth map at hand that lie inside a *sphere* centered on the point encoded at $\mathbf{x}$. We reason in terms of such patches in Chapters 5 and 6, with the advantages of jointly ameliorating the problem of straddling object boundaries and of providing additional robustness to shot noise, as we illustrate in Figure 1.3b.

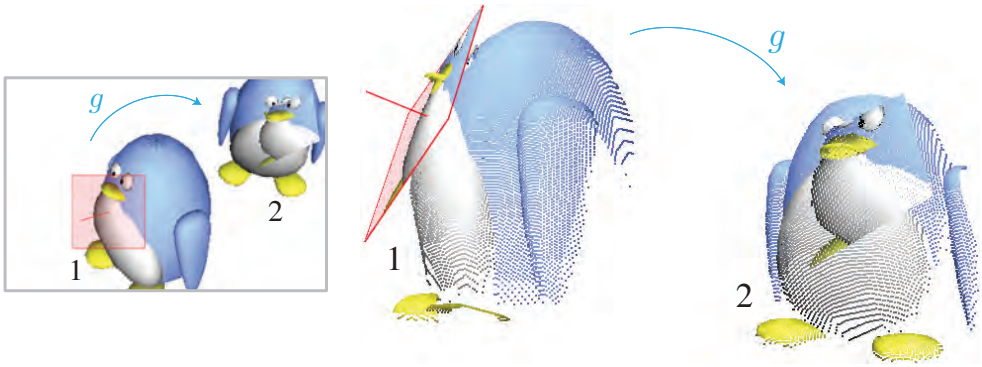
Motion Model. The model according to which the pixels (or points) that constitute a patch undergo motion can take any of a number of forms. A hierarchy of image space motion models that can be described in terms of 2D *homographies* is presented in Section 2.2, beginning with 2 degree of freedom (DoF) translations and ending with 8 DoF projectivities at the most general.



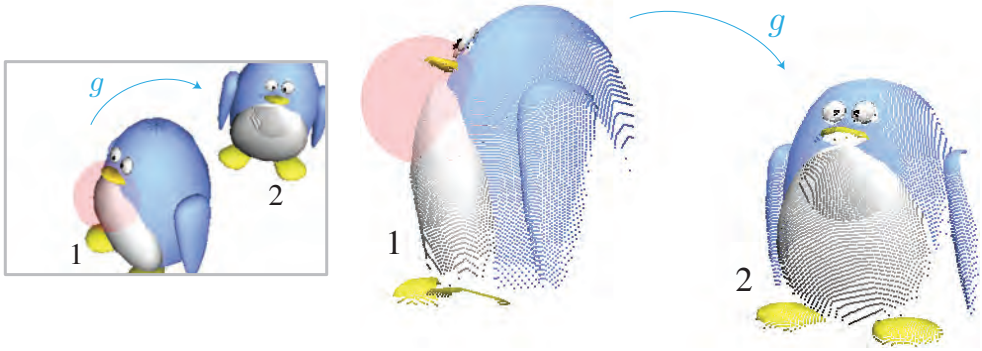
(a) The points that constitute a traditional $n \times n$ image space patch centered on a pixel $\mathbf{x}$, comprising all points encoded in the depth map that lie inside the unbounded pyramid determined by the camera center as apex and the respective viewing rays corresponding to the four corners of the patch.

(b) The points encoded in the depth map that—in our manner—constitute a patch as the inliers of a *sphere*, centered on the point encoded in the depth map at the same pixel $\mathbf{x}$. Note that the sphere radius was chosen in order to match the pixel extent in image space of the $n \times n$ patch above (cf. Section 6.2).

Figure 1.3: Patches as pyramids contrasted with patches as spheres, again in the presence of dense depth data. (a) While identifying the points encoded in the depth map that constitute a patch in terms of the pixels occupied by a traditional $n \times n$ image space patch has the advantage of being easy to implement, a glaring disadvantage of such patches is that they occupy all points inside the unbounded pyramid determined by the camera center (cf. Section 2.4) and the four corners of the patch, with the problem of including points straddling object boundaries and points corrupted by noise. (b) Our manner of proceeding instead to identify the points encoded in the depth map that constitute a patch as the inliers of a sphere has the advantage of jointly ameliorating the problem of straddling object boundaries and of providing robustness to shot noise. Time of flight (ToF) depth map provided by Oisin Mac Aodha.



(a) Matching under the assumption of local surface planarity, with motion in the image plane expressed in terms of a homography parameterized as a 3 DoF 3D plane undergoing a 6 DoF 3D rigid body motion g, for a total of 9 DoF.



(b) Matching with respect to the 3D points encoded in the available depth maps, with the points that constitute a patch—here comprising, in our manner, the inliers of a sphere—undergoing the same 6 DoF 3D rigid body motion g as above.

Figure 1.4: Relaxing the local surface planarity assumption, in the presence of dense depth data. (a) While at small displacement even a traditional $n \times n$ patch undergoing strictly *translational* motion in image space can be expected to capture a correspondence, as displacements become large even plane-induced homographies (cf. Section 2.6) may ultimately fail. (b) In addition to allowing for larger-displacement matching over non-planar geometry than is possible in general on the assumption of local surface planarity, reasoning in terms of 3D rigid body motions applied directly to the available 3D points additionally allows for optimizing over fewer degrees of freedom and allows for comparing point similarity in terms of Euclidean distances in world space. The value of identifying the points that constitute a patch of points as the inliers of a sphere is illustrated in Figure 1.3.

With 9 DoF, a homography can be *overparameterized* to model a 3 DoF 3D plane undergoing a 6 DoF 3D rigid body motion (cf. Figure 1.4a and Section 2.6), a parameterization turn to in Chapter 7 to compute optical flow from a pair of RGB frames. As 2D motion in the image plane is ultimately a function of 3D motion in the underlying scene, an interesting attribute of such a parameterization is that it allows for matching patches of pixels in image space by reasoning in terms of possible underlying 3D rigid motion in world space. In Chapters 5 and 6, we likewise reason in terms of 6 DoF 3D rigid body motions, with the difference that we apply such motions directly to points encoded in the available depth maps (cf. Figure 1.4b); in addition to allowing for larger displacements over geometry that is not planar in the extent of the patch, proceeding in such a manner has the advantage of allowing for optimizing over fewer degrees of freedom and for evaluating *point similarity by computing Euclidean distances* directly in world space.

Energy. A fruitful way of thinking about correspondence search is in terms of energy minimization over a label space, where a *label* is understood to *index* a possible motion derived from a chosen motion model (indeed, we shall often use the terms *motion* and *label* interchangeably in the pages of this thesis). Given a patch associated with a pixel $\mathbf{x}$, the 'goodness' of the match specified by a particular label is computed by evaluating some form of *matching cost* function over the pixels (or points) of the patch with respect to the motion indexed by the label. Such a function is called a *unary potential* since it is a function of only a single label, and the *energy* of a labeling is computed by summing over the respective unary potentials of each pixel in the reference image. In order to promote smoothness of the labeling, the energy might additionally encompass higher-order terms such as *pairwise potentials* that are a function of the labels assigned to neighboring pixels. We rely heavily on image gradients in Chapters 6 and 7 in the aim of ameliorating the problem of the assumption of brightness constancy breaking down as displacements become large.

Optimization. Taking into account even only 2 DoF translational motions, in order to tackle large displacements even an unacceptably coarse discretization of the label space may result in a prohibitively large number of possible labels to consider exhaustively for dense matching. Consider a source image with a spatial resolution of $w \times h$ pixels, such that any pixel $\mathbf{x}$ is able to undergo a 2 DoF motion to any of the $w \times h$ pixels of a target image (thus giving a total of $w \times h$ labels), where w and h denote image width and height, respectively. Taking into account unary terms alone, the logarithmic complexity of exhaustive 2 DoF matching for all $w \times h$ pixels of the source image is $O\big((h \times w)^2\big)$; taking into account higher-order interactions, the worst-case complexity of exhaustive matching becomes $O\big((h \times w)^{h \times w}\big)$. On the example of a moderate spatial resolution of 320×480, dense exhaustive matching in the former setting calls for evaluating a total of 23,592,960,000 candidate configurations; in the latter, dense exhaustive matching calls for evaluating a number of candidate configurations that *exceeds the estimated number of atoms in the observable universe*.[2] For large enough w and h, the daunting size of the space of possible configurations of motions for even a pixel-accurate discretization of the 2 DoF space considered above is only intensified in the context of our continuous 6 DoF and 9 DoF

[2] An estimate of the number of atoms in the observable universe is set between 10^{78} and 10^{82} at `http://www.universetoday.com/36302/atoms-in-the-universe/`.

settings. In order to keep this problem of dense matching in high-dimensional, continuous label spaces tractable, our strategy is to proceed to *grow and refine sparse seeds* in the manner of the PatchMatch correspondence search algorithm [BSFG09, BSGF10], which is to say by means of a combination of tailored *spatial propagation* (cf. Figure 1.5) and *randomization* (a form of *resampling*, to use the language of Chapter 3) applied to initial sparse correspondence seeds.

1.2 Contributions

Our **main contribution** is to show that variants of PatchMatch can be well-suited to obtaining dense correspondence fields in our *continuous*, *high-dimensional* label spaces that arise from *geometrically motivated* reasoning, understood to mean casting the motion of pixels in terms of the motion of points—whether encoded in a depth map or parameterized by a 3D plane—undergoing 3D *rigid body* motions. To be able to reason in such a manner is attractive since 2D motion in the image plane is ultimately a function of 3D motion in the scene. In applying 3D rigid body motions directly to patches of 3D points encoded in the available depth maps, we succeed in *overcoming the local surface planarity assumption*. A list of secondary contributions per domain—depth SR, RGB-D scene flow, and optical flow—is provided below.

Depth SR. We present a depth SR method that differs from all previous depth SR methods in that we make *no use of ancillary data* like a color image at the target resolution, multiple aligned depth maps, or a database of high-resolution depth patches. Instead, we reason only in terms of matches across depth between patches of the points encoded in the single input depth map, consisting of the inliers of spheres and undergoing respective 6 DoF 3D rigid body motions. We show that our results are highly competitive with those of alternative techniques that do leverage ancillary data.

RGB-D Scene Flow. Building upon the spherical patches introduced in our work on depth SR, we are able to show attractive scene flow results on challenging synthetic and real-world scenes that push the practical limits of the assumptions of brightness constancy and local surface planarity. An important novelty over the spherical patches of our depth SR work is to reason not in terms of a single fixed sphere radius r, but in terms of adaptive radii $r_{\mathbf{x}}$ at each pixel $\mathbf{x}$; proceeding in this manner allows for ameliorating the problem of sphere inlier counts varying as a function of sphere depth, thereby in turn allowing for a more uniform matching quality than is possible for fixed r. Additionally, as a consequence of our approach our output is a dense field of 6 DoF 3D rigid body motions, in contrast to the 3D translation vectors that are the norm in scene flow.

Optical Flow. We show that a variant of PatchMatch presents itself as an effective optimizer for what is ostensibly an extreme *overparameterization* of 2 DoF optical flow in terms of 9 DoF plane-induced homographies, such that each pixel be assigned a 3 DoF 3D plane and a 6 DoF 3D rigid body motion that the plane undergoes. Proceeding in this manner has the advantage of reasoning about matching in a pair of RGB frames in terms of possible 3D motions, recognizing that 2D motion in the image plane is ultimately a function of 3D motion in the scene.

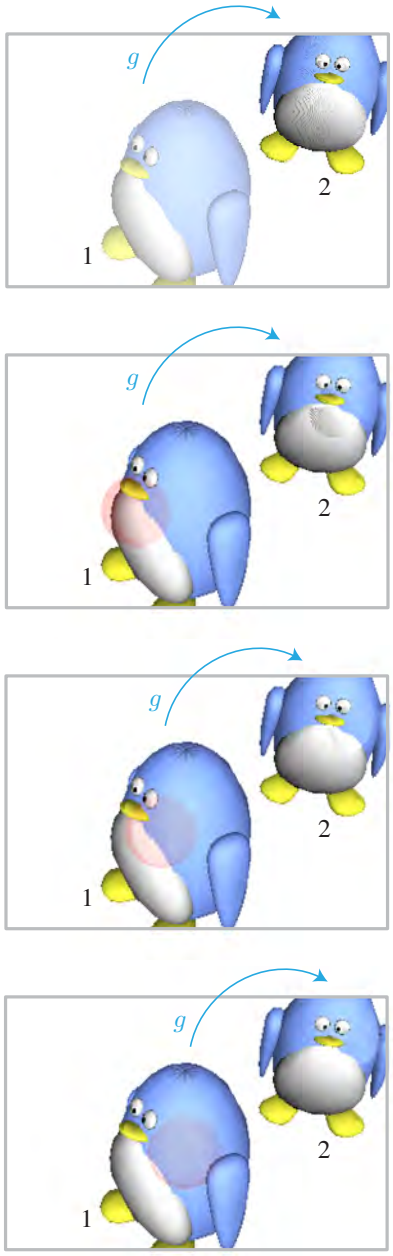


Figure 1.5: Geometric rationale of PatchMatch spatial propagation with respect to 6 DoF 3D rigid body motions. PatchMatch spatial propagation adapted to 6 DoF rigid body motions consists in considering the adoption, at a pixel $\mathbf{x}$, of the respective motions presently assigned to pixel *neighbors* of $\mathbf{x}$. To proceed in this manner is sensible because if two instances of an object are related (locally) by a 3D rigid body motion g, then so too is any (local) pair of correspondences related by the very same motion g. Note that on the example of the rigidly moving penguin above, *each pair of corresponding patches of 3D points is related by the same 6 DoF 3D rigid body motion g that describes the 3D rigid motion of the penguin itself.*

1.3 Resulting Publications

The following peer-reviewed publications [HRGR13, HFR14, HBK+14] resulted from the work presented in this dissertation, all three of which were accepted to top-tier computer vision venues:

(i) **Michael Hornáček**, Christoph Rhemann, Margrit Gelautz, and Carsten Rother. Depth Super Resolution by Rigid Body Self-Similarity in 3D. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1123–1130, 2013.

(ii) **Michael Hornáček**, Andrew Fitzgibbon, and Carsten Rother. SphereFlow: 6 DoF Scene Flow from RGB-D Pairs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3526–3533, 2014.

(iii) **Michael Hornáček**, Frederic Besse, Jan Kautz, Andrew Fitzgibbon, and Carsten Rother. Highly Overparameterized Optical Flow Using PatchMatch Belief Propagation. In *European Conference on Computer Vision*, pages 220–234, 2014.

Additionally, paper (ii) was accepted in modified form under the name *Locally Rigid RGB-D Scene Flow* as an out-of-proceedings peer-reviewed workshop paper (with oral) at the 2014 Computer Vision Winter Workshop (CVWW) held in Křtiny, Czech Republic.

Note on Authorship. Chapters 5, 6, and 7 are taken largely from papers (i), (ii), and (iii), respectively, with the addition of the newly added Sections 5.4, 6.4, and 7.4 intended to reveal insights on algorithm performance that did not make it into the respective published papers. Appendix A is an adaptation of supplementary material provided with paper (ii). Note additionally that an earlier version of paper (iii) had been submitted elsewhere but subsequently withdrawn, with Frederic Besse as first author and myself as second. My contribution to the paper was primarily to integrate geometrically motivated reasoning, including the two geometrically motivated initialization modes, geometrically motivated refinement, view propagation, and the requirement that normals point to the camera that led to improving the results. The question of the value of these additions is taken up in Section 7.4, as exemplified on three image pairs by showing the effect of rolling back these changes one at a time to the state of the algorithm prior to my involvement.

1.4 Thesis Organization

The seven chapters (and single appendix) that remain of this dissertation are organized in the manner outlined below:

Chapter 2. The chapter on *geometric foundations* aims to explain concepts from geometry that are central to understanding the chapters ahead. These include a discussion of introductory projective geometry, 2D homographies, 3D rigid body motions, the finite projective camera, epipolar geometry, and plane-induced homographies, which we derive in a manner that confirms

the intuition that fixed plane, moving camera and fixed camera with plane undergoing the inverse motion give rise to the *same plane-induced homography*, a fact called upon in Chapter 7. The discussion draws mainly on Hartley and Zisserman [HZ06], and to a lesser extent on Ma et al. [MSKS03] and on the highly readable tutorial introduction to projective geometry for computer vision of Birchfield [Bir98].

Chapter 3. In the chapter on *MAP inference using PatchMatch and PatchMatch Belief Propagation (PMBP)*, our aim is to derive the energy minimization frameworks—which is to say PatchMatch and PMBP—that we call on to carry out dense matching in the chapters that follow in terms of MAP inference by means of message passing. The chapter begins with a review of basic concepts from probability theory and probabilistic graphical models, proceeds to introduce message passing on trees, and then addresses message passing for graphs that contain loops. PMBP and PatchMatch are finally derived, similarly to Besse et al. [BRFK12], in terms of a flavor of message passing for loopy graphs tailored to MAP inference in continuous label spaces. We draw mainly on Bishop [Bis06] and Nowozin and Lampert [NL11] for the sections leading up to (and including) message passing for loopy graphs.

Chapter 4. The *literature review* is intended to outline the major trends across the field of computer vision in correspondence search. We consider methods for sparse matching, optical flow, stereo, and scene flow. A general reference for this chapter was the textbook of Szeliski [Sze11]. Section 4.2 on optical flow draws in particular on Fleet and Weiss [FW06]. We defer an overview of the literature on depth SR to Chapter 5, since only a subset of depth SR techniques involve matching.

Chapter 5. The chapter on *depth super resolution* (SR) relates our first paper, which attempts to address the question of just how far one can push jointly increasing the spatial resolution and apparent measurement accuracy of an input low-resolution, noisy, and perhaps heavily quantized depth map, using only the information available in the input depth map itself. Accordingly, our method is different from all previous depth SR techniques in that we make no use of ancillary data like a color image at the target resolution, multiple aligned depth maps, or a database of high-resolution depth patches. Inspired by the 'single image' SR approach of Glasner et al. [GBI09] for super resolving a color image by matching patches of pixels across *scale*, we instead proceed by merging patches—matched across *depth* under respective 6 DoF 3D rigid body motions—of the 3D points encoded in the depth map, understanding a patch to comprise the points that lie within a radius of a center point encoded at some pixel. Notably, we show that our results are highly competitive with those of alternative techniques that do leverage ancillary data. We add detail not included in the paper owing to paucity of space, in particular a far more thorough explanation of the patch upscaling and merging step.

Chapter 6. The chapter on *RGB-D scene flow* relates our second paper, where we expand upon the dense matching in terms of patches as the 3D points that are inliers of spheres that we introduced in our work described in the previous chapter, tailoring the matching to the problem of computing scene flow from a pair of RGB-D frames. As a consequence of our approach, our

output is a dense field of 6 DoF 3D rigid body motions, in contrast to the 3D translation vectors that are the norm in the literature on scene flow. Reasoning in our manner additionally allows for us to carry out occlusion handling using a 6 DoF consistency check for the flow computed in both directions (frame 1 to frame 2, frame 2 to frame 1), and to promote smoothness of the flow fields using an intuitive 6 DoF local rigidity prior. We show attractive flow results on challenging synthetic and real-world scenes that push the practical limits of the assumptions of brightness constancy and local surface planarity.

Chapter 7. The chapter on *highly overparameterized optical flow* relates the third and final paper, which computes optical flow from an RGB image pair in a manner that attempts to reason in terms of the 3D motion in the scene that gives rise to the 2D motion in the image plane. Accordingly, we propose to compute 2 DoF optical flow using what is ostensibly an extreme overparameterization: depth, surface normal, and frame-to-frame 6 DoF 3D rigid body motion at every pixel, giving a total of 9 DoF. The advantages of such an overparameterization are twofold: first, geometrically meaningful reasoning can be called upon in the optimization, reflecting possible 3D motion in the underlying scene; second, the 'fronto-parallel' assumption implicit in traditional pixel window-based matching is ameliorated because the parameterization determines a plane-induced homography at every pixel. We show that the resulting flow fields compare favorably to the state of the art on a number of small- and large-displacement datasets, giving especially competitive results as displacements become large.

Chapter 8. The final chapter contains the *concluding remarks* of this thesis. Additionally, the chapter provides a list of suggestions for directions for future work.

Appendix A. The single appendix gives a derivation of the relationship used extensively in Chapter 6 relating the Euclidean distance between two points both situated at equal depth and that depth when the two points project to neighboring pixels. The appendix was included in the corresponding paper's supplementary material.

1.5 Notational Conventions

Throughout this thesis, we attempt to remain consistent with the familiar notational conventions of Hartley and Zisserman [HZ06]. Accordingly, scalars x are expressed in italics, vectors $\mathbf{x}$ in bold, and matrices $\mathtt{K}$ in typewriter font. Additionally, vectors are by default understood to be column vectors, and so are written out as $(x_1, \ldots, x_n)^\top$. In Euclidean space, we express 2D points $\mathbf{x} = (x, y)^\top \in \mathbb{R}^2$ using small letters, and 3D points $\mathbf{X} = (X, Y, Z)^\top \in \mathbb{R}^3$ using capitals; we express their homogeneous counterparts as $\tilde{\mathbf{x}} \sim (x, y, 1)^\top \in \mathbb{P}^2$ and $\tilde{\mathbf{X}} \sim (X, Y, Z, 1)^\top \in \mathbb{P}^3$, respectively. Where we depart from their notational conventions is that we place a tilde on vectors $\tilde{\mathbf{x}}$ that express the homogeneous coordinates of a point, omitting a tilde on vectors that express a point's Euclidean coordinates, which is the opposite of Hartley and Zisserman's convention. In Sections 3.2 and 3.3 on probabilisitic graphical models and inference by message passing on trees, respectively, we try to remain consistent with the notation of Nowozin and Lampert [NL11]. Section 4.2 on optical flow attempts to remain broadly consistent with

the notation commonly used in the optical flow literature in referring to the image intensity at a pixel $\mathbf{x} = (x, y)^\top$ in an image acquired at time step t as $I(x, y, t)$, and to its correspondence in an image acquired at time step $t + \Delta t$ as $I(x + u_{\mathbf{x}}, y + v_{\mathbf{x}}, t + \Delta t)$, with $(u_{\mathbf{x}}, v_{\mathbf{x}})^\top$ the 2D flow vector originating at $\mathbf{x}$ that relates the two.

CHAPTER 2

Geometric Foundations

Recognizing that 2D motion in the image plane is ultimately a function of 3D motion in the underlying scene, we proceed to *overparameterize* the 2D motions that candidate matches are allowed to undergo in the image plane in terms of 3D rigid body motions applied to *patches of 3D points*, the sense in which we qualify our matching as *geometrically motivated*. This chapter is accordingly of necessity in large part about the *finite projective camera*, a generalization of the *pinhole camera* model that models central projection using projective geometry, *finite* in the sense that the center of projection be modeled to lie at a finite distance from the image plane. The pinhole camera model describes the projection of a 3D point as the intersection with the image plane of the ray passing from that point through an infinitesimal aperture (or 'pinhole'), in the manner of an idealized *camera obscura*. With this formalism in hand, we are able to reason about mapping a point to its projected pixel in the image plane, and a pixel back to the ray along which the projecting point is required to lie. The motion of 2D pixels can then be described in terms of the motion of 3D points, which throughout this thesis we model in terms of 3D rigid body motions.

We begin this chapter in Section 2.1 with an introduction to homogeneous coordinates, a generalization of Euclidean coordinates that will allow us to cast the 3D rigid motion of points and their projection to the image plane—both non-linear transformations over points expressed in Euclidean coordinates—as linear transformations, in turn conveniently allowing for reasoning about such transformations using matrix algebra. Next, we discuss projective transformations in 2D and 3D, focusing our attention in Section 2.2 on the classical motion models defined in terms of *homographies* over pixels expressed as points in projective 2-space, and in Section 2.3 on the rigid body motions over projective 3-space that underlie our reasoning about motion. We then proceed to detail the finite projective camera in Section 2.4. Epipolar geometry and the constraints on matching pixel correspondences between a pair of views is treated briefly in Section 2.5. Finally, in Section 2.6, we address the action of 3D rigid motion on points and planes given a pair of cameras, by means of plane-induced homographies.

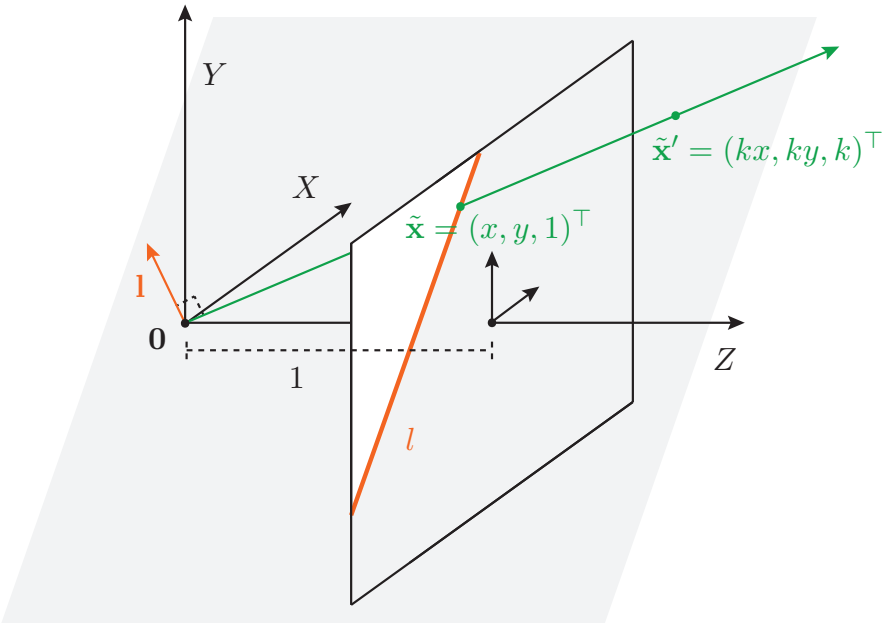


Figure 2.1: The projective plane $\mathbb{P}^2$. The points $\tilde{\mathbf{x}}, \tilde{\mathbf{x}}' \in \mathbb{P}^2$ in projective 2-space both express the same point $(x, y)^\top \in \mathbb{R}^2$ in Euclidean 2-space. The line $\mathbf{l} \in \mathbb{P}^2$ expresses the same line as $l \subset \mathbb{R}^2$, which is obtained by the intersection of $Z = 1$ with the plane through the origin $\mathbf{0}$ whose normal vector is $\mathbf{l}$, called the line's interpretation plane. Points and lines are thus expressed as the intersection of rays and planes with $Z = 1$.

2.1 Homogeneous Coordinates

Points in $\mathbb{P}^n$. A point in $\mathbb{R}^n$ is expressed as an n-dimensional vector $(x_1, \ldots, x_n)^\top$, the familiar Euclidean coordinates of the point. The same point can be expressed in *homogeneous* coordinates as any $(n+1)$-dimensional vector $(kx_1, \ldots, kx_n, k)^\top \in \mathbb{P}^n = \mathbb{R}^{n+1} \setminus \{\mathbf{0}\}, k \neq 0$. We indicate that two homogeneous vectors $\tilde{\mathbf{x}}, \tilde{\mathbf{x}}' \in \mathbb{P}^n$ are members of the same equivalence class by $\tilde{\mathbf{x}} \sim \tilde{\mathbf{x}}' \Leftrightarrow \exists k \neq 0 : \tilde{\mathbf{x}} = k\tilde{\mathbf{x}}'$ (read 'is proportional to'), which is to say they represent the same point. Note that for brevity, we shall often refer to these vectors simply as points, recognizing that they are in fact vectors that *represent* points. Given a point expressed in homogeneous coordinates $(kx_1, \ldots, kx_n, k)^\top \in \mathbb{P}^n, k \neq 0$, its analogue in Euclidean coordinates is $\mathbf{x} = (x_1/k, \ldots, x_n/k)^\top \in \mathbb{R}^n$. Note that any point in $\mathbb{R}^n$ can be expressed in $\mathbb{P}^n$.
Homogeneous vectors of $\mathbb{P}^2$ scaled such that $x_3 = 1$ lie in the plane $x_3 = 1$.[1] The plane $x_3 = 1$ can be thought of as an embedding of the Euclidean plane $\mathbb{R}^2$ (also known as Euclidean 2-space) in $\mathbb{P}^2$, obtained by the unit translation of the Euclidean plane $\mathbb{R}^2$ along the positive x_3-axis of the 3-dimensional Euclidean coordinate frame (cf. Figure 2.1). The vector space $\mathbb{P}^2$ is accordingly called the *projective plane* (or projective 2-space). Henceforth, we shall refer to points in $\mathbb{R}^2$ and $\mathbb{P}^2$ by $(x, y)^\top$ and $(x, y, 1)^\top$, respectively, rather than use the above subscript notation.

[1] In this thesis, we do not consider *points at infinity*, which are points $(x_1, \ldots, x_n, 0)^\top \in \mathbb{P}^n$.

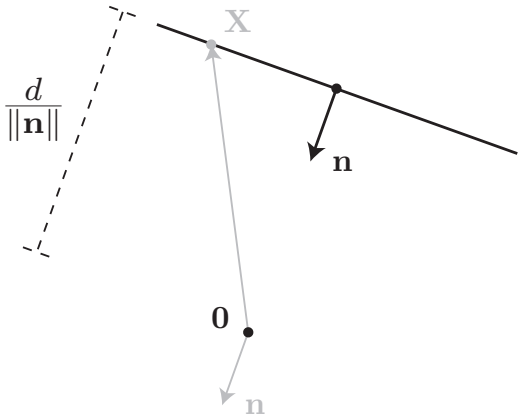


Figure 2.2: A plane $\boldsymbol{\pi} = (\mathbf{n}^\top, -d)^\top \in \mathbb{P}^3$. Any point $(\mathbf{X}^\top, 1)^\top \in \mathbb{P}^3$ that satisfies the equation $\mathbf{n}^\top\mathbf{X} - d = 0$ is in the plane $\boldsymbol{\pi}$. Note that the quantity $\mathbf{n}^\top\mathbf{X}$ happens to be the scalar projection of the *vector* $\mathbf{X}$ onto the vector $\mathbf{n}$ (both shown in gray), expressed in units of $\|\mathbf{n}\|$.

By analogy, $\mathbb{P}^3$ is referred to as projective 3-space, and we refer to points in $\mathbb{R}^3$ and $\mathbb{P}^3$ by $(X, Y, Z)^\top$ and $(X, Y, Z, 1)^\top$, respectively.

Lines in $\mathbb{P}^2$. Let us consider the familiar general form[2] equation of a line $l \subset \mathbb{R}^2$ in the Euclidean plane,

$$ax + by + c = 0. \tag{2.1}$$

Rewriting Equation (2.1) as the scalar product of two vectors,

$$(a, b, c)(x, y, 1)^\top = 0, \tag{2.2}$$

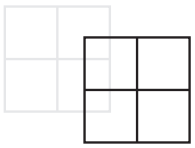
expresses an *incidence relation* between the homogeneous vector of a 2-dimensional point $\tilde{\mathbf{x}} = (x, y, 1)^\top \in \mathbb{P}^2$ and a second vector $\mathbf{l} = (a, b, c)^\top \in \mathbb{P}^2$, with two vectors qualified as *incident* if they are orthogonal. A geometric interpretation of this incidence relation is of $\mathbf{l}$ as the normal vector of a plane passing through the origin $\mathbf{0}$ of the coordinate frame, with $\tilde{\mathbf{x}}$ a homogeneous vector orthogonal to $\mathbf{l}$ and hence lying in the plane (cf. Figure 2.1). The intersection of this plane with $x_3 = 1$ gives l. Since scaling the vector $\mathbf{l}$ by a non-zero scalar has no effect on its incidence with $\tilde{\mathbf{x}}$, the vector $\mathbf{l}$ is itself homogeneous. Accordingly, we understand the vector $\mathbf{l} \in \mathbb{P}^2$ to *represent* the same line as $l \subset \mathbb{R}^2$.

Planes in $\mathbb{P}^3$. The analogue in projective 3-space of the incidence relation expressed in Equation (2.2) can be interpreted geometrically as one between a 3D point $(\mathbf{X}^\top, 1)^\top \in \mathbb{P}^3$ and a 3D *plane* $\boldsymbol{\pi} = (\mathbf{n}^\top, -d)^\top \in \mathbb{P}^3$,

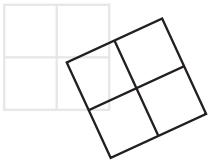
$$(\mathbf{n}^\top, -d)(\mathbf{X}^\top, 1)^\top = 0. \tag{2.3}$$

This immediately gives the familiar *Hessian normal form* $\mathbf{n}^\top\mathbf{X} - d = 0$ of the plane, where $\mathbf{n} \in \mathbb{R}^3$ denotes the plane's normal vector and d the distance in units of $\|\mathbf{n}\|$ from the origin of

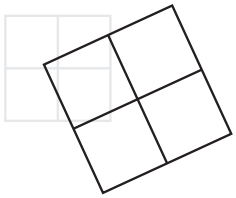
[2] In slope-intercept form, the same line is expressed as $y = -(a/b)x - c/b$.



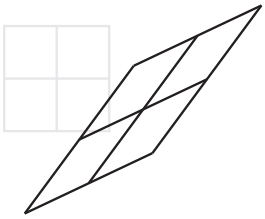
(a) Translation (2 DoF).



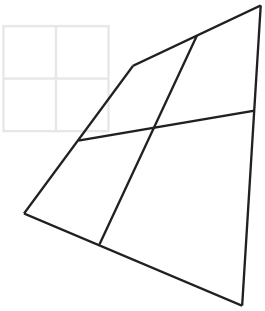
(b) Euclidean (3 DoF).



(c) Similarity (4 DoF).



(d) Affinity (6 DoF).



(e) Projectivity (8 DoF).

Figure 2.3: Different classes of homographies over $\mathbb{P}^2$. *Only those patches of an image whose motion can be explained by mappings of lines to lines can have their motion explained using a homography.* Note that it is in this sense that homographies over $\mathbb{P}^2$ implicitly invoke the assumption of local surface planarity.

the coordinate frame to the plane, as illustrated in Figure 2.2. Any point $\mathbf{X}' \in \mathbb{R}^3$ for which $\mathbf{n}^\top \mathbf{X}' - d = 0$ holds is likewise in the plane $\boldsymbol{\pi}$.

2.2 Homographies in 2D

A *homography* is defined as an invertible mapping $H : \mathbb{P}^2 \mapsto \mathbb{P}^2$ such that if three points $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \tilde{\mathbf{x}}_3 \in \mathbb{P}^2$ are collinear, then so too are the transformed points $H(\tilde{\mathbf{x}}_1), H(\tilde{\mathbf{x}}_2), H(\tilde{\mathbf{x}}_3)$. It follows immediately from the definition of a homography as a mapping of lines to lines in $\mathbb{P}^2$ that *only those patches of an image whose motion can be explained by mappings of lines to lines can have their motion explained using a homography* (cf. Figure 2.3). It can be shown that any homography H can be expressed in terms of a corresponding invertible 3×3 matrix $\mathtt{H}$. As a

consequence, if the points $\tilde{\mathbf{x}}_i$ are collinear with a line $\mathbf{l}$, then the transformed points $\tilde{\mathbf{x}}'_i = \mathtt{H}\tilde{\mathbf{x}}_i$ lie on precisely the line $\mathbf{l}' = \mathtt{H}^{-\top}\mathbf{l}$, since

$$\mathbf{l}'^{\top}\tilde{\mathbf{x}}'_i = (\mathtt{H}^{-\top}\mathbf{l})^{\top}\mathtt{H}\tilde{\mathbf{x}}_i = \mathbf{l}^{\top}\mathtt{H}^{-1}\mathtt{H}\tilde{\mathbf{x}}_i = \mathbf{l}^{\top}\tilde{\mathbf{x}}_i = 0, \tag{2.4}$$

thereby preserving point-line incidence. Note that the matrix $\mathtt{H}$ is itself homogeneous, since scaling $\mathtt{H}$ by a non-zero scalar amounts to scaling a vector to which $\mathtt{H}$ is applied by the same factor. A homography H at its most general can be described using 8 DoF, since 8 ratios determine the matrix $\mathtt{H}$ up to a non-zero scalar. We examine below a hierarchy of possible homographies, beginning with the *translations* as the most specialized that we consider and building up through the *Euclidean* motions, *similarities*, and *affinities* to the *projectivities*, which are the most general homographies.

Translations. The simplest class of homographies we consider are the 2D *translations*. A general translation is expressed, in matrix form, as a linear transformation over $\mathbb{P}^2$ is given by

$$\mathtt{H}_{\mathtt{T}} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathtt{I} & \mathbf{t} \\ \mathbf{0}^{\top} & 1 \end{bmatrix} \tag{2.5}$$

which encodes a translation by $\mathbf{t} = (t_x, t_y)^{\top}$. Translations are 2 DoF transformations.

Euclidean Motions. The 2D *Euclidean*[3] motions encompass planar rotations and translations. The form of the general matrix form of a rigid body motion is

$$\mathtt{H}_{\mathtt{E}} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^{\top} & 1 \end{bmatrix}, \tag{2.6}$$

where $\mathtt{R}$ is a 2×2 rotation matrix parameterized by θ and $\mathbf{t} \in \mathbb{R}^2$ a translation vector, for a total of 3 DoF. Note that rotation (about the origin of the coordinate frame) is carried out first, translation second.

Similarities. The 2D *similarities* encompass uniform scaling in addition to rotations, translations and reflections. The matrix form of a general similarity transformation is

$$\mathtt{H}_{\mathtt{S}} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ \mathbf{0}^{\top} & 1 \end{bmatrix}, \tag{2.7}$$

where $s \in \mathbb{R}$ is a scaling factor. Like general isometries, similarities have 4 DoF.

[3] Another name for a Euclidean motion is a *rigid body* motion. Note, however, that in order to avoid confusion, we shall in this thesis understand a rigid body motion to refer to a 6 DoF 3D rigid body motion, which we describe in Section 2.3, rather than to a 3 DoF 2D rigid body motion as described here.

Affinities. In addition to uniform scaling, rotations, translations and reflections, the 2D *affinities* additionally allow for non-uniform scaling, and are expressed in matrix form as

$$\mathtt{H}_{\mathrm{A}} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{2.8}$$

a 6 DoF transformation.

Projectivities. At their most general, homographies can also capture central projection and all compositions of projectivities. We have already seen that a general 2D *projectivity* is given by an arbitrary invertible 3×3 matrix; the matrix form differs from that of the affinities in that the last row is unrestricted:

$$\mathtt{H}_{\mathrm{P}} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ v_1 & v_2 & v \end{bmatrix} = \begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{v}^\top & v \end{bmatrix}, \tag{2.9}$$

and, as explained above, can be described using 8 DoF.

2.3 Rigid Body Motions in 3D

In 3D, our interest in projective transformations in this thesis lies squarely with the 3D analogue of the 2D Euclidean motions described in the previous section: 3D *rigid body* motions. We shall call on such motions to relate world and camera coordinate frames in Section 2.4, and again in Section 2.6 to explain the motion of points in 3D lying on a 3D plane, of which we later make use in Chapter 7. In Chapters 5 and 6 we use such motions to explain the motion of patches of 3D points identified as the inliers of *spheres* in 3D. Accordingly, we restrict our attention here to 3D rigid body motions and their attributes of interest for the chapters ahead. Let $g = (\mathtt{R}, \mathbf{t}) \in SE(3)$ be the notation used to denote a 3D rigid body motion, expressed as a linear transformation over $\mathbb{P}^3$ in the form of a 4×4 matrix as

$$\mathtt{T}_{\mathrm{E}} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{2.10}$$

where $\mathtt{R} \in SO(3)$ here denotes a 3 DoF 3×3 3D rotation matrix—encoding a 1 DoF angle of rotation per axis of the coordinate frame—and $\mathbf{t} \in \mathbb{R}^3$ a 3 DoF 3D translation vector, together giving a total of 6 DoF. Applying a point $\tilde{\mathbf{X}} = (\mathbf{X}^\top, 1)^\top \in \mathbb{P}^3$ to (2.10) transforms the point according to the mapping

$$\begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} \mathtt{R}\mathbf{X} + \mathbf{t} \\ 1 \end{pmatrix} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix}, \tag{2.11}$$

which, similarly as by (2.6), has the effect of first rotating $\mathbf{X}$ about the origin of the coordinate frame by $\mathtt{R}$ and then translating the resulting point by $\mathbf{t}$.

Inversion. Given a rigid body motion $g = (\mathtt{R}, \mathbf{t})$, the inverse rigid body motion g^{-1} is given in matrix form by

$$\begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathtt{R}^{-1} & -\mathtt{R}^{-1}\mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{2.12}$$

Accordingly, if $g = (\mathtt{R}, \mathbf{t})$ denotes a rigid body motion, we write $g^{-1} = (\mathtt{R}^{-1}, -\mathtt{R}^{-1}\mathbf{t})$ to denote the inverse motion.

Perturbation. In Chapters 5, 6, and 7, we shall want to try gently perturbing a given 3D rigid body motion $g = (\mathtt{R}, \mathbf{t})$ in order to determine whether doing so improves a patch matching cost. Suppose for illustration that g describes the motion of a plane passing through a point $\mathbf{X} \in \mathbb{R}^3$ whose normal vector is denoted by $\mathbf{n} \in \mathbb{R}^3$. Achieving the effect of perturbing $\mathtt{R}\mathbf{X} + \mathbf{t}$ by a translational displacement $\mathbf{t}'$ while keeping the transformed normal $\mathtt{R}\mathbf{n}$ fixed can be achieved by simply replacing g with $g' = (\mathtt{R}, \mathbf{t} + \mathbf{t}')$. However, achieving the effect of perturbing the normal direction by a rotational displacement $\mathtt{R}'$ while keeping $\mathtt{R}\mathbf{X} + \mathbf{t}$ fixed is more tricky. One way to correctly achieve the effect is by computing the perturbed motion g' according to the matrix concatenation $\mathtt{T}_3\mathtt{T}_2\mathtt{T}_1\mathtt{T}_\mathrm{E}$ of rigid body motions, where $\mathtt{T}_\mathrm{E}$ denotes the matrix form of g in (2.10), the matrix $\mathtt{T}_1$ serves to translate from $\mathtt{R}\mathbf{X} + \mathbf{t}$ to the origin $\mathbf{0}$ of the coordinate frame,

$$\mathtt{T}_1 = \begin{bmatrix} \mathtt{I} & -\mathtt{R}\mathbf{X} - \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{2.13}$$

the matrix $\mathtt{T}_2$ applies the desired rotational perturbation $\mathtt{R}'$,

$$\mathtt{T}_2 = \begin{bmatrix} \mathtt{R}' & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{2.14}$$

and the matrix $\mathtt{T}_3$ serves finally to translate from the origin $\mathbf{0}$ of the coordinate frame back to $\mathtt{R}\mathbf{X} + \mathbf{t}$,

$$\mathtt{T}_3 = \begin{bmatrix} \mathtt{I} & \mathtt{R}\mathbf{X} + \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{2.15}$$

2.4 Finite Projective Camera

Given a scene, let us understand a *view* to encompass an image of the scene and a *camera* that produced the image, which is to say a mathematical description of the mapping from points in the scene to pixels in the image plane. Let us first consider a classical *camera obscura*, consisting of only a dark chamber with a small aperture at one end and an image plane at the other. As we shrink the aperture, in the limit only those rays of light reflecting from a points outside the chamber that pass directly through the aperture are allowed to reach the image plane. The resulting idealized construction is called a *pinhole camera*. Let the *camera center* $\mathbf{C} \in \mathbb{R}^3$ denote the position of the infinitesimal aperture, placed at the origin $\mathbf{0}$ of the coordinate frame, and let $Z = -f$ denote the image plane. The Z-axis is termed the *optical axis*, and its intersection with the image plane is called the *principal point*. Given a point $\mathbf{X} = (X, Y, Z)^\top \in \mathbb{R}^3$ expressed in this coordinate frame, the projection of $\mathbf{X}$ is obtained by intersecting the image

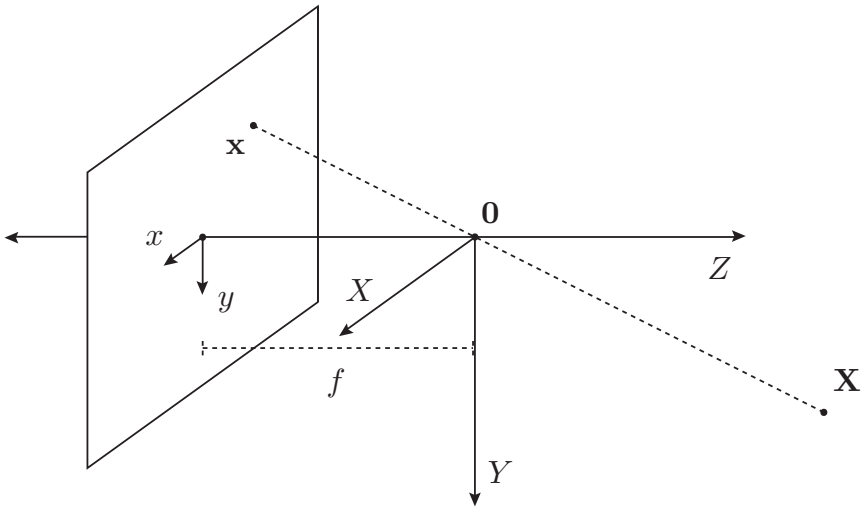


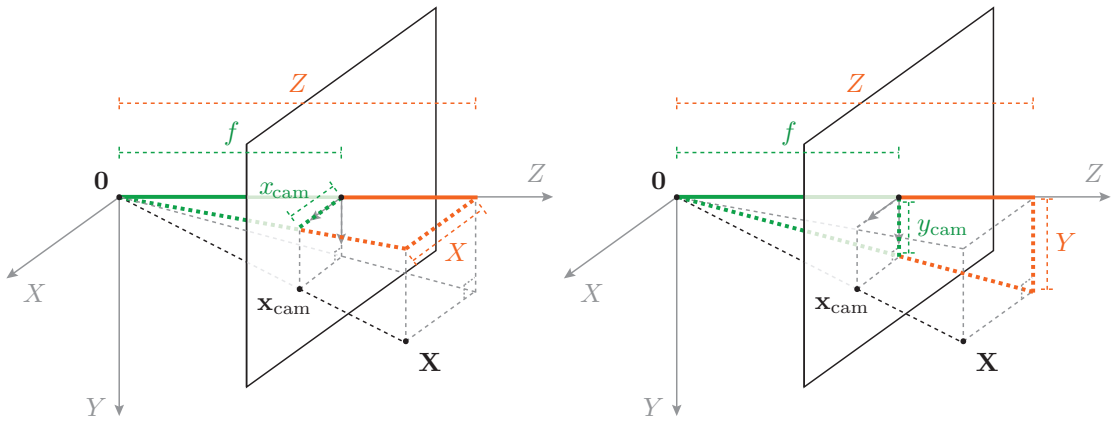
Figure 2.4: The pinhole camera, modeling an idealized *camera obscura* with infinitesimal aperture. The point $\mathbf{X} \in \mathbb{R}^3$ projects to $\mathbf{x} \in \mathbb{R}^2$, obtained by intersecting the image plane at $Z = -f$ with the line through $\mathbf{X}$ and the camera center $\mathbf{C} = \mathbf{0} \in \mathbb{R}^3$, which models the location of the aperture. Note that with the image plane at $Z = -f$, images appear flipped about the image plane's x- and y-axes.

plane with the line joining $\mathbf{C}$ and $\mathbf{X}$, called *central projection*. Since we desire that image coordinates be indexed such that x grow from left to right and y grow from top to bottom, the coordinate frame is placed such that the Y-axis point downwards (cf. Figure 2.4). Note, however, that like a genuine *camera obscura*, the image of a scene appears to be flipped along the x- and y-axes of the image plane. The effect of undoing this image flip is achieved by placing the image plane instead at $Z = f$, giving rise to the *frontal pinhole camera*. The projection $\mathbf{x} = (x, y)^\top \in \mathbb{R}^2$ of a point $\mathbf{X} = (X, Y, Z)^\top \in \mathbb{R}^3$ can be computed by similar triangles (cf. Figure 2.5), giving the mapping

$$(X, Y, Z)^\top \mapsto (fX/Z, fY/Z)^\top, \tag{2.16}$$

expressed in the same world units as those of the projecting 3-dimensional points (e.g., mm). Note that the division by Z in (2.16)—called *perspective division*—renders the mapping non-linear in X, Y, Z; for convenience, this non-linear mapping can be expressed in terms of a linear transformation from $\mathbb{P}^3$ to $\mathbb{P}^2$, since

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX/Z \\ fY/Z \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.17}$$



(a) $x_{\text{cam}} = fX/Z$. (b) $y_{\text{cam}} = fY/Z$.

Figure 2.5: The frontal pinhole camera. Placing the image plane at $Z = f$ has the effect of undoing the flipping of the image characteristic of the pinhole camera. Note that the projection $\mathbf{x}_{\text{cam}} = (x_{\text{cam}}, y_{\text{cam}})^\top = (fX/Z, fY/Z)^\top \in \mathbb{R}^2$ in camera coordinates of a point $\mathbf{X} = (X, Y, Z)^\top \in \mathbb{R}^3$ to the image plane can be obtained by similar triangles.

The 3×4 matrix in (2.17), denoted by $\mathtt{P}$, is called a *camera projection matrix*. This matrix can be further decomposed as

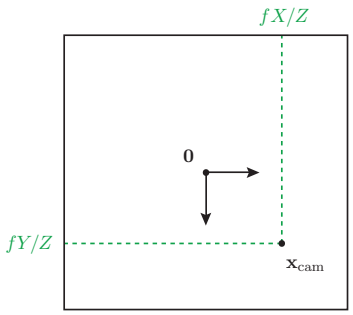
$$\mathtt{P} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}], \tag{2.18}$$

where the right 3×4 matrix $[\mathtt{I} \mid \mathbf{0}]$ in (2.18) is called the *canonical projection matrix*, and the left 3×3 matrix $\mathtt{K}$ is called a *camera calibration matrix*,

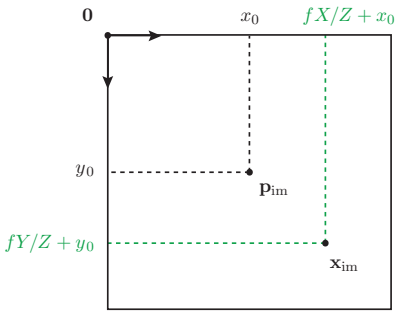
$$\mathtt{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.19}$$

World-to-Camera Coordinates. Projection by (2.18) assumes points $\mathbf{X} \in \mathbb{R}^3$ are expressed relative to the coordinate frame of the camera at hand, whereby the camera center $\mathbf{C}$ is situated at the origin $\mathbf{0}$ of the coordinate frame, the image plane is situated at $Z = f$, and the x- and y-axes of the image plane are aligned with the X- and Y-axes of the coordinate frame. Let this standard camera pose of the camera be called the *canonical pose*. More generally, projecting points expressed in the world coordinate frame to a camera at arbitrary pose can be achieved by transforming the points by the very rigid body motion $(\mathtt{R}, \mathbf{t})$ that transforms the camera in non-canonical pose to canonical pose (cf. Figure 2.7). Given a point $\tilde{\mathbf{X}} = (\mathbf{X}^\top, 1)^\top \in \mathbb{P}^3$ expressed in the world coordinate frame, the same point $\tilde{\mathbf{X}}_{\text{cam}} \in \mathbb{P}^3$ expressed in that camera's coordinate frame is given by

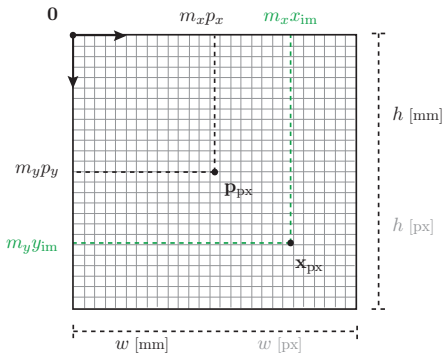
$$\tilde{\mathbf{X}}_{\text{cam}} = \begin{pmatrix} \mathtt{R}\mathbf{X} + \mathbf{t} \\ 1 \end{pmatrix} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \tilde{\mathbf{X}}. \tag{2.20}$$



(a) Camera coordinates $\mathbf{x}_{\text{cam}} = (x_{\text{cam}}, y_{\text{cam}})^\top$.



(b) Image coordinates $\mathbf{x}_{\text{im}} = (x_{\text{im}}, y_{\text{im}})^\top$.



(c) Pixel coordinates $\mathbf{x}_{\text{px}} = (x_{\text{px}}, y_{\text{px}})^\top$.

Figure 2.6: An image expressed in camera, image, and pixel coordinates. (a) In camera coordinates, the origin of the coordinate system of the image plane is situated at the principal point, obtained by the intersection of the image plane with the optical axis. Note that the projection $\mathbf{x}_{\text{cam}} = (fX/Z, fY/Z)^\top \in \mathbb{R}^2$ is expressed in the same world units (e.g., millimeters) as the projecting point $\mathbf{X} = (X, Y, Z)^\top \in \mathbb{R}^3$. (b) The projection $\mathbf{x}_{\text{im}} = (x_{\text{cam}} + x_0, y_{\text{cam}} + y_0)^\top \in \mathbb{R}^2$ in image coordinates places this origin of the image plane's coordinate system at the top left of the image plane, where $(x_0, y_0)^\top$ gives the location of the principal point in image coordinates. (c) Let world units be millimeters. Image coordinates in terms of pixel units are obtained by expressing the projection $\mathbf{x}_{\text{px}} = (m_x x_{\text{im}}, m_y y_{\text{im}})^\top \in \mathbb{R}^2$ in terms of pixel coordinates, where $m_x = w$ [px] $/w$ [mm] and $m_y = h$ [px] $/h$ [mm], where in turn h [mm] and w [mm] are the CCD height and width in world units and h [px] and w [px] the height and width of the image plane in pixels, respectively.

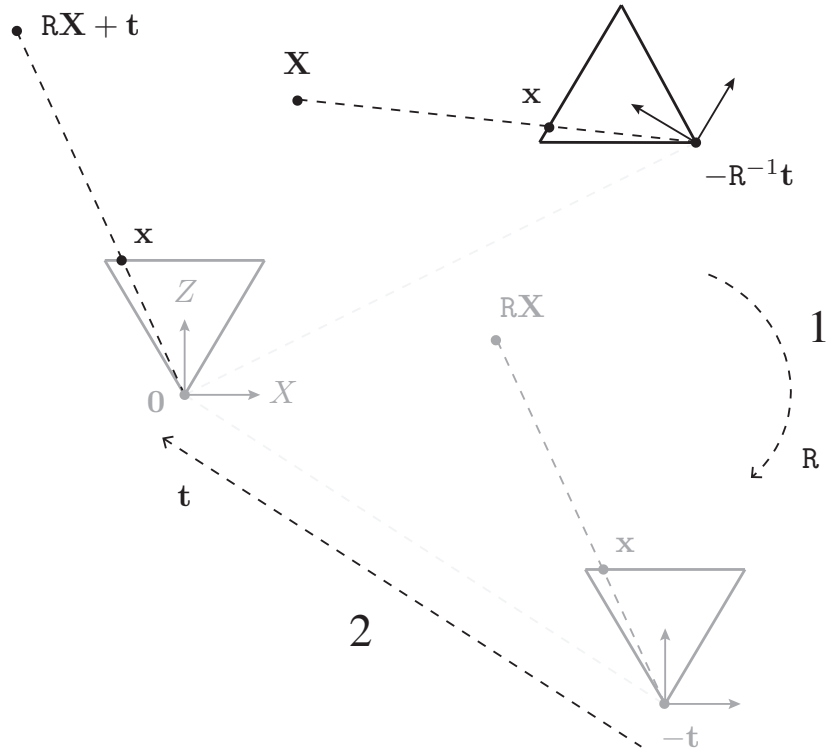


Figure 2.7: Projection $(\mathbf{x}^\top, 1)^\top \sim \mathtt{K}(\mathtt{R}\mathbf{X}+\mathbf{t}) = \mathtt{K}[\mathtt{R} \mid \mathbf{t}](\mathbf{X}^\top, 1)^\top$ of a point $\mathbf{X} \in \mathbb{R}^3$ to a camera in non-canonical pose expressed by $\mathtt{P} = \mathtt{K}[\mathtt{R} \mid \mathbf{t}]$. Projecting $\mathbf{X}$ by $\mathtt{P}$ calls for first expressing $\mathbf{X}$ in the camera coordinate frame of $\mathtt{P}$, which is achieved by transforming $\mathbf{X}$ by the rigid body motion $(\mathtt{R}, \mathbf{t})$. Note that the pose of the camera itself relative to canonical pose is given by the inverse motion $(\mathtt{R}^{-1}, -\mathtt{R}^{-1}\mathbf{t})$, giving camera center $\mathbf{C} = \mathtt{R}^{-1}\mathbf{0} - \mathtt{R}^{-1}\mathbf{t} = -\mathtt{R}^{-1}\mathbf{t}$.

Combining (2.20) and (2.18) gives the projection $\tilde{\mathbf{x}} \sim (\mathbf{x}^\top, 1)^\top \in \mathbb{P}^2$ of the point $\tilde{\mathbf{X}}$ expressed in world coordinates as

$$\tilde{\mathbf{x}} \sim \mathtt{K}[\mathtt{R} \mid \mathbf{t}]\tilde{\mathbf{X}} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}] \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \tilde{\mathbf{X}} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}]\tilde{\mathbf{X}}_{\text{cam}}. \tag{2.21}$$

The corresponding camera matrix $\mathtt{P}$ for a camera in non-canonical pose then takes the form

$$\mathtt{P} = \mathtt{K}[\mathtt{R} \mid \mathbf{t}] = \mathtt{K}[\mathtt{I} \mid \mathbf{0}] \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}. \tag{2.22}$$

The rigid body motion $(\mathtt{R}, \mathbf{t})$ constitutes the *extrinsic parameters* (or *extrinsics*) of the camera. Note that the pose of the camera itself relative to canonical pose is given by the inverse motion $(\mathtt{R}^{-1}, -\mathtt{R}^{-1}\mathbf{t})$, with camera center $\mathbf{C} = \mathtt{R}^{-1}\mathbf{0} - \mathtt{R}^{-1}\mathbf{t} = -\mathtt{R}^{-1}\mathbf{t}$. If the camera is in canonical pose, $(\mathtt{R}, \mathbf{t})$ is simply the identity motion $(\mathtt{I}, \mathbf{0})$, giving exactly (2.18) for (2.22).

Camera-to-Image Coordinates. The projection by (2.16) of a point $\mathbf{X} = (X, Y, Z)^\top \in \mathbb{R}^3$ expressed in the camera coordinate frame is expressed in terms of the the principal point as the origin of the coordinate frame of the image plane (cf. Figure 2.6a). In order to instead place this origin at the top-left corner of the image—as is the convention in image indexing—the camera calibration matrix $\mathtt{K}$ in (2.19) is generalized to

$$\mathtt{K} = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.23}$$

where $(-x_0, -y_0)^\top$ are the coordinates of the upper-left corner of the image plane when the origin is at the principal point, and hence $(x_0, y_0)^\top$ are the coordinates of the principal point when the origin is at the upper-left corner. The focal length f and principal point $(x_0, y_0)^\top$ constitute the *intrinsic parameters* (or *intrinsics*) of the camera. The mapping in (2.16) accordingly becomes

$$(X, Y, Z)^\top \mapsto (fX/Z + x_0, fY/Z + y_0)^\top, \tag{2.24}$$

as illustrated in Figure 2.6b.

Image-to-Pixel Coordinates. The units of a projection $\mathbf{x} = (x, y)^\top \in \mathbb{R}^2$ obtained by (2.24) of a point $\mathbf{X} = (X, Y, Z)^\top \in \mathbb{R}^3$ expressed in the camera coordinate frame remain the same as those of the projecting point. Suppose, without loss of generality, that these world units are millimeters. Let $m_x = w$ [px] $/w$ [mm] and $m_y = h$ [px] $/h$ [mm], where in turn h [mm] and w [mm] are the CCD height and width in world units and h [px] and w [px] the height and width of the image plane in pixels, respectively. In order to obtain the projection in units of pixels, the mapping in (2.24) becomes

$$(X, Y, Z)^\top \mapsto \big(m_x(fX/Z + x_0), m_y(fY/Z + y_0)\big)^\top, \tag{2.25}$$

as illustrated in Figure 2.6c. This effect is achieved by adjusting the camera calibration matrix $\mathtt{K}$ in (2.23) to

$$\mathtt{K} = \begin{bmatrix} m_x f & 0 & m_x x_0 \\ 0 & m_y f & m_y y_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.26}$$

World-to-Pixel Coordinates. By (2.22) and (2.26), the final world-to-pixel 3×4 camera projection matrix $\mathtt{P}$ is given by

$$\mathtt{P} = \begin{bmatrix} m_x f & 0 & m_x x_0 \\ 0 & m_y f & m_y y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} = \mathtt{K}[\mathtt{R} \mid \mathbf{t}]. \tag{2.27}$$

Note that in practice it is typically taken for granted that $m_x = m_y$, which holds when pixels are square. The quantity $m_x f = m_y f$ then gives the focal length in units of pixels, and $(m_x x_0, m_y y_0)^\top$ gives the principal point, likewise in pixels. Note that as of this point for the remainder of this thesis, when we refer to the focal length f or principal point $(x_0, y_0)^\top$ we shall understand them to be expressed in units of pixels.

Back-projection. The vector in $\mathbb{R}^3$ (such that $Z = 1$) from the origin $\mathbf{0}$ in the direction of the location of the pixel $\mathbf{x} = (x, y)^\top \in \mathbb{R}^2$ in the image plane expressed in the camera coordinate frame is called the *back-projection* of $\mathbf{x}$, and is determined in closed form by

$$\mathtt{K}^{-1}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} 1/f & 0 & -x_0/f \\ 0 & 1/f & -y_0/f \\ 0 & 0 & 1 \end{bmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{x-x_0}{f} \\ \frac{y-y_0}{f} \\ 1 \end{pmatrix}, \tag{2.28}$$

where, as indicated above, f is the focal length and $(x_0, y_0)^\top$ the principal point, each expressed in units of pixels.

Pre-image. The back-projection determines the orientation of the ray in the camera coordinate frame through $\mathbf{0}$ along which any point projecting to the pixel $\mathbf{x} \in \mathbb{R}^2$ is required by central projection to lie. Since the Z-coordinate of (2.28) is 1, it follows that the point $\mathbf{X} \in \mathbb{R}^3$ situated at depth Z projecting to $\mathbf{x}$ is given by

$$Z \cdot \mathtt{K}^{-1}(\mathbf{x}^\top, 1)^\top. \tag{2.29}$$

This point $\mathbf{X}$ can be considered the *pre-image* of $\mathbf{x}$ given Z.

2.5 Epipolar Geometry

The *epipolar geometry* between two views determines the mapping between a pixel in one view and the space of the pixel's possible correspondences in the other. That space takes the form of a line, and is used especially in stereo matching to restrict the search for correspondences to only those pixels that lie along that line. Let $\mathtt{P} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}]$ and $\mathtt{P}' = \mathtt{K}'[\mathtt{R} \mid \mathbf{t}]$ be two camera projection matrices encoding the cameras of two such views, which for convenience of language we shall call the left and right views, respectively, and where the right camera is expressed relative to the camera coordinate frame of the left. Let $\mathbf{X} \in \mathbb{R}^3$ denote a point about which we know only that it projects to the pixel $\mathbf{x}$ in the left view. The *epipolar constraint* expresses the requirement that the projection $\mathbf{x}'$ of $\mathbf{X}$ to the right view lie on the *epipolar line* $\mathbf{l}' \in \mathbb{P}^2$ corresponding to $\mathbf{x}$, determined by the intersection of the right view's image plane with the plane $\boldsymbol{\pi}$ through the ray along the back-projection $\mathtt{K}^{-1}(\mathbf{x}^\top, 1)^\top$ of $\mathbf{x}$ and the two camera centers $\mathbf{C} = \mathbf{0}, \mathbf{C}' = -\mathtt{R}^{-1}\mathbf{t}$ (cf. Figure 2.8).

Relative Pose. Let $\mathtt{P} = \mathtt{K}[\mathtt{R} \mid \mathbf{t}]$ and $\mathtt{P}' = \mathtt{K}'[\mathtt{R}' \mid \mathbf{t}']$ be two camera projection matrices encoding the cameras of two views, such that neither camera is in canonical pose. Expressing $\mathtt{P}'$ relative to $\mathtt{P}$ is achieved by right multiplying $\mathtt{P}'$ with the matrix form of the inverse of $(\mathtt{R}, \mathbf{t})$, since by (2.22)

$$\begin{aligned} \mathtt{K}[\mathtt{R} \mid \mathbf{t}]\begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}^{-1} &= \mathtt{K}[\mathtt{I} \mid \mathbf{0}]\begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}\begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}^{-1} \\ &= \mathtt{K}[\mathtt{I} \mid \mathbf{0}]. \end{aligned} \tag{2.30}$$

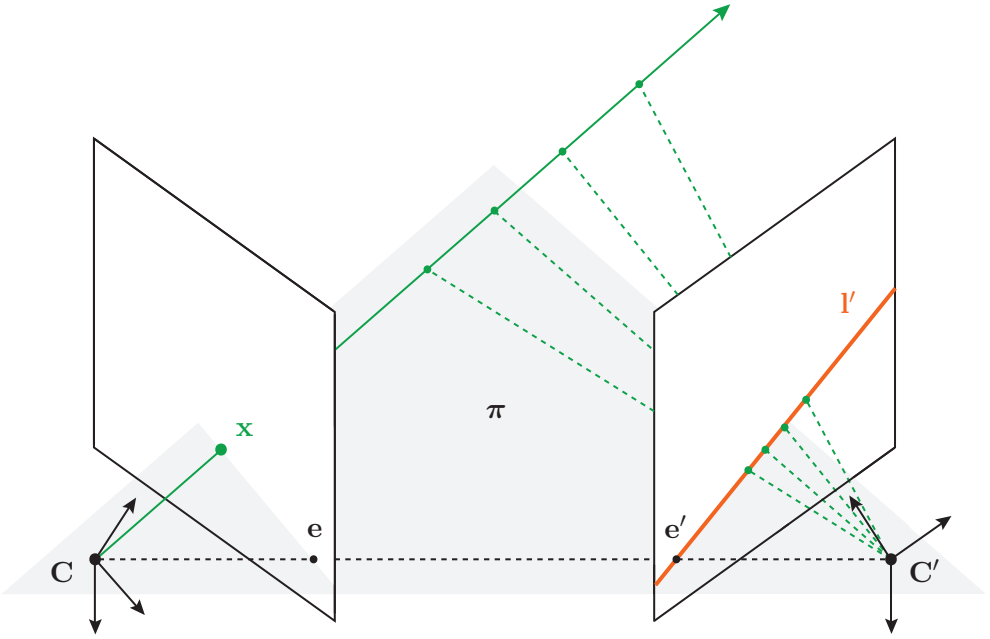


Figure 2.8: Epipolar geometry. The 3D point projecting to the pixel $\mathbf{x}$ in the left view is required to lie along the ray from the camera center $\mathbf{C}$ of the left camera in the direction of the back-projection of $\mathbf{x}$. Accordingly, the epipolar constraint states that the projection $\mathbf{x}'$ of that point to the right view must lie along the epipolar line $\mathbf{l}'$ corresponding to $\mathbf{x}$, obtained by intersecting the image plane of the right view with the plane $\boldsymbol{\pi}$ through that ray and the camera centers $\mathbf{C}, \mathbf{C}'$.

The camera projection matrix of the right view expressed relative to the camera coordinate frame of the left is, additionally by (2.12), accordingly

$$\begin{aligned}
\mathtt{K}'[\mathtt{R}' \mid \mathbf{t}'] \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}^{-1} &= \mathtt{K}'[\mathtt{I} \mid \mathbf{0}] \begin{bmatrix} \mathtt{R}' & \mathbf{t}' \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}^{-1} \\
&= \mathtt{K}'[\mathtt{I} \mid \mathbf{0}] \begin{bmatrix} \mathtt{R}' & \mathbf{t}' \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathtt{R}^{-1} & -\mathtt{R}^{-1}\mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \\
&= \mathtt{K}'[\mathtt{I} \mid \mathbf{0}] \begin{bmatrix} \mathtt{R}'\mathtt{R}^{-1} & -\mathtt{R}'\mathtt{R}^{-1}\mathbf{t} + \mathbf{t}' \\ \mathbf{0}^\top & 1 \end{bmatrix} \\
&= \mathtt{K}'[\mathtt{R}'\mathtt{R}^{-1} \mid -\mathtt{R}'\mathtt{R}^{-1}\mathbf{t} + \mathbf{t}'].
\end{aligned} \tag{2.31}$$

The ability to express any pair of cameras in terms of the camera coordinate frame of one of them allows us to reason in terms of the relative pose of the two cameras, of which we make use in our discussion of the essential matrix below.

The Essential Matrix. Let $\tilde{\mathbf{m}} = \mathtt{K}^{-1}(\mathbf{x}^\top, 1)^\top \in \mathbb{P}^2$ and $\tilde{\mathbf{m}}' = \mathtt{K}'^{-1}(\mathbf{x}'^\top, 1)^\top \in \mathbb{P}^2$, called the *normalized coordinates* of the pixels $\mathbf{x} \in \mathbb{R}^2$ and $\mathbf{x}' \in \mathbb{R}^2$ in the left and right view, respectively.

Given a pair of cameras $\mathtt{P} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}], \mathtt{P}' = \mathtt{K}'[\mathtt{R} \mid \mathbf{t}]$, the 3×3 *essential matrix* $\mathtt{E} = [\mathbf{t}]_\times \mathtt{R}$[4] encodes the epipolar geometry between $\mathtt{P}$ and $\mathtt{P}'$ such that the epipolar line $\mathbf{l}' \in \mathbb{P}^2$ in the right view corresponding to the pixel $\mathbf{x}$ in the left is given by

$$\mathbf{l}' = \mathtt{E}\tilde{\mathbf{m}}, \tag{2.32}$$

and, similarly, the epipolar line $\mathbf{l} \in \mathbb{P}^2$ in the left view corresponding to the pixel $\mathbf{x}'$ in the right is given by

$$\mathbf{l} = \mathtt{E}^\top \tilde{\mathbf{m}}'. \tag{2.33}$$

Note that if $\mathbf{x}, \mathbf{x}'$ satisfy the epipolar constraint, it follows by point-line incidence that

$$\tilde{\mathbf{m}}'^\top \mathtt{E} \tilde{\mathbf{m}} = 0. \tag{2.34}$$

We refer the reader to other texts for a derivation of $\mathtt{E} = [\mathbf{t}]_\times \mathtt{R}$, as well as for a discussion and derivation of the closely related *fundamental matrix* $\mathtt{F} = \mathtt{K}'^{-\top}[\mathbf{t}]_\times \mathtt{R}\mathtt{K}^{-1}$ (e.g., Hartley and Zisserman [HZ06] or Birchfield [Bir98]). The key point we wish to make here is that an essential matrix $\mathtt{E}$ can be computed from sparse image correspondences in normalized coordinates using any of a variety of so-called minimal algorithms. One way to proceed is to use the 5 point algorithm [Nis04], which estimates a matrix $\mathtt{E}$ from five such correspondence pairs, in combination with RANSAC [FB81], whereby candidates for $\mathtt{E}$ are computed from five correspondence pairs chosen at random, returning finally the candidate for which the subset of total correspondence pairs deemed inliers of $\mathtt{E}$ is maximized. Assuming the left camera is expressed in canonical pose, such a matrix $\mathtt{E}$ can be decomposed using SVD into four possible choices of rigid body motion $(\mathtt{R}, \mathbf{t})$ for the right camera, based on two possible choices of rotation matrix $\mathtt{R}$ and two possible signs for the translation vector $\mathbf{t}$ (cf. Hartley and Zisserman [HZ06]), among which one is chosen such that both recovered cameras point in the same direction. It is in this manner that we proceed in Chapter 7 to recover a dominant rigid body motion to explain motion in the scene from sparse correspondences between a pair of RGB frames.

Rectification. Given a view, the effect of rotating the corresponding camera—expressed in canonical pose and with camera calibration matrix $\mathtt{K}$—about its camera center according to a 3D rotation matrix $\mathtt{R}$ can be achieved by means of a homography encoded in the form of a 3×3 matrix by

$$\mathtt{K}\mathtt{R}\mathtt{K}^{-1}, \tag{2.35}$$

which serves to map a pixel in the rotated view to its correspondence in the unrotated original view. Warping the images of a two-view setup such that pixels in one view and their corresponding epipolar lines in the other lie on the same horizontal scanline (cf. Figure 2.9) is termed *rectification*, and rectifying homographies can be computed using any of a number of techniques

[4] If $\mathbf{a} = (a_1, a_2, a_3)^\top$ and $\mathbf{b} = (b_1, b_2, b_3)^\top$ then $\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b}$, where

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}.$$

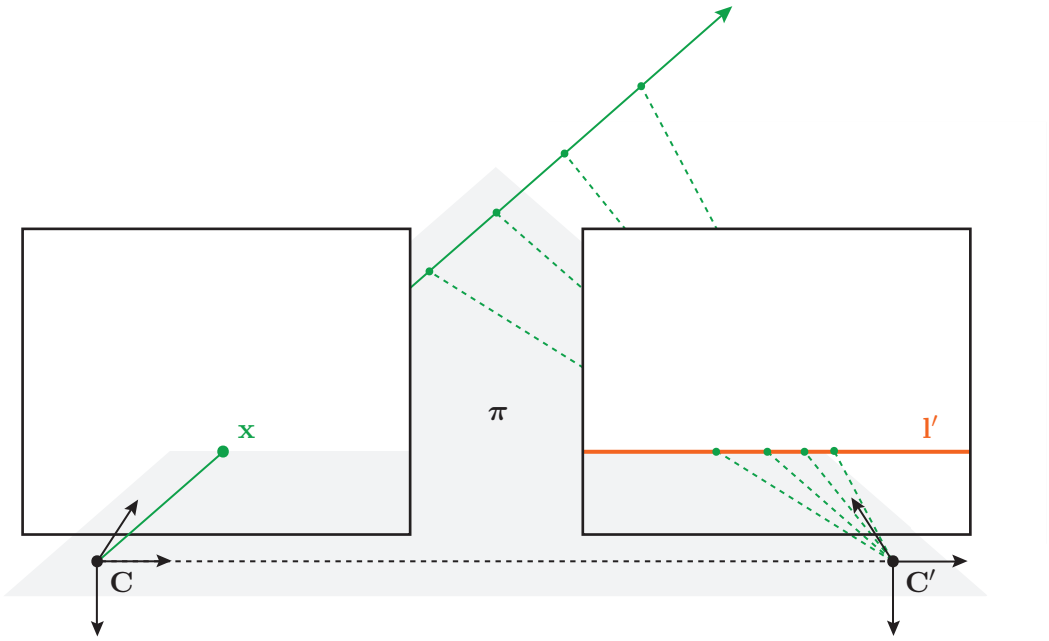


Figure 2.9: Rectified epipolar geometry. The images are projectively warped by applying appropriate rectifying homographies with the effect of having rotated the corresponding cameras about their respective camera centers, such that the epipolar line $\mathbf{l}'$ corresponding to any pixel $\mathbf{x}$ lie on the same scanline as $\mathbf{x}$. Note that the majority of stereo algorithms assume assume the input image pair has been rectified.

(e.g., Bouguet's algorithm [Bou]). Let B denote the Euclidean distance between the two camera centers, termed the *baseline*, and let the rotated right camera be expressed relative to the rotated left camera in canonical pose. Given a point $\mathbf{X} \in \mathbb{R}^3$, its projections $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^2$ to the rectified left and right views, respectively, are given according to

$$\mathbf{x} = (x, y)^\top = \big(fX/Z + x_0, fY/Z + y_0\big)^\top \tag{2.36}$$

and

$$\mathbf{x}' = (x', y)^\top = \big(f(X - B)/Z + x_0, fY/Z + y_0\big)^\top. \tag{2.37}$$

The horizontal displacement $x - x' \geq 0$ is termed the *disparity* between $\mathbf{x}$ and its correspondence $\mathbf{x}'$. Observe that

$$\frac{Bf}{x - x'} = \frac{Bf}{f\big(X - (X - B)\big)/Z + x_0 - x_0} = Z, \tag{2.38}$$

which gives a convenient expression for the relationship between a pixel correspondence in a rectified setting and the depth of the corresponding projecting point. For the majority of stereo algorithms (cf. Section 4.3), the input is assumed to be a rectified image pair, and the output takes the form of a *disparity map* encoding the disparities per pixel of one of the two rectified

images with respect to recovered correspondences in the other. The corresponding *depth map* can easily be computed by applying (2.38), and a corresponding point cloud by subsequently applying (2.29).

2.6 Homography Induced by the Plane

Let $\boldsymbol{\pi} = (\mathbf{n}^\top, -d)^\top \in \mathbb{P}^3$ denote a plane in 3D. Let $\mathbf{X} \in \mathbb{R}^3$ denote a point about which we know only that it lie in that plane and that it project to the pixel $\mathbf{x} \in \mathbb{R}^2$. A plane-induced homography gives a mapping $\mathbf{x} \leftrightarrow \mathbf{x}'$ with the effect of recovering the point $\mathbf{X}$ by intersecting the plane $\boldsymbol{\pi}$ with the ray along the back-projection $\mathbf{p} = \mathtt{K}^{-1}(\mathbf{x}^\top, 1)^\top$ of $\mathbf{x}$, then transforming $\mathbf{X}$ according to a 3D rigid body motion $(\mathtt{R}, \mathbf{t})$, and finally projecting the transformed point $\mathbf{X}' = \mathtt{R}\mathbf{X} + \mathbf{t}$ back to image space. The homogeneous counterpart $\tilde{\mathbf{X}} = (\mathbf{p}^\top, \lambda)^\top \in \mathbb{P}^3$ of $\mathbf{X}$ encodes precisely this ray along the back-projection $\mathbf{p}$ of $\mathbf{x}$, parameterized by λ. Since the point $\mathbf{X}$ lies in the plane $\boldsymbol{\pi}$, it follows that $\boldsymbol{\pi}^\top(\mathbf{p}^\top, \lambda)^\top = \mathbf{n}^\top\mathbf{p} - d\lambda = 0$, which gives $\lambda = \mathbf{n}^\top\mathbf{p}/d$. Noting that by (2.28) the Z-coordinate of $\mathbf{p}$ is 1, the depth Z of the inhomogeneous point $\mathbf{X}$ then happens to be $1/\lambda = d/\mathbf{n}^\top\mathbf{p}$, giving $\mathbf{X}$ by (2.29) as

$$\mathbf{X} = \mathbf{p}/\lambda = \frac{d}{\mathbf{n}^\top\mathbf{p}}\mathtt{K}^{-1}(\mathbf{x}^\top, 1)^\top, \tag{2.39}$$

a fact used to generate Figure 1.4a. The desired mapping $\mathbf{x} \leftrightarrow \mathbf{x}'$, however, can be expressed without explicitly recovering the three coordinates of the point $\mathbf{X}$, since we can write

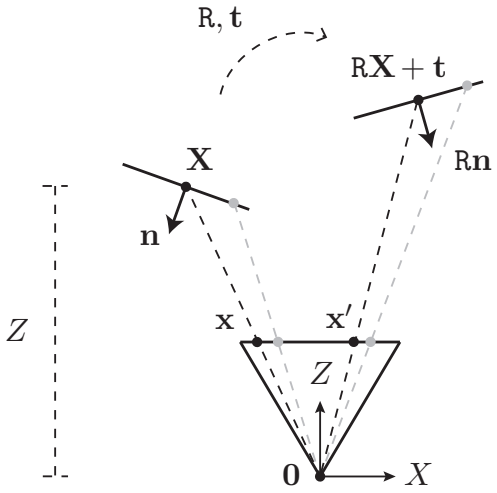
$$\begin{aligned}
(\mathbf{x}'^\top, 1)^\top &\sim \mathtt{K}[\mathtt{R} \mid \mathbf{t}] \begin{pmatrix} \mathbf{p}^\top, \lambda \end{pmatrix}^\top \\
&= \mathtt{K}[\mathtt{R} \mid \mathbf{t}] \begin{pmatrix} \mathbf{p}^\top, \mathbf{n}^\top\mathbf{p}/d \end{pmatrix}^\top \\
&= \mathtt{K} \begin{pmatrix} \mathtt{R}\mathbf{p} + \mathbf{t}\mathbf{n}^\top\mathbf{p}/d \end{pmatrix} \\
&= \mathtt{K} \begin{pmatrix} \mathtt{R} + \mathbf{t}\mathbf{n}^\top/d \end{pmatrix} \mathbf{p} \\
&= \mathtt{K} \begin{pmatrix} \mathtt{R} + \mathbf{t}\mathbf{n}^\top/d \end{pmatrix} \mathtt{K}^{-1}(\mathbf{x}^\top, 1)^\top.
\end{aligned}$$

Accordingly, the final homography over $\mathbb{P}^2$ induced by the plane $\boldsymbol{\pi} = (\mathbf{n}^\top, -d)^\top \in \mathbb{P}^3$ (3 DoF) and 3D rigid body motion $(\mathtt{R}, \mathbf{t})$ (6 DoF) can be expressed as the 3×3 matrix

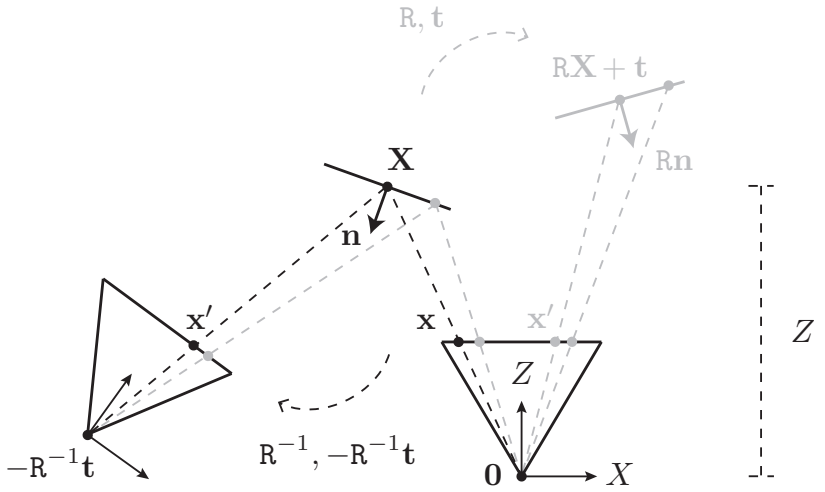
$$\mathtt{K} \begin{pmatrix} \mathtt{R} + \mathbf{t}\mathbf{n}^\top/d \end{pmatrix} \mathtt{K}^{-1}, \tag{2.40}$$

giving a total of 9 DoF for camera calibration matrix $\mathtt{K}$ fixed. Note that such a homography can be interpreted in terms of a plane undergoing rigid motion relative to a single fixed camera $\mathtt{P} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}]$, or in terms of a fixed plane viewed by two cameras $\mathtt{P} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}]$ and $\mathtt{P}' = \mathtt{K}[\mathtt{R} \mid \mathbf{t}]$ (cf. Figure 2.10). On the latter interpretation, note that for $\mathtt{P}' = \mathtt{K}'[\mathtt{R} \mid \mathbf{t}]$ (2.40) becomes

$$\mathtt{K}' \begin{pmatrix} \mathtt{R} + \mathbf{t}\mathbf{n}^\top/d \end{pmatrix} \mathtt{K}^{-1}. \tag{2.41}$$



(a) Interpretation as fixed camera, moving plane.



(b) Interpretation as fixed plane, moving camera.

Figure 2.10: Two interpretations of a plane-induced homography. (a) A point in the plane—obtained by intersecting the plane with a pixel's back-projection—undergoes a 3D rigid body motion $(\mathtt{R}, \mathbf{t})$ and is then projected back to image space. (b) The source camera is in the canonical pose, the pose of the destination camera relative to canonical pose is given by $(\mathtt{R}^{-1}, -\mathtt{R}^{-1}\mathbf{t})$; projection of the same point in the plane to the destination view calls for expressing the point in the camera coordinate frame of the destination camera, which is achieved by transforming the point likewise by the motion $(\mathtt{R}, \mathbf{t})$.

CHAPTER 3

MAP Inference by PatchMatch and PMBP

A fruitful way to approach the problem of dense correspondence search is to cast it as a labeling problem solved in terms of maximizing a joint probability distribution, which is to say by *maximum a posteriori* (MAP) inference. A *label* is understood here to serve as an *index* to one of a subset of conceivable motions defined, perhaps, in terms of 2 DoF translations or, as in the chapters that follow, in terms of 6 DoF 3D rigid body motions or 9 DoF plane-induced homographies. The aim is to assign such a label to each pixel of an image such that the resulting *labeling* be optimal with respect to a chosen *objective function* defined over the label space and the space of input images. More precisely, such an objective function might be defined, for example, in terms of only functions called *unary potentials* intended to penalize labelings that are inconsistent with the given image data, or perhaps additionally by means of *pairwise potentials* intended to promote smoothness of the recovered correspondence field. *Probabilistic graphical models* provide a framework for reasoning about such labeling problems in terms of maximizing a joint probability distribution such that conditional independence properties of the joint distribution are encoded in a graph. Exploiting these very conditional independence properties is a key ingredient in many inference algorithms. In general, however, the problem of computing such a labeling is NP-hard [BVZ01], with many popular approximative algorithms intended for tackling only discrete label spaces (e.g., graph cuts [BVZ01, KZ04], loopy belief propagation [YFW00], or tree-reweighted message passing [Kol06]), often having the consequence of requiring a coarse discretization of the underlying space of conceivable motions. The recently introduced PatchMatch Belief Propagation (PMBP) algorithm [BRFK12] is an approximative method that provides an avenue to recovering dense correspondences over *continuous* label spaces by leveraging PatchMatch [BSFG09, BSGF10] for exploiting spatial coherence characteristic of typical correspondence fields by sampling motions from pixel neighbors (termed *spatial propagation*), and a flavor of loopy belief propagation called particle belief propagation (PBP) for explicitly promoting smoothness.

The objective of this chapter is to give a self-contained derivation—drawing mainly on Bishop [Bis06] and Nowozin and Lampert [NL11]—of PMBP and PatchMatch[1] in terms of maximizing joint probability distributions by message passing. This is in contrast to [BRFK12], where PMBP is derived in a considerably more high level manner in terms of a formulation of PBP the authors themselves qualify as spartan. We begin this chapter in Section 3.1 with a brief review of some essential concepts of probability theory. We then introduce probabilistic graphical models in Section 3.2. Message passing for exact inference on graphs that do not contain loops is treated in Section 3.3, and extended to loopy graphs in Section 3.4. Finally, in Section 3.5 we detail PMBP and PatchMatch, in terms of which we carry out our correspondence search in the chapters to come.

3.1 Variables, Events, and Probabilities

Let X denote a *discrete random variable*, which is a *variable* in the sense that X can take on any of a set $\mathcal{L} = \{x^1, \ldots, x^K\}$ of possible *values* (or *labels*) $x \in \mathcal{L}$, *discrete* in the sense that the number $|\mathcal{L}| = K$ of possible values of X is finite, and *random* in the sense that rather than be known with certainty, the occurrence of an *event* $X = x$ has associated with it a probability $p(X = x)$. The probability $p(X = x)$ associated with each event $X = x$ is described by a function $p : \mathcal{L} \to \mathbb{R}^+ \cup \{0\}$, which is a *probability distribution* and as such is required to satisfy $\sum_{x \in \mathcal{L}} p(X = x) = 1$. Let us introduce the notation $x^i \in \mathcal{L}$, such that $i \in \{1, \ldots, K\}$, in order to be able to speak, when needed, in clearer terms about a particular member of the set $\mathcal{L}$. The probability $p(X = x^i)$ of an event $X = x^i$ is then defined by n_i/n, where n_i is the number of times the event $X = x^i$ occurs from a total of n trials, taken in the limit as $n \to \infty$. Let X_1, X_2 denote a pair of discrete random variables, and let X now denote a discrete random variable such that $X = x$ be understood to denote $X_1 = x_1, X_2 = x_2$, where $x = (x_1, x_2) \in \mathcal{L}_1 \times \mathcal{L}_2$. The *joint probability distribution* $p(X = x) = p(X_1 = x_1, X_2 = x_2)$ of $X_1 = x_1$ and $X_2 = x_2$ is defined by

$$p(X_1 = x_1^i, X_2 = x_2^j) = \frac{n_{ij}}{n}, \tag{3.1}$$

where $p(X_1 = x_1, X_2 = x_2)$ is understood to mean $p(X_1 = x_1 \text{ and } X_2 = x_2)$ and n_{ij} is the number of trials for which both $X_1 = x_1^i$ and $X_2 = x_2^j$ as $n \to \infty$. Suppose the variable X_1 represents a marble's color, and X_2 the box to which the marble belongs. The number n_{ij} is then the number of marbles of color x_1^i belonging to box x_2^j.

Conditional Probability. Suppose it is known that $X_1 = x_1^i$. We obtain the *conditional probability distribution* of $X_2 = x_2$ given $X_1 = x_1^i$ as

$$\begin{aligned} p(X_2 = x_2^j \mid X_1 = x_1^i) &= \frac{p(X_1 = x_1^i, X_2 = x_2^j)}{p(X_1 = x_1^i)} \\ &= \frac{n_{ij}/n}{\sum_j n_{ij}/n} = \frac{n_{ij}}{\sum_j n_{ij}}, \end{aligned} \tag{3.2}$$

[1] As in [BRFK12], we treat PatchMatch as a specialization of PMBP.

where $\sum_j$ is shorthand for $\sum_{j\in\{1,\dots,|\mathcal{L}_2|\}}$. In our example, (3.2) would give the number n_{ij} of marbles of color x_1^i belonging to the box x_2^j as a fraction of the number $\sum_j n_{ij}$ of marbles of color x_1^i contained across all possible boxes $x_2 \in \mathcal{L}_2$. Note that both joint probability distributions and conditional probability distributions generalize to arbitrary numbers of variables.

Sum Rule. The *marginal probability distribution* $p(X_s = x_s)$ of a joint probability distribution $p(X_1 = x_1, \dots, X_N = x_N)$ for a variable X_s of the joint distribution can be computed by applying the *sum rule* of probability:

$$p(X_s = x_s) = \sum_{x_1} \cdots \sum_{x_{s-1}} \sum_{x_{s+1}} \cdots \sum_{x_N} p(x_1, \dots, x_{s-1}, x_s, x_{s+1}, \dots, x_N), \tag{3.3}$$

where $p(x_1, \dots, x_N)$ is shorthand for $p(X_1 = x_1, \dots, X_N = x_N)$ and $\sum_{x_t}$ is shorthand for $\sum_{x_t\in\mathcal{L}_t}$. Accordingly, summation is carried out over the joint probabilities of all $|\mathcal{L}_1| \times \cdots \times |\mathcal{L}_{s-1}| \times |\mathcal{L}_{s+1}| \times \cdots \times |\mathcal{L}_N|$ possible events $X = x$ such that $x \in \mathcal{L}_1 \times \cdots \times \mathcal{L}_{s-1} \times \{x_s\} \times \mathcal{L}_{s+1} \times \cdots \times \mathcal{L}_N$ for $X_s = x_s$ fixed. This process is called *marginalization*. In our example of marbles and boxes, it follows that the marginal $p(X_1 = x_1)$ is obtained by

$$p(X_1 = x_1^i) = \sum_{x_2^j \in \mathcal{L}_2} p(X_1 = x_1^i, X_2 = x_2^j) = \sum_j \frac{n_{ij}}{n}, \tag{3.4}$$

giving the probability of choosing a marble of color x_1^i.

Product Rule. It can be shown by induction that any joint probability distribution $p(X_1 = x_1, \dots, X_N = x_N)$ can be *factorized* into a product of conditional probabilities:

$$p(X_1 = x_1, \dots, X_N = x_N) = p(x_N \mid x_1, \dots, x_{N-1}) \cdots p(x_3 \mid x_1, x_2) p(x_2 \mid x_1) p(x_1), \tag{3.5}$$

where $p(x_2 \mid x_1)$ is shorthand for $p(X_2 = x_2 \mid X_1 = x_1)$. This gives the *product rule* (or *chain rule*) of probability. By (3.2) and (3.4), the joint probability distribution $p(X_1 = x_1, X_2 = x_2)$ from our example factorizes accordingly as

$$\begin{aligned} p(X_1 = x_1^i, X_2 = x_2^j) &= \frac{n_{ij}}{n} \\ &= \frac{n_{ij}}{\sum_j n_{ij}} \cdot \frac{\sum_j n_{ij}}{n} = p(X_2 = x_2^j \mid X_1 = x_1^i) p(X_1 = x_1^i) \qquad (3.6) \\ &= \frac{n_{ij}}{\sum_i n_{ij}} \cdot \frac{\sum_i n_{ij}}{n} = p(X_1 = x_1^i \mid X_2 = x_2^j) p(X_2 = x_2^j), \qquad (3.7) \end{aligned}$$

where $\sum_i$ is shorthand for $\sum_{i\in\{1,\dots,|\mathcal{L}_1|\}}$.

Independence. Given a joint probability distribution $p(X_1 = x_1, X_2 = x_2)$, to write of a particular configuration $(x_1, x_2) \in \mathcal{L}_1 \times \mathcal{L}_2$ that

$$p(X_1 = x_1 \mid X_2 = x_2) = p(X_1 = x_1) \tag{3.8}$$

is to say that knowing $X_2 = x_2$ reveals nothing about the probability of $X_1 = x_1$. The two *events* $X_1 = x_1, X_2 = x_2$ are called *independent* if and only if, by 3.5 and (3.8), it holds that

$$\begin{aligned} p(X_1 = x_1, X_2 = x_1) &= p(X_1 = x_1 \mid X_2 = x_2)p(X_2 = x_2) \\ &= p(X_1 = x_1)p(X_2 = x_2). \end{aligned} \tag{3.9}$$

Conditional Independence. Two *random variables* X_1, X_2 of a joint probability distribution $p(X_1 = x_1, X_2 = x_2, X_3 = x_3)$ are said to be *conditionally independent* given a third variable X_3 if and only if for *all* configurations $(x_1, x_2, x_3) \in \mathcal{L}_1 \times \mathcal{L}_2 \times \mathcal{L}_3$, it holds that

$$p(X_1 = x_1, X_2 = x_2 \mid X_3 = x_3) = p(X_1 = x_1 \mid X_3 = x_3)p(X_2 = x_2 \mid X_3 = x_3). \tag{3.10}$$

In words, (3.10) states that for each such configuration (x_1, x_2, x_3) the events $X_1 = x_1, X_2 = x_2$ are independent in their joint distribution conditioned on the event $X_3 = x_3$. Note that, in conjunction with (3.8), it follows that

$$p(X_1 = x_1 \mid X_2 = x_2, X_3 = x_3) = p(X_1 = x_1 \mid X_3 = x_3), \tag{3.11}$$

which is to say that, given any $X_3 = x_3$, no $X_2 = x_2$ has any impact on the probability of the occurrence of any $X_1 = x_1$. Similarly,

$$p(X_2 = x_2 \mid X_1 = x_1, X_3 = x_3) = p(X_2 = x_2 \mid X_3 = x_3). \tag{3.12}$$

This conditional independence property over the variables X_1, X_2, X_3 for *all* configurations $(x_1, x_2, x_3) \in \mathcal{L}_1 \times \mathcal{L}_2 \times \mathcal{L}_3$ is expressed more compactly using the notation

$$X_1 \perp X_2 \mid X_3. \tag{3.13}$$

Bayes' Theorem. Observe that by equating (3.6) and (3.7), it follows for a joint probability distribution $p(X_1 = x_1, X_2 = x_2)$ that

$$p(X_2 = x_2 \mid X_1 = x_1) = \frac{p(X_1 = x_1 \mid X_2 = x_2)p(X_2 = x_2)}{p(X_1 = x_1)}, \tag{3.14}$$

where the denominator $p(X_1 = x_1)$ can be obtained from $p(X_1 = x_1, X_2 = x_2)$ by applying the sum rule in (3.3), which in conjunction with (3.7) gives

$$\begin{aligned} p(X_1 = x_1) &= \sum_{x_2 \in \mathcal{L}_2} p(X_1 = x_1, X_2 = x_2) \\ &= \sum_{x_2 \in \mathcal{L}_2} p(X_1 = x_1 \mid X_2 = x_2)p(X_2 = x_2). \end{aligned} \tag{3.15}$$

Note that each constituent probability distribution of (3.14) is given a name, such that

$$\text{posterior} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}. \tag{3.16}$$

3.2 Probabilistic Graphical Models

Computing the marginal distribution $p(X_s = x_s)$ of a joint probability distribution $p(X_1 = x_1, \ldots, X_N = x_N)$ for a variable X_s of the joint distribution by the sum rule in (3.3) is a process that calls for exhaustively enumerating all possible configurations $x = (x_1, \ldots, x_N) \in \mathcal{L} = \mathcal{L}_1 \times \cdots \times \mathcal{L}_N$, and hence takes $O(K^N)$ time if $|\mathcal{L}_1| = \cdots = |\mathcal{L}_N| = K$. Exploiting conditional independence properties between variables of the joint distribution offers a way around the exponential cost of exhaustively enumerating possible configurations. *Probabilistic graphical models* provide an avenue to identifying conditional independence properties of a joint probability distribution by reading those properties out from an appropriately structured graph. Let $G = (\mathcal{V}, \mathcal{E})$ denote a *graph*, defined in terms of a tuple comprising a set $\mathcal{V}$ of *vertices* (or *nodes*) $s \in \mathcal{V} \subset \mathbb{Z}^+$ and a set $\mathcal{E}$ of edges linking a subset of pairs of vertices $s, t \in \mathcal{V}$. Two common classes of probabilistic graphical models are *Bayesian networks* (or *directed models*) and *Markov random fields* (or *undirected models*). We shall—reflecting the aim of this chapter to derive PMBP and PatchMatch—concern ourselves here with only the latter, expressed using *directed* graphs whereby the edges $\{s, t\} \in \mathcal{E}$ do not encode order, contrary to the ordered edges $(s, t) \in \mathcal{E}$ of a Bayesian network.

Markov Random Fields. Let $G = (\mathcal{V}, \mathcal{E})$ denote an *undirected* graph, such that for each random variable $X_1, \ldots, X_N$ of a joint probability distribution $p(X_1 = x_1, \ldots, X_N = x_N)$ there be a corresponding vertex in $\mathcal{V} = \{1, \ldots, N\}$, with the edge set $\mathcal{E}$ serving to encode probabilistic relationships between the variables. A *clique* of the graph G is a subset $\mathcal{V}_C \subset \mathcal{V}$ of the vertices of G such that there exists an edge $\{s, t\} \in \mathcal{E}$ between every pair of vertices $s, t \in \mathcal{V}_C$, which is to say that the subgraph $G_C = (\mathcal{V}_C, \mathcal{E})$ spanned by the clique is *connected*. A clique is qualified as *maximal* if there exists no vertex $s \in \mathcal{V} \setminus \mathcal{V}_C$ such that there exists an edge $\{s, t\} \in \mathcal{E}$ for each vertex $t \in \mathcal{V}_C$. The joint distribution is a *Gibbs distribution* if it factorizes over maximal cliques of G as

$$p(X = x) = \frac{1}{Z} \prod_{C \in \mathcal{C}(G)} \psi_C(x_C), \tag{3.17}$$

where $p(X = x) = p(X_1 = x_1, \ldots, X_N = x_N)$, and $\mathcal{C}(G)$ serves as an index set of the set $\bigcup_{C \in \mathcal{C}(G)} \{\mathcal{V}_C\}$ of maximal cliques of G, and where functions $\psi_C : \mathcal{L}_C \to \mathbb{R}^+$—called *potential functions* (also known as *potentials* or *factors*)—are defined over the possible configurations $x_C \in \mathcal{L}_C = \bigtimes_{s \in \mathcal{V}_C} \mathcal{L}_s$ for $C \in \mathcal{C}(G)$. Note that the configuration $x_C \in \mathcal{L}_C$ in the right hand side of (3.17) is read out from the configuration $x = (x_1, \ldots, x_N) \in \mathcal{L} = \mathcal{L}_1 \times \cdots \times \mathcal{L}_N$ provided in the left hand side. The *partition function* Z in (3.17) is obtained by

$$Z = \sum_{x \in \mathcal{L}} \prod_{C \in \mathcal{C}(G)} \psi_C(x_C), \tag{3.18}$$

computed by summing over all possible configurations $x \in \mathcal{L}$, and serves to ensure that (3.17) satisfy $\sum_{x \in \mathcal{L}} p(X = x) = 1$. By slight abuse of notation, let us understand $p(\bigcup_{s \in \mathcal{V}} X_s = x_s)$ to express $p(X_1 = x_1, \ldots, X_N = x_N)$. By the Hammersley-Clifford theorem, it follows

that the graph G specifies a *Markov random field* (MRF), with the property—called the *local Markov property*—that for any variable X_s of $p(X_1 = x_1, \ldots, X_N = x_N)$ and any configuration $(x_1, \ldots, x_N) \in \mathcal{L}_1 \times \cdots \times \mathcal{L}_N$, it holds that $p(X_s = x_s \mid \bigcup_{t \in \mathcal{V} \setminus \{s\}} X_t = x_t) = p(X_s = x_s \mid \bigcup_{t \in \mathcal{N}_s} X_t = x_t)$, where $\mathcal{N}_s \subset \mathcal{V}$ denotes the set of vertices $t \in \mathcal{E}$ such that $\{s, t\} \in \mathcal{E}$. By similar slight abuse of the notation of conditional independence, this is to say that

$$X_s \perp \bigcup_{t \in \mathcal{V} \setminus \mathcal{N}_s^+} X_t \mid \bigcup_{t \in \mathcal{N}_s} X_t, \tag{3.19}$$

where $\mathcal{N}_s^+ = \mathcal{N}_s \cup \{s\}$. More generally, the Hammersley-Clifford theorem states that the set of distributions that are consistent with the conditional independence properties encoded in an MRF specified by an undirected graph G is identical to the set of distributions that can be factorized over the maximal cliques of G as in (3.17), provided that $\psi_C(x_C) > 0$ for each $x_C \in \mathcal{L}_C, C \in \mathcal{C}(G)$. Accordingly, it is convenient to define the potentials in terms of exponential functions, such that

$$\psi_C(x_C) = \exp\big(-E_C(x_C)\big), \tag{3.20}$$

where $E_C : \mathcal{L}_C \to \mathbb{R}$ is called an *energy function*.

Conditional Random Fields. The factorization of the joint distribution in (3.17) can be written more generally to reflect conditioning on an *observation* $Y = y$, giving

$$p(X = x \mid Y = y) = \frac{1}{Z(y)} \prod_{C \in \mathcal{C}(G)} \psi_C(x_C; y), \tag{3.21}$$

where the partition function $Z(y)$ now depends on y:

$$Z(y) = \sum_{x \in \mathcal{L}} \prod_{C \in \mathcal{C}(G)} \psi_C(x_C; y). \tag{3.22}$$

Such an observation $Y = y$ can be understood to encode, e.g., a depth map, a pair of RGB-D frames, or a pair of images, as in the chapters that follow. Note that we may write conditioning explicitly as in (3.21), or understand it from the context to be encoded implicitly in the potential functions of the factorization in (3.17). If the conditioning is written explicitly, the corresponding MRF is called a *conditional random field* (CRF).

Pairwise MRFs. Note that nothing speaks against a clique potential $\psi_C : \mathcal{L}_C \to \mathbb{R}^+$ in (3.17) being itself defined in terms of a product of potentials over the variables corresponding to $\mathcal{V}_C$. Suppose, for instance, that $\mathcal{L}_C = \mathcal{L}_1 \times \mathcal{L}_2$; a conceivable factorization of $\psi_C(x_1, x_2)$ might then take the form

$$\psi_C(x_1, x_2) = \psi_1(x_1)\psi_{1,2}(x_1, x_2), \tag{3.23}$$

where $\psi_1 : \mathcal{L}_1 \to \mathbb{R}^+$ is called a *unary potential* and $\psi_{1,2} : \mathcal{L}_1 \times \mathcal{L}_2 \to \mathbb{R}^+$ a *pairwise potential*. Of particular interest in computer vision and in this thesis are Gibbs distributions that

factorize—subject to an underlying factorization over maximal cliques—into a product of unary potentials $\psi_s(x_s; y)$ conditioned on an observation $Y = y$ and pairwise potentials $\psi_{s,t}(x_s, x_t)$,

$$p(X = x \mid Y = y) = \frac{1}{Z(y)} \prod_{s \in \mathcal{V}} \psi_s(x_s; y) \prod_{\{s,t\} \in \mathcal{E}} \psi_{s,t}(x_s, x_t), \tag{3.24}$$

giving rise to what are called *pairwise* MRFs. The topology of the undirected graph $G = (\mathcal{V}, \mathcal{E})$ that specifies the MRF corresponding to (3.24) is that of a lattice, such that vertices $s \in \mathcal{V}$ can be arranged in the form of a regular 2D grid and $\mathcal{E} = \bigcup_{s \in \mathcal{V}} \big\{\{s, t\} \mid t \in \mathcal{N}_s\big\}$, where $\mathcal{N}_s \subset \mathcal{V}$ here denotes the set of 4-connected vertex neighbors of $s \in \mathcal{V}$. Pairwise MRFs are accordingly commonly used to carry out inference over images, such that nodes are made to correspond to pixels and edges to interactions between 4-connected pixel neighbors. Note that such a graph G is an example of what is called a *loopy* graph, since there exists at least one pair of vertices in $\mathcal{V}$ for which there exists more than one sequence of edges in $\mathcal{E}$ connecting the two. By analogy to (3.16), the four constituent terms of (3.24) are given names; specifically, $p(X = x \mid Y = y)$ is called the *posterior*, with $\prod_{s \in \mathcal{V}} \psi_s(x_s; y)$, $\prod_{\{s,t\} \in \mathcal{E}} \psi_{s,t}(x_s, x_t)$, and $Z(y)$ called the *likelihood*, *prior*, and *evidence*, respectively.

3.3 Message Passing on Chains and Trees

The task of computing the marginal distribution $p(X_s = x_s)$ for a variable X_s of a joint probability distribution $p(X_1 = x_1, \ldots, X_N = x_N)$ can be solved exactly by applying the sum rule in (3.3). However, as we saw in Section (3.2), if $|\mathcal{L}_1| = \cdots = |\mathcal{L}_N| = K$, then computing the marginal takes $O(K^N)$ time, exponential in the number N of variables of the joint distribution. In this section, we introduce *message passing* for getting around this exponential complexity on the example of exactly computing a marginal distribution of a joint probability distribution whose corresponding undirected graph takes the form of a chain, a process that takes only $O(NK^2)$ time by exploiting the conditional independence properties encoded in the graph. Proceeding in a similar vein, we show how to obtain a MAP configuration on such a joint distribution, likewise in $O(NK^2)$ time. We then generalize message passing to undirected graphs that take the form of trees—i.e., undirected graphs that are connected and that do not have loops—by introducing the *sum-product* algorithm for computing marginal distributions, and the closely related *max-product* algorithm for obtaining a MAP configuration.

3.3.1 Inference on Chains

A joint probability distribution $p(X_1 = x_1, \ldots, X_N = x_N)$ corresponds to an MRF specified by an undirected graph $G = (\mathcal{V}, \mathcal{E})$ that takes the form of a *chain* if the factorization of the joint distribution over the cliques of G takes the form

$$p(X = x) = \frac{1}{Z} \psi_{1,2}(x_1, x_2) \cdots \psi_{N-1,N}(x_{N-1}, x_N), \tag{3.25}$$

where $x \in \mathcal{L} = \mathcal{L}_1 \times \cdots \times \mathcal{L}_N$. We consider two questions: the question of how to exactly compute the marginal distribution $p(X_x = x_x)$ of any variable X_s of the joint distribution, and that of how to obtain the configuration $x^* \in \mathcal{L}$ that maximizes the joint probability.

Marginals. By the sum rule in (3.3), the marginal distribution $p(X_s = x_s)$ for a variable X_s of the joint distribution in (3.25) is evaluated in $O(K^N)$ time according to

$$p(X_s = x_s) = \frac{1}{Z} \sum_{x_1} \cdots \sum_{x_{s-1}} \sum_{x_{s+1}} \cdots \sum_{x_N} \psi_{1,2}(x_1, x_2) \cdots \psi_{s-1,s}(x_{s-1}, x_s) \psi_{s,s+1}(x_s, x_{s+1}) \cdots \psi_{N-1,N}(x_{N-1}, x_N), \tag{3.26}$$

where $\sum_{x_t}$ is again shorthand for $\sum_{x_t \in \mathcal{L}_t}$. Observe, however, that in (3.26) only the potential $\psi_{N-1,N} : \mathcal{L}_{N-1} \times \mathcal{L}_N \to \mathbb{R}^+$ depends on $x_N \in \mathcal{L}_N$, and so can be replaced by a function $m_\beta : \mathcal{L}_{N-1} \to \mathbb{R}^+$ called a *message*, defined by

$$m_\beta(x_{N-1}) = \sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N), \tag{3.27}$$

which can be encoded as a vector in $\mathbb{R}^{|\mathcal{L}_{N-1}|}$ with one element per $x_{N-1} \in \mathcal{L}_{N-1}$. While there is no requirement that the size of the label set of each variable of the joint distribution be the same, let us again assume that $|\mathcal{L}_1| = \cdots = |\mathcal{L}_N| = K$, in order to derive the claim asserted in the opening paragraph of this section that computing marginals on chains can be carried out by message passing in $O(NK^2)$ time. The cost of computing the message in (3.27) is then determined by evaluating the factor $\psi_{N-1,N}(x_{N-1}, x_N)$ for all K^2 possible configurations $(x_{N-1}, x_N) \in \mathcal{L}_{N-1} \times \mathcal{L}_N$ and then summing over each $x_N \in \mathcal{L}_N$ for $x_{N-1} \in \mathcal{L}_{N-1}$ fixed, and so is performed in $O(K^2)$ time. Observe now that since only $\psi_{N-2,N-1}(x_{N-2}, x_{N-1})\psi_{N-1,N}(x_{N-1}, x_N)$ depends on x_{N-1}, we can replace it in turn with its own message $m_\beta : \mathcal{L}_{N-2} \to \mathbb{R}^+$:

$$\begin{aligned} m_\beta(x_{N-2}) &= \sum_{x_{N-1}} \psi_{N-2,N-1}(x_{N-2}, x_{N-1}) \left(\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right) \\ &= \sum_{x_{N-1}} \psi_{N-2,N-1}(x_{N-2}, x_{N-1}) m_\beta(x_{N-1}), \end{aligned} \tag{3.28}$$

which takes an additional $O(K^2)$ time over the $O(K^2)$ time for precomputing (3.27). Proceeding analogously for all $N-1$ edges in the chain, we find that we can obtain the marginal distribution $p(X_s = x_s)$ in (3.26) recursively by

$$\begin{aligned} p(X_s = x_s) &= \frac{1}{Z} \left(\sum_{x_{s-1}} \psi_{s-1,s}(x_{s-1}, x_s) \left(\cdots \sum_{x_2} \psi_{2,3}(x_2, y_3) \left(\sum_{x_1} \psi_{1,2}(x_1, x_2) \right) \right) \right) \cdot \\ &\quad \left(\sum_{x_{s+1}} \psi_{s,s+1}(x_s, x_{s+1}) \left(\cdots \sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right) \right) \\ &= \frac{1}{Z} \underbrace{\left(\sum_{x_{s-1}} \psi_{s-1,s}(x_{s-1}, x_s) m_\alpha(x_{s-1}) \right)}_{m_\alpha(x_s)} \cdot \underbrace{\left(\sum_{x_{s+1}} \psi_{s,s+1}(x_s, x_{s+1}) m_\beta(x_{s+1}) \right)}_{m_\beta(x_s)}, \end{aligned} \tag{3.29}$$

where the function $m_\beta : \mathcal{L}_s \to \mathbb{R}^+$ is called a *backward* message and the function $m_\alpha : \mathcal{L}_s \to \mathbb{R}^+$ a *forward* message, thus giving an expression for the marginal distribution of X_s that can be interpreted in terms of *message passing* and evaluated in $O(NK^2)$ time, having computed $N-1$ messages in total. This is a significant improvement over the exponential complexity of computing the marginal distribution by naïvely applying the sum rule in (3.3), and is achieved by exploiting the conditional independence properties encoded in the graph. The partition function Z in (3.29) is given by

$$Z = \sum_{x_s \in \mathcal{L}_s} m_\alpha(x_s) m_\beta(x_s). \tag{3.30}$$

Note that independently invoking (3.29) to compute each marginal for all N nodes in the graph takes $O(N^2K^2)$ time, but happens to involve wasteful repeated computation of identical messages. Alternatively, by sweeping the chain forwards to precompute a forward message per node, and proceeding analogously to precompute a backward message per node, computing each marginal for all N nodes requires only twice the number of message computations as to evaluate a single marginal in the manner of (3.29), and thus remains $O(NK^2)$. The partition function Z must be computed only once, for any arbitrary $s \in \mathcal{V}$.

MAP Configurations. Given the same joint distribution $p(X_1 = x_1, \ldots, X_N = x_N)$ whose factorization corresponds to an MRF specified by a chain, it happens to be the case that similar reasoning in terms of messages to that applied to obtaining the marginals of the joint distribution can be used to exactly obtain *a* configuration[2] $x^* \in \mathcal{L} = \mathcal{L}_1 \times \cdots \times \mathcal{L}_N$ that maximizes the joint probability, so that

$$\begin{aligned} x^* &= \arg\max_{x \in \mathcal{L}} p(X = x) \\ &= \arg\max_{x \in \mathcal{L}} \frac{1}{Z} \psi_{1,2}(x_1, x_2) \cdots \psi_{N-1,N}(x_{N-1}, x_N) \\ &= \arg\max_{x \in \mathcal{L}} \quad \psi_{1,2}(x_1, x_2) \cdots \psi_{N-1,N}(x_{N-1}, x_N), \end{aligned} \tag{3.31}$$

which is to say that $p(X = x^*)$ gives the MAP probability of the joint distribution, and so x^* is a *MAP configuration*. Note that (3.31) does not require computation of the partition function, since multiplying a function by a nonzero scalar factor only scales the respective values of the function's extrema, but does not change where those extrema occur. Observe that

$$\begin{aligned} \max_{x \in \mathcal{L}} p(X = x) &= \max_{x \in \mathcal{L}} \psi_{1,2}(x_1, x_2) \cdots \psi_{N-1,N}(x_{N-1}, x_N) \\ &= \max_{x_1} \cdots \max_{x_N} \psi_{1,2}(x_1, x_2) \cdots \psi_{N-1,N}(x_{N-1}, x_N), \end{aligned} \tag{3.32}$$

where $\max_{x_s}$ is shorthand for $\max_{x_s \in \mathcal{L}_s}$. As it was with the scope of the summation $\sum_{x_N}$ in (3.26) for computing marginals, the fact that only the potential $\psi_{N-1,N} : \mathcal{L}_{N-1} \times \mathcal{L}_N \to \mathbb{R}^+$

[2]The subtlety is a noteworthy one: for a given joint distribution $p(X_1 = x_1, \ldots, X_N = x_N)$, there may well exist more than one configuration in $\mathcal{L}_1 \times \cdots \times \mathcal{L}_N$ that maximizes the joint probability.

is in the scope of $\max_{x_N}$ allows for replacing $\max_{x_N} \psi_{x_{N-1},x_N}(x_{N-1}, x_N)$ in (3.32) with the message $m : \mathcal{L}_{N-1} \to \mathbb{R}^+$,

$$m(x_{N-1}) = \max_{x_N} \psi_{x_{N-1},x_N}(x_{N-1}, x_N). \tag{3.33}$$

Proceeding analogously to compute a message for each of the $N-1$ edges in the chain gives the MAP probability recursively as

$$\max_{x \in \mathcal{L}} p(X = x) = \max_{x_1} \underbrace{\left(\max_{x_2} \left(\psi_{1,2}(x_1, x_2) \cdots \max_{x_N} \left(\psi_{N-1,N}(x_{N-1}, x_N) \right) \right) \right)}_{m(x_1)} \tag{3.34}$$

in $O(NK^2)$ time, provided $|\mathcal{L}_1| = \cdots = |\mathcal{L}_N| = K$. The MAP configuration x^* itself can be obtained by having stored the maximizing value of each maximization involved in recursively computing (3.34). Retrieving these stored values is called *back tracking*.

3.3.2 Sum-Product Algorithm

An undirected graph $G = (\mathcal{V}, \mathcal{E})$ is a *tree* if it is connected and if for each pair of vertices $s, t \in \mathcal{V}$, there exists at most one path in $\mathcal{E}$ that joins s with t, which is to say that G does not contain any loops. The message passing algorithm for exactly computing marginals over joint probability distributions that correspond to MRFs specified by undirected graphs that are chains generalizes to joint distributions that correspond to MRFs specified by trees,[3] giving the *sum-product* algorithm. The marginal $p(X_s = x_s)$ for a variable X_s of such a joint probability distribution $p(X_1 = x_1, \ldots, X_N = x_N)$ is then computed, in the same vein as (3.29), by multiplying over all incoming *factor-to-variable* messages $m_{C \to s} : \mathcal{L}_s \to \mathbb{R}^+$ of X_s:

$$p(X_s = x_s) = \frac{1}{Z} \prod_{C \in \mathcal{N}_s} m_{C \to s}(x_s), \tag{3.35}$$

where $\mathcal{N}_s \subset \mathcal{C}(G)$ indexes the clique potentials that are functions of $x_s \in \mathcal{L}_s$. The factor-to-variable messages—from whose form the sum-product algorithm derives its name—are given by

$$m_{C \to s}(x_s) = \sum_{x_C \in \mathcal{L}_C^{x_s}} \left(\psi_C(x_C) \prod_{t \in \mathcal{N}_C \setminus \{s\}} m_{t \to C}(x_{Ct}) \right), \tag{3.36}$$

where $\mathcal{L}_C^{x_s}$ denotes the set of all configurations in $\mathcal{L}_C$ such that $X_s = x_s$ and $x_{Ct} \in \mathcal{L}_t$ denotes the value of X_t as encoded in $x_C \in \mathcal{L}_C$, and $\mathcal{N}_C \subset \mathcal{V}$ indexes the variable nodes over which the clique potential $\psi_C : \mathcal{L}_C \to \mathbb{R}^+$ is defined. The factor-to-variable messages are defined in terms of *variable-to-factor* messages $m_{s \to C} : \mathcal{L}_s \to \mathbb{R}^+$:

$$m_{s \to C}(x_s) = \prod_{C' \in \mathcal{N}_s \setminus \{C\}} m_{C' \to s}(x_s), \tag{3.37}$$

[3] Note that the sum-product algorithm, like the max-product algorithm that we treat next, is applicable to an even larger class of graphs than only to undirected graphs that are trees, namely to tree-structured factor graphs. We omit factor graphs from our discussion for the benefit of simpler and more focused presentation, and instead refer the reader to Bishop [Bis06] or Nowozin and Lampert [NL11] for more details.

which are themselves defined recursively in terms of factor-to-variable messages. Note that to start the recursion, variable-to-factor messages for variables that correspond to leaves of the tree are initialized to 1. The partition function Z in (3.35) is given by

$$Z = \sum_{x_s \in \mathcal{L}_s} \prod_{C \in \mathcal{N}_s} m_{C \to s}(x_s). \tag{3.38}$$

As in the case of chains, proceeding in the above manner separately for each variable in the joint distribution to compute the corresponding marginal involves wasteful repeated computation, since identical messages are recomputed unnecessarily. A less expensive way to proceed is to choose an arbitrary $s \in \mathcal{V}$ and recursively compute each of the incoming messages from the leaves of the tree to s by (3.35). Next, compute outgoing messages from s to the leaves of the tree. In this manner, every node $t \in \mathcal{V}$ will have received a factor-to-variable message for each $C \in \mathcal{N}_t$, thereby giving each marginal in only twice the number of message computations needed to compute a single marginal. As for chains, the partition function Z must be computed only once, for any arbitrary $s \in \mathcal{V}$.

3.3.3 Max-Product Algorithm

Closely related to the sum-product algorithm is the *max-product* algorithm, which—given a joint probability distribution $p(X_1 = x_1, \ldots, X_N = x_N)$ whose factorization corresponds to an MRF specified by a tree—returns a MAP configuration $x^* \in \mathcal{L} = \mathcal{L}_1 \times \cdots \times \mathcal{L}_N$, computed at some arbitrarily chosen node $s \in \mathcal{V}$, again in terms of all incoming factor-to-variable messages $m_{C \to s} : \mathcal{L}_s \to \mathbb{R}^+$ of X_s:

$$\begin{aligned} p(X = x^*) &= \frac{1}{Z} \max_{x_s \in \mathcal{L}_s} \prod_{C \in \mathcal{N}_s} m_{C \to s}(x_s) \\ &\propto \max_{x_s \in \mathcal{L}_s} \prod_{C \in \mathcal{N}_s} m_{C \to s}(x_s), \end{aligned} \tag{3.39}$$

where $\mathcal{N}_s \subset \mathcal{C}(G)$ indexes the clique potentials that are functions of $x_s \in \mathcal{L}_s$. The factor-to-variable messages—from whose form, in turn, the max-product algorithm takes its name—are defined by replacing the summation in (3.36) with a maximization, giving

$$m_{C \to s}(x_s) = \max_{x_C \in \mathcal{L}_C^{x_s}} \left(\psi_C(x_C) \prod_{t \in \mathcal{N}_C \setminus \{s\}} m_{t \to C}(x_{Ct}) \right), \tag{3.40}$$

where $\mathcal{L}_C^{x_s}$ again denotes the set of all configurations in $\mathcal{L}_C$ such that $X_s = x_s$ and $x_{Ct} \in \mathcal{L}_t$ denotes the value of X_t as encoded in $x_C \in \mathcal{L}_C$, and $\mathcal{N}_C \subset \mathcal{V}$ indexes the variable nodes over which the clique potential $\psi_C : \mathcal{L}_C \to \mathbb{R}^+$ is defined. The variable-to-factor messages $m_{s \to C} : \mathcal{L}_s \to \mathbb{R}^+$ in terms of which (3.40) is defined are themselves defined in the same manner as in (3.37). As in the sum-product algorithm, variable-to-factor messages for variables that correspond to leaves of the tree likewise initialized to 1 to start the recursion. The MAP configuration x^* itself can be obtained by having stored the maximizing value of each maximization involved in recursively computing (3.39).

MAP Inference as Energy Minimization. If $a > b$ holds then so too does $\log a > \log b$, and so it follows that finding a configuration that *maximizes* the joint probability can be cast in terms of energy *minimization* in log-space:

$$\begin{aligned}
\arg\max_{x\in\mathcal{L}} \log\big(p(X=x)\big) &= \arg\max_{x\in\mathcal{L}} \log\left(\frac{1}{Z}\prod_{C\in\mathcal{C}(G)}\psi_C(x_C)\right) \\
&= \arg\max_{x\in\mathcal{L}} \log\left(\prod_{C\in\mathcal{C}(G)}\psi_C(x_C)\right) \\
&= \arg\max_{x\in\mathcal{L}} \log\left(\prod_{C\in\mathcal{C}(G)}\exp\big(-E_C(x_C)\big)\right) \\
&= \arg\max_{x\in\mathcal{L}} \log\left(\exp\left(-\sum_{C\in\mathcal{C}(G)}E_C(x_C)\right)\right) \\
&= \arg\max_{x\in\mathcal{L}} \quad -\sum_{C\in\mathcal{C}(G)}E_C(x_C) \\
&= \arg\min_{x\in\mathcal{L}} \quad \sum_{C\in\mathcal{C}(G)}E_C(x_C) \\
&= \arg\min_{x\in\mathcal{L}} \quad E(x), \qquad (3.41)
\end{aligned}$$

where the factors $\psi_C(x_C)$ are defined as in (3.20) in terms of respective *energy functions* $E_C(x_C)$ and $E(x) = \sum_{C\in\mathcal{C}(G)} E_C(x_C)$. Expressing (3.39) as a minimization in log-space gives

$$\min_{x\in\mathcal{L}}\sum_{C\in\mathcal{C}(G)}E_C(x_C) = \min_{x_s\in\mathcal{L}_s}\sum_{C\in\mathcal{N}_s}M_{C\to s}(x_s), \qquad (3.42)$$

where the log-space factor-to-variable messages $M_{C\to s} : \mathcal{L}_C \to \mathbb{R}$ are given by

$$M_{C\to s}(x_s) = \min_{x_C\in\mathcal{L}_C^{x_s}}\left(E_C(x_C) + \sum_{t\in\mathcal{N}_C\setminus\{s\}}M_{t\to C}(x_{Ct})\right), \qquad (3.43)$$

with the log-space variable-to-factor messages $M_{s\to C} : \mathcal{L}_C \to \mathbb{R}$ in terms of which (3.43) is defined given by

$$M_{s\to C}(x_s) = \sum_{C'\in\mathcal{N}_s\setminus\{C\}}M_{C'\to s}(x_s). \qquad (3.44)$$

In order to start the recursion, the log-space variable-to-factor messages that correspond to leaf nodes of the tree are initialized to $\log(1) = 0$. Practically speaking, reasoning in log-space provides an avenue to ameliorating problems of numerical underflow potentially caused by multiplying together many small probabilities in the process of computing (3.39).

3.4 Max-Product Loopy Belief Propagation

Of particular interest in computer vision and in this thesis is the recovery of MAP configurations for joint probability distributions whose factorization takes the form (3.24), which we rewrite here with the conditioning expressed implicitly in the factors:

$$p(X = x) = \frac{1}{Z} \prod_{s \in \mathcal{V}} \psi_s(x_s) \prod_{\{s,t\} \in \mathcal{E}} \psi_{s,t}(x_s, x_t), \tag{3.45}$$

where $\mathcal{E} = \bigcup_{s \in \mathcal{V}} \big\{ \{s,t\} \mid t \in \mathcal{N}_s \big\}$, and where in turn $\mathcal{N}_s \subset \mathcal{V}$ denotes the set of 4-connected vertex neighbors of $s \in \mathcal{V}$ given an arrangement of the vertices in $\mathcal{V}$ as a regular 2D grid. The corresponding MRF—called a pairwise MRF—is specified by a graph that contains loops, since for any pair of vertices $s, t \in \mathcal{V}$ there exists more than one path in $\mathcal{E}$ joining the two. While the max-product algorithm presented in Section 3.3 is guaranteed to give a MAP configuration only for graphs that do not contain loops, applying message passing to loopy graphs has been observed in many settings to be an effective heuristic for approximative MAP inference. More concretely, a MAP configuration for (3.45) is estimated iteratively by maximizing the *belief* $b_s^n : \mathcal{L}_s \to \mathbb{R}^+$ at each node $s \in \mathcal{V}$ for iteration n:

$$b_s^n(x_s) = \psi_s(x_s) \prod_{t \in \mathcal{N}_s} m_{t \to s}^n(x_s), \tag{3.46}$$

computed in terms of incoming messages $m_{t \to s}^n : \mathcal{L}_s \to \mathbb{R}^+$, defined recursively by

$$\begin{aligned} m_{t \to s}^n(x_s) &= \max_{x_t \in \mathcal{L}_t} \left(\psi_t(x_t) \psi_{s,t}(x_s, x_t) \prod_{u \in \mathcal{N}_t \setminus \{s\}} m_{u \to t}^{n-1}(x_t) \right) \\ &= \max_{x_t \in \mathcal{L}_t} \left(\psi_t(x_t) \psi_{s,t}(x_s, x_t) \prod_{u \in \mathcal{N}_t} m_{u \to t}^{n-1}(x_t) \frac{1}{m_{s \to t}^{n-1}(x_t)} \right) \\ &= \max_{x_t \in \mathcal{L}_t} \left(\psi_{s,t}(x_s, x_t) \frac{b_t^{n-1}(x_t)}{m_{s \to t}^{n-1}(x_t)} \right). \end{aligned} \tag{3.47}$$

Note that the incoming messages in (3.46) *serve as weights on the possible values* $x_s \in \mathcal{L}_s$ of X_s. The messages are initialized for $n = 0$ to 1 for each $x_s \in \mathcal{L}_s$, and are stored after being computed in order to prevent wasteful recomputation. Updating the beliefs in (3.46) calls for deciding on a *schedule* according to which the nodes $s \in \mathcal{V}$ are visited over the course of an iteration; for example, the classical PatchMatch schedule has nodes visited in scanline order from the upper-left to the lower-right for odd iterations n, and in the opposite order for iterations when n is even. The estimated MAP configuration $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_N) \in \mathcal{L} = \mathcal{L}_1 \times \cdots \times \mathcal{L}_N$, after $n_{\max}$ iterations, is then obtained by evaluating

$$\hat{x}_s = \arg \max_{x_s \in \mathcal{L}_s} b_s^{n_{\max}}(x_s) \tag{3.48}$$

for each node $s \in \mathcal{V}$.

Energy Formulation. Casting the problem of MAP inference for a joint distribution that factorizes as (3.45) in terms of energy minimization in log-space, the objective becomes to find a configuration $x^* \in \mathcal{L}$ that minimizes the energy function $E : \mathcal{L} \to \mathbb{R}$, which is defined by

$$E(x) = \sum_{s \in \mathcal{V}} E_s(x_s) + \sum_{\{s,t\} \in \mathcal{E}} E_{s,t}(x_s, x_t), \tag{3.49}$$

where $\mathcal{E}$ again denotes the set $\bigcup_{s \in \mathcal{V}} \big\{\{s,t\} \mid t \in \mathcal{N}_s\big\}$ of all 4-connected neighbors. Accordingly, expressed in log-space the belief $b_s^n : \mathcal{L}_s \to \mathbb{R}^+$ in (3.46) becomes the *log-disbelief* $B_s^n : \mathcal{L}_s \to \mathbb{R}$:

$$B_s^n(x_s) = E_s(x_s) + \sum_{t \in \mathcal{N}_s} M_{t \to s}^n(x_s), \tag{3.50}$$

where $E_s(x_s) = -\log\big(\psi_s(x_s)\big)$. The log-disbelief at a node $s \in \mathcal{V}$ for iteration n is computed in terms of incoming log-space messages $M_{t \to s}^n : \mathcal{L}_s \to \mathbb{R}$—based on (3.47)—that are defined recursively according to

$$M_{t \to s}^n(x_s) = \min_{x_t \in \mathcal{L}_t} \Big(E_{s,t}(x_s, x_t) + B_t^{n-1}(x_t) - M_{s \to t}^{n-1}(x_t) \Big), \tag{3.51}$$

where, similarly, $E_{s,t}(x_s, x_t) = -\log\big(\psi_{s,t}(x_s, x_t)\big)$. The log-space messages are initialized for $n = 0$ to $\mathbf{0} \in \mathbb{R}^{|\mathcal{L}_s|}$. After $n_{\max}$ iterations, the estimated minimizer $\hat{x} \in \mathcal{L}$ of the energy in (3.49) is then obtained by computing the minimization

$$\hat{x}_s = \arg\min_{x_s \in \mathcal{L}_s} B_s^{n_{\max}}(x_s), \tag{3.52}$$

again for each node $s \in \mathcal{V}$.

Max-Product PBP. Note that choosing a fixed set $\mathcal{L}_s = \{x_s^1, \ldots, x_s^K\}$ of K possible labels for each node $s \in \mathcal{V}$ might of necessity call for a coarse discretization of the label space, in order to take into account considerations of runtime or memory for inference. Rather than keep each set $\mathcal{L}_s$ fixed, the max-product *particle belief propagation* (PBP) variant [KPS11] of max-product belief propagation maintains, for each node $s \in \mathcal{V}$, a set $\mathcal{P}_s^n = \{x_s^{(1)}, \ldots, x_s^{(P)}\}$ of P labels called *particles*, which are sampled from an underlying continuous label space. Each such particle is allowed to undergo change between successive iterations in what amounts to a form of Markov Chain Monte Carlo (MCMC) sampling *around* the given particle. Restricting our attention to the formulation of max-product PBP in terms of energy minimization in log-space, the messages $M_{t \to s}^n : \mathcal{L}_s \to \mathbb{R}$ defined in (3.51) are instead defined in terms of $M_{t \to s}^n : \mathcal{P}_s^n \to \mathbb{R}$, such that

$$M_{t \to s}^n(x_s) = \min_{x_t \in \mathcal{P}_t^{n-1}} \Big(E_{s,t}(x_s, x_t) + B_t^{n-1}(x_t) - M_{s \to t}^{n-1}(x_t) \Big), \tag{3.53}$$

where the minimization is carried out over the particle set assigned to $t \in \mathcal{V}$ in the *previous* iteration, since it is over the particle sets of the previous iteration that the messages of the previous iteration—in terms of which (3.53) is in turn calculated—were themselves computed and stored.

Note that if $|\mathcal{P}_t^{n-1}| = 1$, the incoming messages of node t have no impact on the minimization in (3.53), thereby allowing for (3.53) to be replaced with

$$M_{t\to s}^n(x_s) = E_t(x_t) + E_{s,t}(x_s, x_t), \tag{3.54}$$

where x_t is the single member of $\mathcal{P}_t^{n-1}$. The log-disbeliefs $B_s^n : \mathcal{P}_s^n \to \mathbb{R}$ are computed as in (3.50), with the difference that the domain of B_s^n becomes $\mathcal{P}_s^n$ in order to match the domain $\mathcal{P}_s^n$ over which the messages defined in (3.53) are computed and stored. After $n_{\max}$ iterations, the estimated minimizer $\hat{x} \in \mathcal{P}_1^{n_{\max}} \times \cdots \times \mathcal{P}_N^{n_{\max}}$ is obtained, for each $s \in \mathcal{V}$, from (3.52) by

$$\hat{x}_s = \arg \min_{x_s \in \mathcal{P}_s^{n_{\max}}} B_s^{n_{\max}}(x_s). \tag{3.55}$$

Let $\mathcal{S}$ denote a set and $f(\mathcal{S}) = \{f(s) \mid s \in \mathcal{S}\}$ the application of a function $f : \mathcal{S} \to \mathbb{R}$ to each of the set's members $s \in \mathcal{S}$. Additionally, let $\phi^n : \mathcal{V} \to \mathbb{Z}$ denote a *schedule function*, so that $t \in \mathcal{V}$ is visited before $s \in \mathcal{V}$ if $\phi^n(t) < \phi^n(s)$, and let $\Phi_s^n = \{t \mid \phi^n(t) < \phi^n(s), t \in \mathcal{V}\}$ denote the corresponding *predecessor set* of $s \in \mathcal{V}$ for iteration n. Making use of these terms, we give pseudocode for max-product PBP in Algorithm 3.1, juxtaposed with the pseudocode of the PMBP and PatchMatch algorithms that we finally describe in the section that follows.

3.5 PatchMatch and PMBP

Closely related to the max-product PBP algorithm from Section 3.4—intended for minimizing continuous labeling problems defined over both unary and pairwise terms—is the PMBP energy minimization algorithm of Besse et al. [BRFK12], with the major difference with respect to max-product PBP being the integration of a spatial propagation step borrowed from PatchMatch. The PatchMatch algorithm of Barnes et al. [BSFG09, BSGF10] was first introduced as a method for obtaining dense approximate nearest neighbor (ANN) fields given a pair of images, assigning to each pixel in one image a label drawn from a continuous label space mapping the patch centered on that pixel to a matching patch in the other image. PatchMatch comprises a random (or semi-random) *initialization* step followed by a number of iterations of *spatial propagation* (a form of sampling from neighboring nodes) coupled with *refinement* (a form of resampling at a given node, drawing samples around the current label assignment). In each step of PatchMatch, a candidate label is adopted at a given pixel if doing so yields a lesser unary matching cost with respect to the label currently assigned. While the majority of (semi-)random label assignments are likely to be incorrect, the PatchMatch algorithm is quick to converge in practice owing especially to the fact that the spatial propagation step serves to exploit spatial coherence characteristic of typical correspondence fields by carrying out what amounts to a form of seed growing. However, the PatchMatch algorithm is only a unary optimizer and as such cannot call on pairwise terms to explicitly promote smoothness of the output labeling, which is the aim of PMBP. We present pseudocode for PMBP and PatchMatch in Algorithm 3.1, juxtaposed with the pseudocode for max-product PBP.

Log-disbeliefs. Computation of log-disbeliefs $B_s^n(x_s)$ in PMBP differs from that in max-product PBP in that the messages $M_{t\to s}^n(x_s)$ at iteration n are computed not based strictly on the

messages $M_{t\to s}^{n-1}(x_s)$ of the preceding iteration $n-1$ as in (3.53), but rather in terms of the most recent available messages. Accordingly, the log-disbelief $B_s^n(x_s)$ of a typical node $s \in \mathcal{V}$ is computed in terms of incoming messages themselves computed at iteration n from the nodes in $\mathcal{N}_s \cap \Psi_s^n$, in addition to incoming messages computed at $n-1$ from the nodes in $\mathcal{N}_s \setminus \{\mathcal{N}_s \cap \Psi_s^n\}$. To proceed in this manner in computing log-disbeliefs is arguably closer in spirit to PatchMatch than is the case in (3.53).

PMBP as PatchMatch. Note that having set all pairwise terms to naught, the PMBP algorithm reduces to the PatchMatch algorithm even though the incoming messages in terms of which the log-disbelief B_s^n in (3.50) at a node $s \in \mathcal{V}$ is computed remain non-zero in general. This is because in the absense of pairwise terms, messages in (3.53) become independent of the value of $x_s \in \mathcal{P}_s^n$:

$$M_{t\to s}^n(x_s) = \min_{x_t \in \mathcal{P}_t^{n-1}} \Big(B_t^{n-1}(x_t) - M_{s\to t}^{n-1}(x_t) \Big), \tag{3.56}$$

and so the minimum of B_s^n over the labels in $\mathcal{P}_s^n$ is determined uniquely by the value of E_s. Note that the number of particles P per node is understood to be 1 in classical PatchMatch.

Algorithm 3.1 Max-Product PBP (A), PMBP (B), and PatchMatch (C)

	A	B	C	
1:	•	•	•	**for each** node $s \in \mathcal{V}$ **do**
2:	•	•	•	initialize $\mathcal{P}_s^0$ with P particles, drawn from some proposal distribution
3:	•	•	-	**for each** neighbor node $t \in \mathcal{N}_s$ **do**
4:	•	•	-	initialize each message $M_{t \to s}^0$ to $\mathbf{0} \in \mathbb{R}^P$
5:	•	•	-	**end for**
6:	•	•	•	**end for**
7:	•	•	•	**for each** iteration $n \in \{1, \ldots, n_{\max} - 1\}$ **do**
8:	•	•	•	**for each** node $s \in \mathcal{V}$, ordered by ϕ_s^n **do**
9:	•	•	•	$\mathcal{P}_s^n \leftarrow \mathcal{P}_s^{n-1}$
10:	-	•	•	**for each** neighbor particle $x_t \in \bigcup_{t \in \mathcal{N}_s \cap \Psi_s^n} \mathcal{P}_t^n$ **do**
11:	-	•	-	$x_s \leftarrow \arg\max_{x_s \in \mathcal{P}_s^n} B_s^n(\mathcal{P}_s^n)$
12:	-	-	•	$x_s \leftarrow \arg\max_{x_s \in \mathcal{P}_s^n} E_s(\mathcal{P}_s^n)$
13:	-	•	-	**if** $B_s^n(x_t) < B_s^n(x_s)$ **then**
14:	-	-	•	**if** $E_s(x_t) < E_s(x_s)$ **then**
15:	-	•	•	replace x_s in $\mathcal{P}_s^n$ with x_t {spatial propagation}
16:	-	•	•	**end if**
17:	-	•	•	**end for**
18:	•	•	•	**for each** particle $x_s \in \mathcal{P}_s^n$ **do**
19:	•	•	•	$x_s^{\langle 0 \rangle} \leftarrow x_s$
20:	•	•	•	**for each** iteration $m \in \{1, \ldots, m_{\max}\}$ **do**
21:	•	•	•	$x_s^{\langle m \rangle} \leftarrow x_s^{\langle m-1 \rangle}$
22:	•	•	•	$\bar{x}_s^{\langle m \rangle} \leftarrow$ sample around $x_s^{\langle m \rangle}$
23:	•	-	-	**if** $B_s^n(\bar{x}_s^{\langle m \rangle}) < B_s^n(x_s^{\langle m \rangle}) - \log(\text{rand})$ **then**
24:	-	•	-	**if** $B_s^n(\bar{x}_s^{\langle m \rangle}) < B_s^n(x_s^{\langle m \rangle})$ **then**
25:	-	-	•	**if** $E_s(\bar{x}_s^{\langle m \rangle}) < E_s(x_s^{\langle m \rangle})$ **then**
26:	•	•	•	$x_s^{\langle m \rangle} \leftarrow \bar{x}_s^{\langle m \rangle}$
27:	-	•	•	replace x_s in $\mathcal{P}_s^n$ with $x_s^{\langle m \rangle}$ {refinement}
28:	•	•	•	**end if**
29:	•	•	•	**end for**
30:	•	-	-	replace x_s in $\mathcal{P}_s^n$ with $x_s^{\langle m_{\max} \rangle}$
31:	•	•	•	**end for**
32:	•	•	•	**end for**
33:	•	•	•	**end for**
34:	•	•	•	**for each** node $s \in \mathcal{V}$ **do**
35:	•	•	-	$\hat{x}_s \leftarrow \arg\min_{x_s \in \mathcal{P}_s^{n_{\max}}} B_s^{n_{\max}}(x_s)$ {output}
36:	-	-	•	$\hat{x}_s \leftarrow \arg\min_{x_s \in \mathcal{P}_s^{n_{\max}}} E_s(x_s)$ {output}
37:	•	•	•	**end for**

CHAPTER 4

Literature Review

This chapter aims to outline the major trends from across the field of computer vision in what concerns correspondence search. Absent modeling every process involved in giving rise to an image, any algorithm designed with the intention of recovering correspondences in a scene from data in images will make simplifying assumptions—whether implicitly or by design—about those underlying processes. These simplifying assumptions are useful if they render matching simpler, faster, or more robust, but come at the cost of restricting the scenarios about which one can hope to reason using a given algorithm. Two of the most common such assumptions permeating the literature in what concerns matching are the assumptions of *brightness constancy* and of *local surface planarity*, both of which tend to break down as displacements become large. It is in part for this reason that matching at large displacements has traditionally been the domain of sparse methods—which are the subject of Section 4.1—where matching is restricted only to sparse *keypoints* described in terms of *features* designed to offer some invariance to changes in appearance that arise at large displacements; another is the rapid growth of the space of possible displacements to consider as displacements become large, as introduced in Section 1.1. In the sections that follow we consider dense methods, which differ with respect to one another to a large extent in what knowledge about the scene is available and in how that knowledge is incorporated. For a pair of images, we begin with the most general in Section 4.2, where we treat the problem of computing optical flow from a pair of images of a scene, where the aim is to recover, for each pixel in one image of the pair, the 2D translation vector mapping the pixel to its correspondence in the other image. In Section 4.3 we treat the problem of stereo matching, which differs from computing optical flow primarily in that the motion in the two images—called a stereo pair—is known to be due strictly to the 3D rigid motion of the camera. In Section 4.4 we provide a treatment of the recovery of scene flow, which is the 3D translational motion that gives rise to 2D optical flow in the image plane, and which is typically computed over either a pair of stereo pairs or a pair of RGB-D frames.

4.1 Sparse Matching

The classical two-image sparse matching pipeline involves first (i) identifying, per image, a set of pixels of interest termed *keypoints* (also known as *corners* or *interest points*), (ii) computing respective *descriptors* (or *features*) to characterize each of those keypoints, and finally (iii) matching features between the two images in the aim of recovering correspondences from among keypoint pairs. Such keypoints tend to be defined as pixels where image properties—perhaps image intensity, color, or texture—differ in some sense from their immediate neighborhood, ideally focusing attention on locations in the image that are in some sense unique and thus easier to match. In order to handle large displacements, an ideal sparse matching pipeline would in each step be invariant to image space appearance changes characteristic of large displacements, which is to say invariant to non-linear changes in color (cf. Figure 1.2) and to projective distortions over arbitrary geometry undergoing 3D motion in the scene (cf. Figure 1.4). An immediate advantage in principle of sparse matching is that by in effect discretizing the search space by focusing attention only on possible pairs of sparse keypoints, the search space can remain manageable—allowing, perhaps, for even *exhaustively comparing all possible keypoint pairs*—even as displacements become large.

Keypoint Detectors. An effective way to determine whether a keypoint situated at a pixel $\mathbf{x} = (x, y)^\top$ can be expected to be *stable* under translations if the $n \times n$ patch centered on the keypoint compared against itself under a variety small translational displacements yields a small cumulative dissimilarity. Among the oldest keypoint detectors widely in use today is the venerable Harris corner detector [HS88], which computes local image intensity gradients at every pixel and identifies those pixels with strong gradients in two distinct directions, and thus offers invariance to image space translations and rotations. The Harris-Laplace, Hessian-Laplace, and Difference of Gaussians (DoG) keypoint detectors additionally offer invariance to scale changes [MS04a, MS01, Low04]. Other keypoint detectors, including Harris-affine or Hessian-affine [MS02], and Maximally Stable Extremal Regions (MSER) [MCUP04] offer invariance to affine distortions; a comparison of affine keypoint detectors is provided in Mikolajczyk et al. [MTS$^+$05].[1]

Keypoint Descriptors and Matching. Having identified keypoints, the simplest keypoint descriptor is a vector of image pixels corresponding to an $n \times n$ patch centered on the keypoint, with matching carried out with respect to some elementwise (dis)similarity measure over vector pairs. The Scale Invariant Feature Transform (SIFT) of Lowe [Low04] combines a keypoint detector based on DoG that is invariant to rotation, translation, and scale, with a descriptor that offers partial invariance to illumination changes, noise, and affine distortion in addition to invariance to rotation, translation, and scale. Variants of SIFT include PCA-SIFT [KS04] and Speeded Up Robust Features (SURF) [BTG06]. More recently the Affine SIFT (ASIFT) algorithm of Morel and Yu [MY09] promises full invariance to affine distortions. Combining sparse matching that offers robustness to affine motions and enforcement of the epipolar constraint (cf. Section 2.5)

[1] Supporting publications and code are available at `http://www.robots.ox.ac.uk/~vgg/research/affine/`.

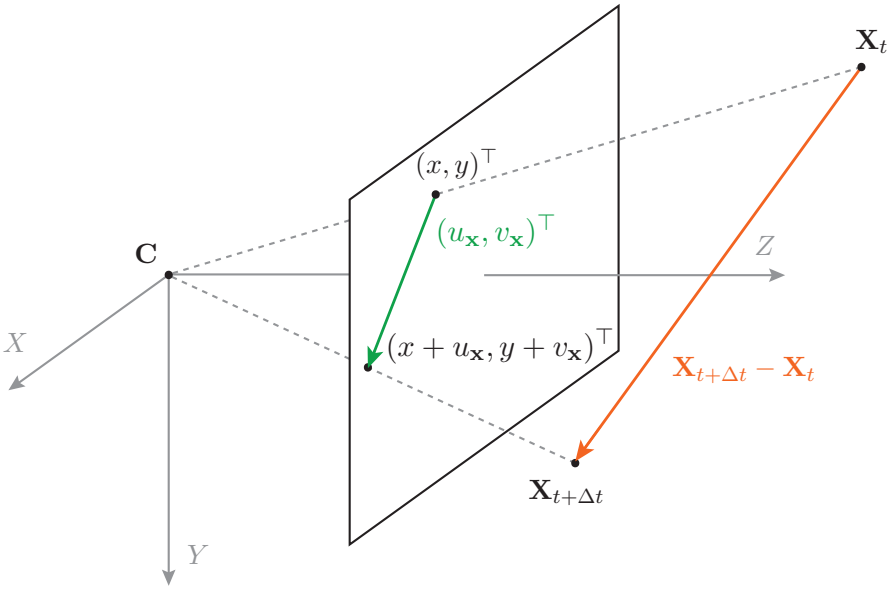


Figure 4.1: The relationship between optical flow (green) and scene flow (orange). The pixel $(x, y)^\top$ is the projection of the point $\mathbf{X}_t \in \mathbb{R}^3$ at time t. At time $t + \Delta t$, the corresponding point has moved to $\mathbf{X}_{t+\Delta t} \in \mathbb{R}^3$, which projects in turn to the pixel $(x+u_\mathbf{x}, y+v_\mathbf{x})^\top$. The vector $\mathbf{X}_{t+\Delta t} - \mathbf{X}_t$ is the *scene flow* at $I(x, y, t)$ with respect to the image $I(t + \Delta t)$; the vector $(u_\mathbf{x}, v_\mathbf{x})^\top = (x + u_\mathbf{x}, y + v_\mathbf{x})^\top - (x, y)^\top$ is the *optical flow* at $I(x, y, t)$ with respect to the image $I(t + \Delta t)$, and is the projection to image space of the scene flow.

gives rise to what amounts to a fast form of stereo matching suited to recovering sparse correspondences that arise from wide-baseline camera motion in a static scene [MOC02, MCUP04]. In Chapter 6, we seed our 6 DoF dense correspondence fields with sparse matches, promoting the propagation of good initial matches in the aim of avoiding local minima; in Chapter 7, we obtain seeds for our 9 DoF correspondence fields by additionally considering sparse matches obtained by enforcing the epipolar constraint with the advantage of allowing for triangulating respective pairs of matches (cf. Figure 7.2).

4.2 Optical Flow

Let $\big(I(t), I(t + \Delta t)\big)$ denote an ordered pair of images of a scene acquired at time steps t and $t + \Delta t$, respectively. Let $I(x, y, t)$ denote the image intensity in image $I(t)$ at pixel $\mathbf{x} = (x, y)^\top$, and let $\mathbf{X}_t \in \mathbb{R}^3$ denote the 3D point that gives rise to $I(x, y, t)$. Additionally, let $I(x + u_\mathbf{x}, y + v_\mathbf{x}, t + \Delta t)$ denote the 2D correspondence of $I(x, y, t)$ in image $I(t + \Delta t)$, itself the projection of a point $\mathbf{X}_{t+\Delta t} \in \mathbb{R}^3$ that is the 3D correspondence at $t + \Delta t$ of $\mathbf{X}_t$. The *optical flow* $(u_\mathbf{x}, v_\mathbf{x})^\top$ at $\mathbf{x}$ in $I(t)$ with respect to the image $I(t + \Delta t)$ describes the 2D motion

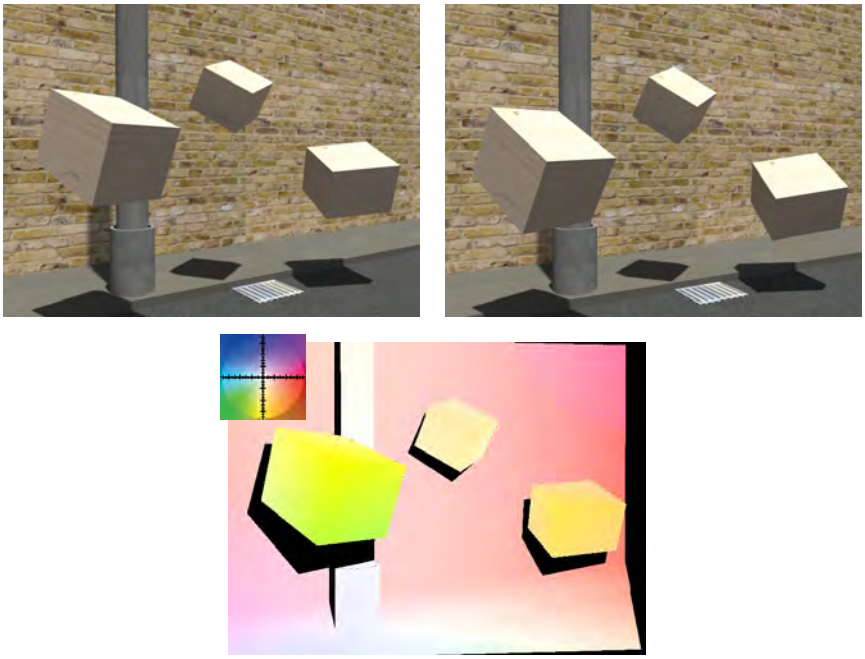


Figure 4.2: 2D flow coloring. 2D flow is typically visualized by coloring each pixel according to the above color wheel, where hue and saturation represent the direction and magnitude, respectively, of the pixel's 2D flow vector. Shown is the ground truth optical flow corresponding to the first frame of the street1Txtr1 image pair of the UCL optical flow data set (black indicates pixels that are occluded in the second frame).

in the image plane between $(x, y)^\top$ at t and its correspondence $(x + u_{\mathbf{x}}, y + v_{\mathbf{x}})^\top$ at $t + \Delta t$.[2] Accordingly, this optical flow at $\mathbf{x}$ can be thought of as the projection of the underlying 3D displacement $\mathbf{X}_{t+\Delta t} - \mathbf{X}_t$ (cf. Figure 4.1), which is in turn termed the *scene flow* [VBR+05]. Such 2D flow is typically visualized by coloring each pixel according to a color wheel such that hue and saturation represent the direction and magnitude, respectively, of the pixel's flow vector [Hor86] (cf. Figure 4.2). The two pioneering techniques in optical flow are the work of Lucas and Kanade [LK81] and that of Horn and Schunck [HS81], both of which are intensity gradient-based methods. The basic assumption underlying both methods is the assumption of brightness constancy, which states that correspondences share the same intensity:

$$I(x, y, t) = I(x + u_{\mathbf{x}}, y + v_{\mathbf{x}}, t + \Delta t). \tag{4.1}$$

[2] Note that while this is the usage of Baker et al. [BSL+11] for 'ground truth flow' in the Middlebury optical flow benchmark, Horn [Hor86] defines optical flow as the *apparent* motion of brightness patterns in the image plane, and refers to the projection of 3D translation vectors relating points in scene instead as the *motion field* of an image. We drop this arguably dated distinction and simply understand the goal of computing the optical flow to be the recovery of the motion field.

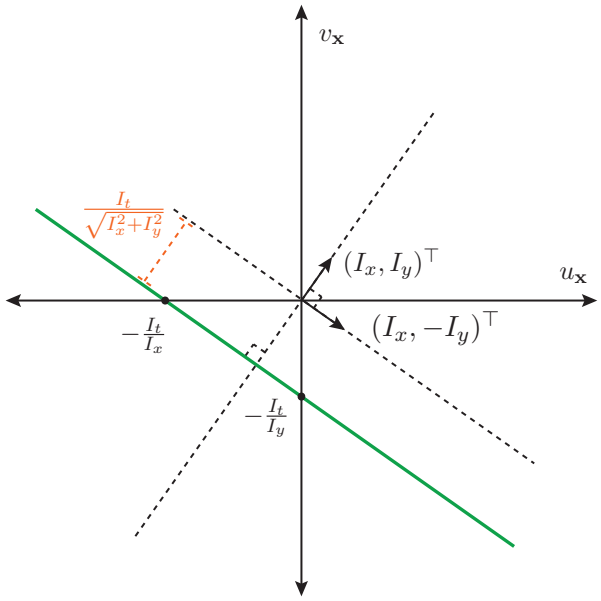


Figure 4.3: The optical flow constraint. The partial derivatives I_x, I_y, I_t constrain the optical flow $(u_{\mathbf{x}}, v_{\mathbf{x}})^\top$ at a pixel $\mathbf{x} = (x, y)^\top$ to lie on a line (green) orthogonal to the intensity gradient vector $(I_x, I_y)^\top$. The component of the optical flow along $(I_x, I_y)^\top$ is called the normal flow, and is fully determined by I_x, I_y, I_t; the component along the direction $(I_x, -I_y)^\top$ orthogonal to the intensity gradient vector remains free. It is for this reason that relying on brightness constancy alone at a single pixel renders the problem of computing the optical flow ill-posed.

The optical flow can be expressed—when displacements are small—in terms of a first-order Taylor expansion applied to the right hand side of (4.1) with higher order terms omitted:

$$I(x, y, t) = I(x, y, t) + \underbrace{\frac{\partial I(x, y, t)}{\partial x}}_{I_x} \Delta x + \underbrace{\frac{\partial I(x, y, t)}{\partial y}}_{I_y} \Delta y + \underbrace{\frac{\partial I(x, y, t)}{\partial t}}_{I_t} \Delta t, \tag{4.2}$$

which in turn gives the *optical flow constraint* (or *gradient constraint equation*)

$$I_x \cdot u_{\mathbf{x}} + I_y \cdot v_{\mathbf{x}} + I_t = (I_x, I_y, I_t)(u_{\mathbf{x}}, v_{\mathbf{x}}, 1)^\top = 0, \tag{4.3}$$

where $u_{\mathbf{x}} = \Delta x / \Delta t$ and $v_{\mathbf{x}} = \Delta y / \Delta t$, thereby imposing a single linear constraint on the two unknowns of the optical flow $(u_{\mathbf{x}}, v_{\mathbf{x}})^\top$ at $\mathbf{x}$. Geometrically, (4.3) constrains the flow $(u_{\mathbf{x}}, v_{\mathbf{x}})^\top$ to lie on a line $(I_x, I_y, I_t)^\top \in \mathbb{P}^2$ (cf. Figure 4.3 and Section 2.1). The component of the vector $(u_{\mathbf{x}}, v_{\mathbf{x}})^\top$ along the intensity gradient vector $(I_x, I_y)^\top$ is called the *normal flow*, with magnitude

$$\frac{I_t}{\sqrt{I_x^2 + I_y^2}} \tag{4.4}$$

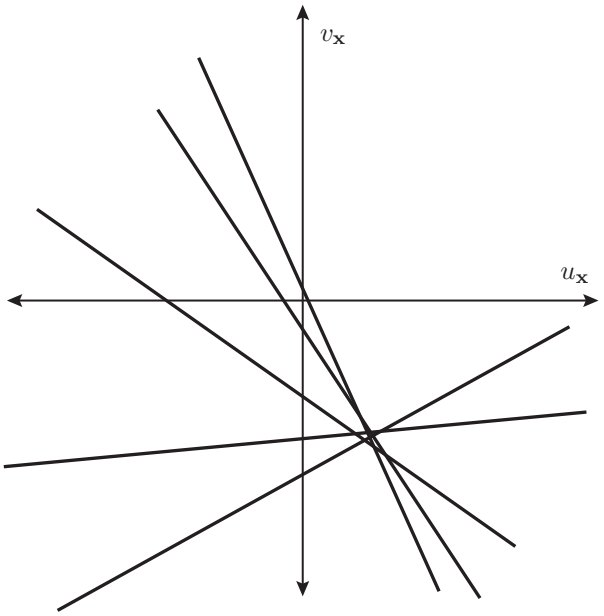


Figure 4.4: Independently for each pixel $\mathbf{x}$, the Lucas-Kanade method [LK81] solves for the optical flow $(u_{\mathbf{x}}, v_{\mathbf{x}})^\top$ by setting up, for the pixels $\mathbf{x}_i$ in a patch $\mathcal{W}_{\mathbf{x}}$ centered on $\mathbf{x}$, a system of linear equations using the optical flow constraint in (4.3). This gives $(u_{\mathbf{x}}, v_{\mathbf{x}})^\top$ as an optimal point of intersection of the respective optical flow constraint lines corresponding to the pixels in the patch $\mathcal{W}_{\mathbf{x}}$.

fully determined by I_x, I_y, I_t. The component of the optical flow in the direction $(I_x, -I_y)^\top$ orthogonal to the intensity gradient vector remains free. It is because only the normal flow can be uniquely determined that relying on the brightness constancy constraint alone renders the problem of solving for the optical flow $(u_{\mathbf{x}}, v_{\mathbf{x}})^\top$ ill-posed. The Lucas-Kanade method is a *local* technique in that it overcomes this ill-posedness of computing the optical flow, individually for each pixel $\mathbf{x}$, by aggregating constraints locally in a patch (or *window*) centered on $\mathbf{x}$; in contrast, the Horn-Schunck method is a *global* technique in the sense that recovering the optical flow is cast in terms of a energy minimization problem over data and smoothness terms encompassing the entirety of the image.

Lucas-Kanade. The Lucas-Kanade method [LK81] overcomes the ill-posedness in computing the flow $(u_{\mathbf{x}}, v_{\mathbf{x}})^\top$ at a pixel $\mathbf{x} = (x, y)^\top$ by seeking, independently for each pixel $\mathbf{x}$, an optimal (in a least squares sense) point of intersection to the optical flow constraint lines corresponding, respectively, to each of the pixels $\mathbf{x}_i \in \mathcal{W}_{\mathbf{x}}$ in an $n \times n$ patch centered on $\mathbf{x}$, as illustrated in Figure 4.4. The Lucas-Kanade method is thus a local method, driven by the assumption that the pixels in the patch undergo similar motion in the image plane. Expressed more formally, the aim is to find, independently for each pixel $I(x, y, t)$, a 2D flow vector $(u_{\mathbf{x}}, v_{\mathbf{x}})^\top$ that minimizes the

objective function

$$E(u_{\mathbf{x}}, v_{\mathbf{x}}) = \sum_{\mathbf{x}_i \in \mathcal{W}_{\mathbf{x}}} \Big(I_x(\mathbf{x}_i) \cdot u_{\mathbf{x}} + I_y(\mathbf{x}_i) \cdot v_{\mathbf{x}} + I_t(\mathbf{x}_i) \Big)^2, \tag{4.5}$$

where $I_x(\mathbf{x}_i), I_y(\mathbf{x}_i), I_t(\mathbf{x}_i)$ are the partial derivatives of $I(t)$ evaluated at the pixel $\mathbf{x}_i$. This objective function is minimized by first writing out constraints over the pixels $\mathbf{x}_i \in \mathcal{W}_{\mathbf{x}}$ in matrix form as

$$\underbrace{\begin{bmatrix} I_x(\mathbf{x}_1) & I_y(\mathbf{x}_1) \\ I_x(\mathbf{x}_2) & I_y(\mathbf{x}_2) \\ \vdots & \vdots \\ I_x(\mathbf{x}_n) & I_y(\mathbf{x}_n) \end{bmatrix}}_{\mathtt{A}} \begin{pmatrix} u_{\mathbf{x}} \\ v_{\mathbf{x}} \end{pmatrix} = \underbrace{\begin{pmatrix} -I_t(\mathbf{x}_1) \\ -I_t(\mathbf{x}_2) \\ \vdots \\ -I_t(\mathbf{x}_n) \end{pmatrix}}_{\mathbf{b}}, \tag{4.6}$$

then left-multiplying both sides of (4.6) by $\mathtt{A}^\top$, giving the *normal equations*

$$\underbrace{\begin{bmatrix} \sum_{i=1}^n I_x(\mathbf{x}_i) I_x(\mathbf{x}_i) & \sum_{i=1}^n I_x(\mathbf{x}_i) I_y(\mathbf{x}_i) \\ \sum_{i=1}^n I_x(\mathbf{x}_i) I_y(\mathbf{x}_i) & \sum_{i=1}^n I_y(\mathbf{x}_i) I_y(\mathbf{x}_i) \end{bmatrix}}_{\mathtt{A}^\top\mathtt{A}} \begin{pmatrix} u_{\mathbf{x}} \\ v_{\mathbf{x}} \end{pmatrix} = \underbrace{\begin{bmatrix} -\sum_{i=1}^n I_x(\mathbf{x}_i) I_t(\mathbf{x}_i) \\ -\sum_{i=1}^n I_y(\mathbf{x}_i) I_t(\mathbf{x}_i) \end{bmatrix}}_{\mathtt{A}^\top\mathbf{b}}, \tag{4.7}$$

from which the least squares fit of $(u_{\mathbf{x}}, v_{\mathbf{x}})^\top$ to the overconstrained system of linear equations in (4.6) is obtained by

$$(u_{\mathbf{x}}, v_{\mathbf{x}})^\top = (\mathtt{A}^\top\mathtt{A})^{-1}\mathtt{A}^\top\mathbf{b}. \tag{4.8}$$

The right hand side of (4.8) can be evaluated when the 2×2 matrix $\mathtt{A}^\top\mathtt{A}$ has full rank, which occurs in patches with intensity gradients running in two dominant directions. The occurrence of rank defficiency of $\mathtt{A}^\top\mathtt{A}$ is a manifestation of the *aperture problem* (cf. Figure 4.5). A first 2D flow field is obtained by evaluating (4.8) at each pixel $I(x, y, t)$; a second iteration is then carried out, initialized with the previous iteration's flow field, and the process is repeated until a maximum number of iterations is reached or until the changes in the flow fall below a threshold. The assumption that neighboring pixels undergo similar motion in the image plane is called the assumption of *spatial coherence*, and holds for 2 DoF translational motion insofar as the underlying surface captured by the moving patch is fronto-parallel[3] and is undergoing motion that is strictly translational in 3D relative to the motion of the camera. One way to address flow discontinuities at object boundaries is to use a weighted variant of (4.8):

$$(u_{\mathbf{x}}, v_{\mathbf{x}})^\top = (\mathtt{A}^\top\mathtt{W}\mathtt{A})^{-1}\mathtt{A}^\top\mathtt{W}\mathbf{b}, \tag{4.9}$$

where $\mathtt{W} = \mathrm{diag}\big(w(\mathbf{x}, \mathbf{x}_1), \ldots, w(\mathbf{x}, \mathbf{x}_n)\big)$, and where in turn $w(\mathbf{x}, \mathbf{x}_i)$ is some weighting function. Another disadvantage of reasoning only in terms of a fixed window around $\mathbf{x}$ is that displacements that exceed the window size are impossible to capture. One common workaround for addressing larger motions is to pre-compute an image pyramid for both frames, respectively, and compute the optical flow in a coarse-to-fine manner with patch size fixed. Bouguet [Bou00]

[3] A surface is *fronto-parallel* if it is planar and is parallel with the image plane.

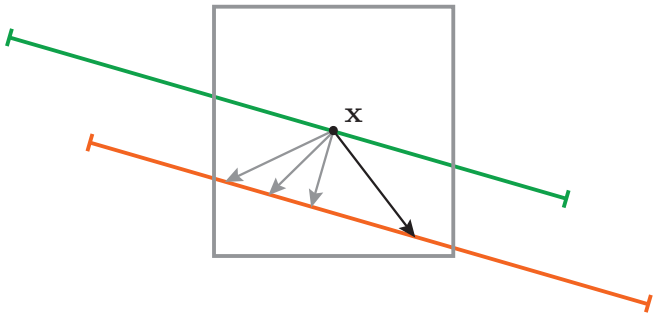


Figure 4.5: The aperture problem. The 2D motion of the line segment—green at time step t, orange at time step $t + \Delta t$—is due to the displacement indicated by the arrow in black, but judging by only the information available in the gray patch—fulfilling here the role of an 'aperture'—centered on the pixel $\mathbf{x}$ the apparent motion of the segment can likewise be explained by any displacement from $\mathbf{x}$ to the portion of the orange segment visible in the patch, as exemplified by the arrows in gray.

gives implementation details for a pyramidal variant of the Lucas-Kanade method, intended for tackling larger displacements. Other variants of Lucas-Kanade involve the use of more expressive motion models (cf. Section 2.2) than the 2D translational model implied in (4.5), such as rigid or affine (e.g., [BM04, JAHH92, BA96]).

Horn-Schunck. Most state-of-the-art techniques are variants of the Horn-Schunck method [HS81] in that they cast the problem of recovering optical flow in terms of a global energy minimization over data and smoothness terms [VRS13], in which regard the method for computing optical flow that we present in Chapter 7 is no exception. In [HS81], the global energy to be minimized is formulated as

$$\int \left(I_x \cdot u_{\mathbf{x}} + I_y \cdot v_{\mathbf{x}} + I_t\right)^2 + \lambda \cdot \left(|\nabla u_{\mathbf{x}}|^2 + |\nabla v_{\mathbf{x}}|^2\right) dx\, dy, \tag{4.10}$$

where $\lambda \geq 0$ controls the relative influence of the data and smoothness terms. An advantage of global smoothing is that it can allow for propagating flow over poorly textured areas, potentially across large displacements. Alternative data term formulations, e.g., in terms of filtered images or gradients, and alternative smoothness term formulations, e.g., in terms of a robust approximation of the L_1 norm [BWS05, PBB$^+$06] (also called *total variation*), offer additional robustness to larger displacements or flow discontinuities at object boundaries. A recent comparative evaluation of data term formulations is provided in [VRS13]. Optimization in global optical flow techniques is often a variation of gradient descent, applied in a coarse-to-fine (pyramidal) manner in order to tackle larger displacements. A notable discrete-continuous alternative is the FusionFlow method [LRR08], which proceeds by fusing proposal flow results computed independently by (continuous) optical flow algorithms, e.g., Lucas-Kanade or Horn-Schunck, casting the fusion as a discrete labeling problem over the available proposals.

4.3 Stereo Matching

Under known relative camera geometry, recovering a correspondence pair $(\mathbf{x}, \mathbf{x}')$ across a pair of images—acquired at different viewpoints of a static scene—allows for *triangulating* the point $\mathbf{X} \in \mathbb{R}^3$ that gives rise to $\mathbf{x}$ and $\mathbf{x}'$ (cf. Section 2.5). Such an image pair is termed a *stereo pair*. As in optical flow, the problem of stereo matching reduces to searching for dense correspondences between a pair of images; the key difference in stereo matching with respect to algorithms tackling optical flow is that knowledge of relative camera geometry allows for reducing the search space for a single pixel's correspondence from one encompassing a search across the entirety of image space to one restricted to a search along a single corresponding epipolar line (cf. Figure 2.8). This constraint *renders stereo matching a much easier task in general than the problem of recovering optical flow*: on the example presented in Section 1.1 of an pair of images both at a spatial resolution in height by width of 320×480, exhaustive search for discretized, pixel-accurate optical flow using a 2 DoF translational motion model for unary terms only called for evaluating a total of 153,600 candidate configurations individually for each of the 320×480 pixels, while at most only 480 would need to be evaluated individually per pixel for stereo. Most stereo matching algorithms expect input stereo pairs to have undergone prior projectively warping called *rectification*—giving rise to a rectified stereo pair—such that respective correspondences more conveniently lie on the same horizontal scanline (cf. Figure 2.9). Given a pixel in the rectified left image, the search for its correspondence in the rectified right image then reduces to finding the horizontal *disparity* relating the two. Note that in the refinement step in Chapter 7, we alternate in perturbing 9 DoF plane-induced homographies (cf. Section 2.6) between fixing depth and normal and allowing the 3D rigid body motion to undergo change, and fixing the rigid motion and allowing change to depth and normal; *the latter variant amounts to a form of unrectified stereo matching.*

The influential survey paper of Scharstein and Szeliski [SS02] presents a taxonomy of stereo algorithms in terms of (i) matching cost computation and aggregation (energy), (ii) disparity computation (optimization), (iii) and disparity refinement (post-processing). For *local* algorithms, computing a dense disparity map reduces to minimizing, independently for each pixel $\mathbf{x}$, a cost function—called a *data* term since it is intended to measure the consistency of the candidate motion with the available data—defined over some form of neighborhood around $\mathbf{x}$ and in terms of a set of motions the pixels in the neighborhood are allowed to undergo; the overall minimization is accordingly carried out with respect to an energy function comprising such data terms. Reasoning in terms of patches implicitly makes the assumption that all pixels in the patch undergo the same motion, thereby encoding an implicit form of smoothness assumption. In contrast, *global* algorithms promote smoothness explicitly by minimizing a global energy function comprising both data and *smoothness* terms, and the tendency in such methods has accordingly been to compute matching cost not over the pixels in a neighborhood centered on a pixel $\mathbf{x}$, but as a function of only the single pixel $\mathbf{x}$ itself.

Local Methods. Local stereo matching algorithms typically perform what amounts to a 'winner takes all' optimization at each pixel, placing the emphasis on the design of the local cost function and on the notion of what constitutes a window. The most common pixelwise cost

functions involve computing squared or absolute differences in color, intensity, color or intensity gradient, some combination of the three, and perhaps with some form of truncation for added robustness to noise or illumination changes; these pixelwise costs can then be aggregated over the pixels of a traditional $n \times n$ patch, using shiftable windows [FRT97, BI99], windows with adaptive size [KSC01, Vek03], or with respect to a color segmentation [TMdSA08]. The challenge being addressed by these various formulations is the confounding desire for windows to be large in order to be adequately discriminative, but at the same time for them to be small in order to mitigate the problem of straddling object boundaries. Another approach to aggregating the pixelwise costs over a window associated with a pixel $\mathbf{x}$ is to weight the influence of the respective pixelwise costs as function of the corresponding pixel's similarity or distance with respect to $\mathbf{x}$, a strategy termed *adaptive support weighting* [YK05]. A comparison of local aggregation techniques is provided in [GYWG07, TMdSA08]. Overparameterizing the disparity in terms of more expressive motion models (cf. Section 2.2) provides an avenue to ameliorating the fronto-parallel assumption; an example of such an approach is PatchMatch Stereo [BRR11], which overparameterizes disparity in terms of 3 DoF slanted planes and uses PatchMatch to obtain a labeling without discretization of the continuous label space.

Global Methods. Global stereo matching algorithms often do not perform any aggregation over windows, as they instead promote smoothness by means of some explicit smoothness constraint. A sizable proportion of global methods are formulated in terms of an energy minimization framework over a discrete label space taking into account smoothness terms $E_{\mathbf{x},\mathbf{x}'} : \mathcal{L}_\mathbf{x} \times \mathcal{L}_{\mathbf{x}'} \to \mathbb{R}$ in addition to (pixelwise) data terms $E_\mathbf{x} : \mathcal{L}_\mathbf{x} \to \mathbb{R}$:

$$E(d) = \sum_{\mathbf{x}} E_\mathbf{x}(d_\mathbf{x}) + \lambda \cdot \sum_{\{\mathbf{x},\mathbf{x}'\} \in \mathcal{E}} E_{\mathbf{x},\mathbf{x}'}(d_\mathbf{x}, d_{\mathbf{x}'}), \tag{4.11}$$

where $d_\mathbf{x}$ denotes the label assigned to $\mathbf{x}$ as encoded in d and $\mathcal{L}_\mathbf{x}$ denotes a set of possible labels—usually understood to denote possible disparities—at $\mathbf{x}$, $\lambda \geq 0$ serves to control the relative influence of data and smoothness terms and where $\mathcal{E}$ denotes the set $\bigcup_\mathbf{x} \left\{ \{\mathbf{x}, \mathbf{x}'\} \mid \mathbf{x}' \in \mathcal{N}_\mathbf{x} \right\}$, with $\mathcal{N}_\mathbf{x}$ typically understood to denote the set of 4-connected neighbors of $\mathbf{x}$. Other possibilities include understanding $\mathcal{N}_\mathbf{x}$ to instead denote the set of 8-connected neighbors of $\mathbf{x}$ [BK03], or to take into account second-order smoothness terms [WTRF09]. Having defined a global energy, any of a number of optimization frameworks may be of interest as candidates for minimizing that energy, including simulated annealing [MMP87], graph cuts [BVZ01], or loopy belief propagation [SSZ03].

4.4 Scene Flow

The term 'scene flow' is due to Vedula et al. [VBR+05], and is understood to denote the translational motion in 3D between points in a scene, with optical flow simply the projection of this scene flow to image space (cf. Figure 4.6). Traditionally, the input to scene flow algorithms has been a pair of stereo pairs (i.e., four RGB images) from a calibrated rig. With the advent of inexpensive 3D cameras like the Microsoft Kinect, investigating scene flow approaches that take as input a pair of RGB-D frames has become an increasingly practical pursuit. For rigid

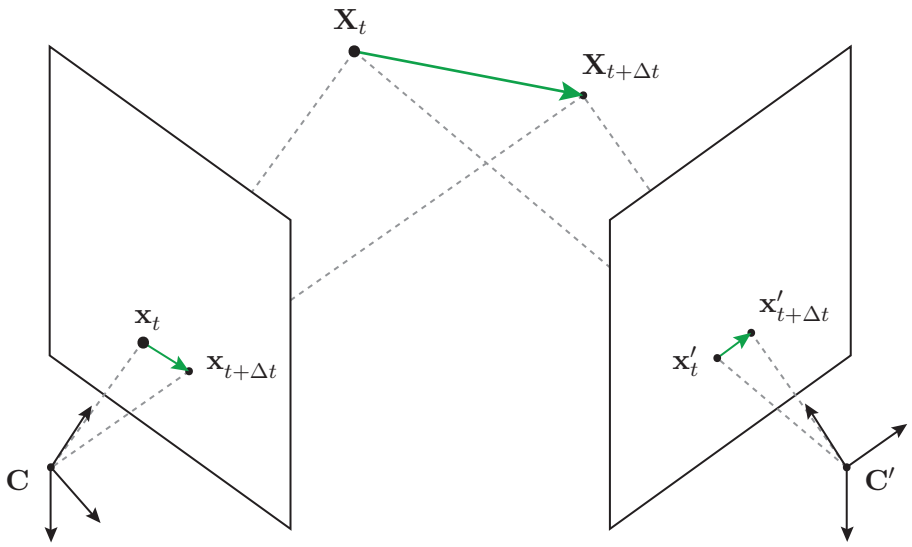


Figure 4.6: Geometry of a typical RGB scene flow setup. Given a pair of stereo pairs acquired at time steps t and $t + \Delta t$ (i.e., four RGB images), the objective is—for each pixel $\mathbf{x}_t$ of the reference frame—to jointly recover the 3D point $\mathbf{X}_t$ and the 3D translation $\mathbf{X}_{t+\Delta t} - \mathbf{X}_t$ that describes the motion of $\mathbf{X}_t$ from t to $t + \Delta t$.

scenes, sparse 3D structure and 6 DoF motion can be obtained using structure from motion (e.g., Pollefeys et al. [PKG99]).

RGB Methods. Notable RGB scene flow methods include Huguet and Devernay [HD07], Min and Sohn [MS06], Zhang and Kambhamettu [ZK01], Basha et al. [BMK13], Čech et al. [vSRH11], and Vogel et al. [VSR11]. Basha et al. use a 3D point cloud representation to directly model the desired 3D unknowns, allowing smoothness assumptions to be imposed directly on the scene flow and structure. Čech et al. compute scene flow across sequences of stereo pairs by growing correspondence seeds. Vogel et al. carry out regularization by encouraging a locally rigid 3D motion field using a rigid motion prior, avoiding systematic biases of 2D isotropic regularization. Pons et al. [PKF07] introduce a variational framework for multiple-view motion-stereo estimation that works directly in world space, evolving a 3D vector field to register the input images captured at different times. Carceroni and Kutulakos [CK02] model the scene as a set of *surfels*, with each surfel described by shape, reflectance, bump map, and affine motion. They recover surfel parameters by maximizing photoconsistency, but require knowledge of relative camera geometry and of the illumination scenario. Devernay et al. [DMG06] likewise proceed by tracking surfels. In both cases, surfels imply the local surface planarity assumption. Vogel et al. [VSR13] assign rigidly moving planes to image segments, parameterized in terms of plane-induced homographies. Wedel et al. [WRV+08] explicitly decouple the position (stereo)

and velocity estimation steps and estimate dense velocities using a variational approach while reporting frame rates of 5 FPS on standard consumer hardware.

RGB-D Methods. Letouzey et al. [LPB11] claim novelty for using RGB-D data obtained from a Kinect. They use sparse feature points to help guide the motion field estimation for wide displacements and use dense normal flow for short ones. Hadfield and Bowden [HB13] use a particle filter while assuming brightness constancy. Herbst et al. [HRF13] compute RGB-D scene flow using a variational technique, which they apply to rigid motion segmentation. Quiroga et al. [QDC13] likewise use a variational technique, combining local and global constraints to compute RGB-D scene flow.

CHAPTER 5

Depth Super Resolution

In this chapter, we tackle the problem of jointly increasing the spatial resolution and apparent measurement accuracy of an input low-resolution, noisy, and perhaps heavily quantized depth map. In stark contrast to earlier work, we make no use of ancillary data like a color image at the target resolution, multiple aligned depth maps, or a database of high-resolution depth exemplars. Instead, we proceed by identifying and merging patch correspondences within the input depth map itself, exploiting patchwise scene self-similarity across depth such as repetition of geometric primitives or object symmetry. Rather than reason in terms of patches of 2D pixels as others have before us, our key contribution is to proceed by reasoning in terms of patches of 3D points, with matched patch pairs related by a respective 6 DoF 3D rigid body motion. We begin in Section 5.1 by introducing our methodology and by placing our algorithm in the context of earlier work. Next, we detail out algorithm in Section 5.2, followed by an evaluation in Section 5.3 showing our results to be highly competitive with those of alternative techniques leveraging even a color image at the target resolution or a database of high-resolution depth exemplars. In Section 5.4 we attempt to emphasize the key elements of the algorithm in what concerns performance, finishing off with concluding remarks in Section 5.5.

5.1 Introduction

With the advent of inexpensive 3D cameras like the Microsoft Kinect, depth measurements are becoming increasingly available for low-cost applications. Acquisitions made by such consumer 3D cameras, however, remain afflicted by less than ideal attributes. Random errors are a common problem. Low spatial resolution is an issue particularly with time of flight (ToF) cameras, e.g., 200×200 for the PMD CamCube 2.0 or 176×144 for the SwissRanger SR3000. In depth maps recovered using stereo techniques, depth resolution decreases as a function of increasing depth from the camera. Common avenues to jointly increasing the spatial resolution and apparent measurement accuracy of a depth map—a problem referred to as depth super resolution (SR)—involve leveraging ancillary data such as a color or intensity image at the target resolution,

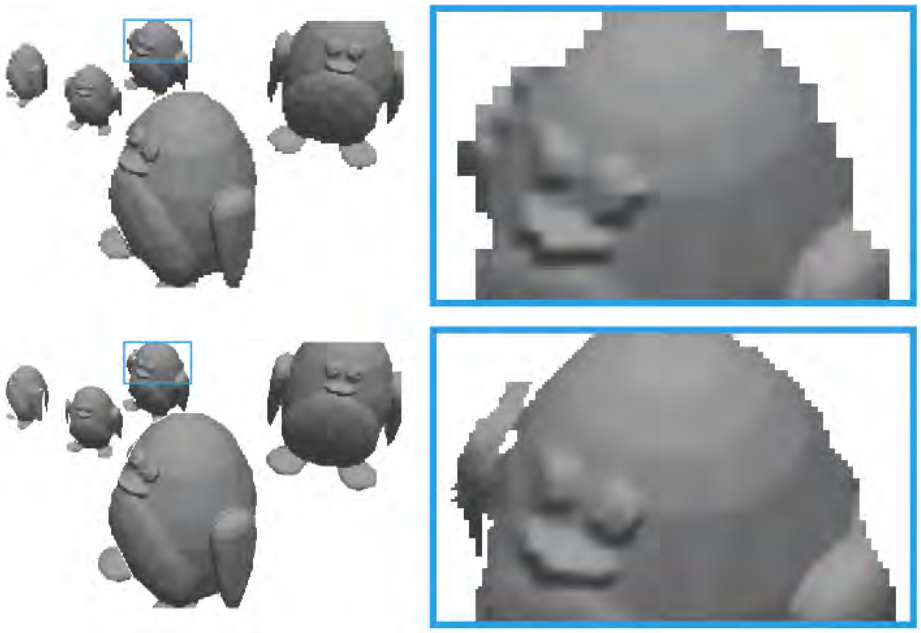
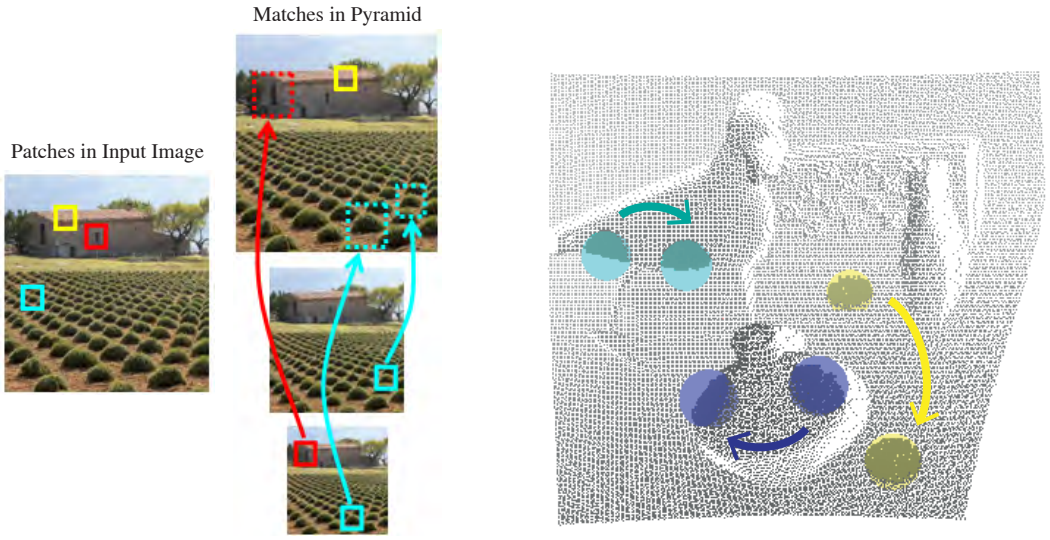


Figure 5.1: Shaded mesh of nearest neighbor upscaling (top) of a noiseless synthetic input depth map and the super resolved output of our algorithm (bottom), both by a factor of 3. In our approach, fine details such as the penguin's eyes, beak, and the subtle polygons across its body are mapped from corresponding patches at lesser depth, and boundaries appear less jagged than in the upscaled counterpart.

multiple aligned depth maps, or a database of high-resolution depth exemplars (patches). Such ancillary data, however, is often unavailable or difficult to obtain.

In this work, we consider the question of how far one can push depth SR using no ancillary data, proceeding instead by identifying and merging patch correspondences from within the input depth map itself. Our observation is that—even in the absence of object repetition of the sort exemplified in the idealized target scenario in Figure 5.1—real-world scenes tend to exhibit patchwise 'self-similarity' such as repetition of geometric primitives (e.g., planar surfaces, edges) or object symmetry (consider a face, a vase). Man-made scenes or objects are often 'self-similar' by design; consider, for instance, the keys of a keyboard. It is primarily this observation that we exploit in this work, coupled with the fact that under perspective projection, an object patch at lesser depth with respect to the camera is acquired with a higher spatial resolution than a corresponding patch situated at greater depth. Our **main contribution** is to proceed not by reasoning in terms of patches of 2D pixels, but rather in terms of patches of 3D points. Additionally, we introduce a new 3D variant of PatchMatch to obtain a dense correspondence field in reasonable time and a simple, yet effective patch upscaling and merging technique to generate the output SR depth map from the correspondences.

The notion of 'single-image' SR has already successfully been applied in the context of color and intensity images in the work of Glasner et al. [GBI09]. Their guiding observation is that within the same image there is often a large across-scale redundancy at the 2D pixel patch level;



(a) Patches of 2D pixels in an image (pixels inside solid squares), matched within and across scale in the 'single image' image SR approach of Glasner et al. Generating the SR output image involves merging high resolution versions of the matches (dashed squares).

(b) Our patches of 3D points encoded in a depth map (points inside spheres), matched within and across depth. Generating the SR output depth map in our 'single image' depth SR approach involves mapping closer patches backward and merging the resulting points.

Figure 5.2: 'Single image' super resolution. (a) The 'single image' SR approach for color and intensity images of Glasner et al. [GBI09] seeks correspondences between small, conventional $n \times n$ pixel patches within and across scale. (b) In our depth SR approach, we seek correspondences between patches of 3D points within and across depth. Advantages of reasoning in terms of such patches of 3D points for depth SR include an allowance for patches and displacements to be larger, robustness to noisy points situated outside the sphere, and that major object boundaries are handled in a natural way. (The image in (a) is adapted from [GBI09].)

for instance, an image of a leafy forest is likely to contain a large number of *small* patches with various configurations of greens and browns that happen to recur across scales of the image. Their strategy is to search for corresponding 5×5 pixel patches across a discrete cascade of downscaled copies of the input image and to exploit sub-pixel shifts between correspondences. Taking their framework to depth SR, three fundamental problems of matching 3D points using $n \times n$ pixel patches treating depth values as intensity present themselves, even at potentially valuable ground truth correspondences in idealized scenarios such as Figure 5.1: patch pairs (i) are situated at different depths or (ii) are subject to projective distortions owing to perspective projection, or (iii) they straddle object boundaries. The problem of projective distortions calls for a small patch size (such as the 5×5 patches of Glasner et al.), but this renders matching particularly sensitive to noise in depth (cf. Figure 1.3), which tends to be more pronounced than noise in color or intensity images. We overcome these problems by reasoning in terms of patches of 3D points, which we define as the respective inliers—from among the 3D points

encoded in the input depth map—within a fixed radius r of a center point and which we match with respect to 3D point similarity over 6 DoF rigid body motions in 3D. Additionally, the availability of depth information suggests reasoning in terms of across-*depth* redundancy rather than redundancy across scale; accordingly, for each such patch of 3D points, what we seek is a matching patch at equal or lesser depth with respect to the camera, as illustrated in Figure 5.2.

5.1.1 Related Work

A number of surveys of *image* SR techniques are available elsewhere, e.g., in van Ouwerkerk [vO06] or Tian and Ma [TM11]. Glasner et al. [GBI09], Yang et al. [YWHM10], and Freeman and Liu [FL11] are image SR techniques against which we compared our algorithm in Section 5.3, by treating input depth maps as if they were ordinary intensity images. Freeman and Liu et al. and Yang et al. both rely on an external database of patches. Broadly speaking, previous work on *depth* SR can be categorized into methods that (i) use a guiding color or intensity image at the target resolution, (ii) merge information contained in multiple aligned depth maps, or (iii) call on an external database of high-resolution depth exemplars, each of which accordingly use ancillary data to compute a super resolved output depth map. We devote the remainder of this section to a discussion of representative or seminal techniques from the depth SR literature.

Image at Target Resolution. The most common depth SR strategy involves using an ancillary color or intensity image at the target resolution to guide the reconstruction of the SR depth map. The underlying assumption is that changes in depth are colocated with edges in the guiding image. Yang et al. [YYDN07] apply joint bilateral upscaling on a cost volume constructed from the low resolution input depth map, followed by Kopf et al. [KCLU07] in a more general framework. Diebel and Thrun [DT05] propose an MRF-based approach with a pairwise smoothness term whose contribution is weighted according to the edges in the high-resolution color image. Park et al. [PKT$^+$11] take this idea further and use a non-local, highly-connected smoothness term that better preserves thin structures in the SR output.

Database of Depth Exemplars. Most closely akin to ours is the work of Mac Aodha et al. [MCNB12]. They propose to assemble the SR depth map from a collection of depth patches. Our approach likewise carries out depth SR by example, but with significant differences. One major difference is that we use patches only from within the input depth map itself, whereas Mac Aodha et al. use an external database of 5.2 *million* high-resolution synthetic, noise-free patches. Another difference is that they carry out their matching in image space over 3×3 patches pixel patches, while ours can have arbitrary size depending on the scale, density, and relative depth of point features one aims to capture. Accordingly, their approach is subject to the problems discussed in Section 5.1 that our reasoning in terms of 3D point patches overcomes. Note that enlarging the patches in their database would lead to an explosion of its size.

Multiple Depth Maps. The Lidarboost approach of Schuon et al. [STDT09] combines several depth maps acquired from slightly different viewpoints. The KinectFusion approach of Izadi et

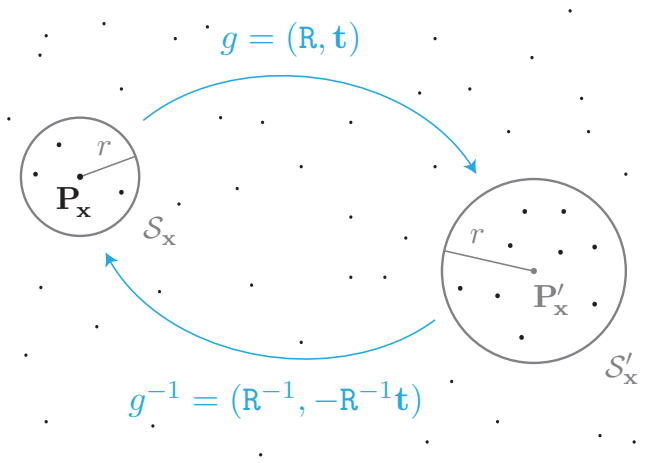


Figure 5.3: A candidate rigid body motion g relating the 3D point patches $\mathcal{S}_\mathbf{x}, \mathcal{S}'_\mathbf{x} \subset \mathbb{R}^3$. The center point $\mathbf{P}_\mathbf{x} \in \mathbb{R}^3$ of the further patch $\mathcal{S}_\mathbf{x}$ is the point encoded at pixel $\mathbf{x}$ in the input depth map. Note that the center point $\mathbf{P}'_\mathbf{x} = g(\mathbf{P}_\mathbf{x})$ of the closer patch $\mathcal{S}'_\mathbf{x}$ is by design not required to be one of the 3D points of the input depth map, hence $\mathbf{P}'_\mathbf{x} \notin \mathcal{S}'_\mathbf{x}$ in general.

al. [IKH+11] produces outstanding results by fusing a sequence of depth maps generated by a tracked Kinect camera into a single 3D representation in real-time.

5.2 Algorithm

Owing to the perspective projection that underlies image formation, object patches situated at a lesser depth with respect to the camera are imaged with a higher spatial resolution (i.e., a greater point density) than corresponding object patches at greater depth. Our depth SR algorithm consists of two steps: (i) find, for each patch in the input depth map, a corresponding patch at lesser or equal depth with respect to the camera, and (ii) use the dense correspondence field to generate the SR output. We begin, in Section 5.2.1, by presenting our notion of '3D point patch' and the matching cost we propose to minimize. Next, we detail the first step of our algorithm in Section 5.2.2, and the second in Section 5.2.3.

5.2.1 3D Point Patches

Let $\mathbf{x} = (x, y)^\top$ be a pixel of the input depth map. The goal of the dense correspondence search algorithm in Section 5.2.2 is to find an optimal rigid body motion $g_\mathbf{x} = (\mathtt{R}_\mathbf{x}, \mathbf{t}_\mathbf{x})$ for each pixel $\mathbf{x}$, mapping the patch corresponding to $\mathbf{x}$ to a valid matching patch at lesser or equal depth with respect to the camera. We shall understand the *3D point patch* corresponding to the pixel $\mathbf{x}$—the *further*[1] patch, for brevity—to be the set $\mathcal{S}_\mathbf{x} \subset \mathbb{R}^3$ of 3D points encoded in the depth map within

[1]We acknowledge that this is something of an abuse of terminology since, strictly speaking, two points can be situated at equal *depth* with respect to the camera center but be at different *distances* from it. Notwithstanding, it is in this sense that we shall mean 'closer' and 'further' in this work.

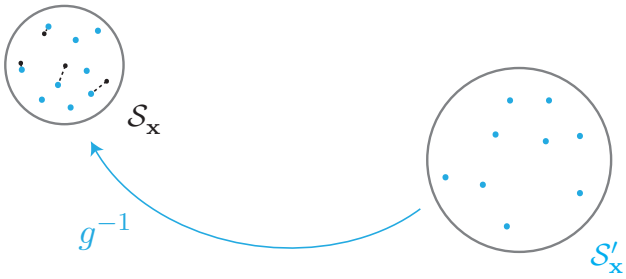


Figure 5.4: Matching cost is computed as a convex combination of 'forward' cost and 'backward' cost. Shown is an illustration of the 'backward' cost, which, given a candidate rigid body motion g, is a function of the 'backward'-transformed points $g^{-1}(\mathcal{S}'_\mathbf{x})$ of the closer patch $\mathcal{S}'_\mathbf{x}$; the 'forward' cost is computed analogously as a function of the 'forward'-transformed points $g(\mathcal{S}_\mathbf{x})$ of the further patch $\mathcal{S}_\mathbf{x}$.

a radius r of the point $\mathbf{P}_\mathbf{x} = Z_\mathbf{x} \cdot \mathtt{K}^{-1}(\mathbf{x}^\top, 1)^\top \in \mathbb{R}^3$ of $\mathbf{x}$, where $Z_\mathbf{x}$ is the depth encoded at the pixel $\mathbf{x}$ in the input depth map and $\mathtt{K}$ is the 3×3 camera calibration matrix (cf. Section 2.4). We carry out radius queries using a kd-tree [ML14]. The 3D points of the corresponding *closer* patch $\mathcal{S}'_\mathbf{x}$ are those within the same radius r of the point $\mathbf{P}'_\mathbf{x} = g_\mathbf{x}(\mathbf{P}_\mathbf{x})$. An illustration of these notions is provided in Figure 5.3.

Matching Cost. A common strategy in the literature on iterative closest point (ICP) algorithms for evaluating the similarity of two point sets is to compute the sum of squared differences (SSD) over each point in one point set with respect to its nearest neighbor (NN) point in the other (cf. Rusinkiewicz and Levoy [RL01]). We proceed in a similar manner to quantify the similarity of two 3D point patches, but normalize the result and allow for computing SSD in both directions in order to potentially obtain a stronger similarity measure, noting that we might be comparing point sets with significantly different point densities owing to relative differences in patch depth. Let $\mathrm{NN}_\mathcal{S}(\mathbf{P})$ denote the function that returns the nearest neighbor to the point $\mathbf{P}$ in the set $\mathcal{S}$. The function $E^\mathrm{b}_\mathbf{x}(g)$ evaluates normalized SSD over the points of the further patch $\mathcal{S}_\mathbf{x}$ subject to each point's respective nearest neighbor among the 'backward'-transformed points $g^{-1}(\mathcal{S}'_\mathbf{x})$ of the closer patch $\mathcal{S}'_\mathbf{x}$:

$$E^\mathrm{b}_\mathbf{x}(g) = \sum_{\mathbf{P} \in \mathcal{S}_\mathbf{x}} \left\| \mathbf{P} - \mathrm{NN}_{g^{-1}(\mathcal{S}'_\mathbf{x})}(\mathbf{P}) \right\|_2^2 / |\mathcal{S}_\mathbf{x}|. \tag{5.1}$$

Analogously, the function $E^\mathrm{f}_\mathbf{x}(g)$ evaluates normalized SSD over the points of the closer patch $\mathcal{S}'_\mathbf{x}$ subject to their respective nearest neighbors among the 'forward'-transformed points $g(\mathcal{S}_\mathbf{x})$ of the further patch $\mathcal{S}_\mathbf{x}$:

$$E^\mathrm{f}_\mathbf{x}(g) = \sum_{\mathbf{P}' \in \mathcal{S}'_\mathbf{x}} \left\| \mathbf{P}' - \mathrm{NN}_{g(\mathcal{S}_\mathbf{x})}(\mathbf{P}') \right\|_2^2 / |\mathcal{S}'_\mathbf{x}|. \tag{5.2}$$

For g to be deemed *valid* at $\mathbf{x}$, we require that the depth of the sphere center point of the matched patch be less than or equal to the depth assigned at the pixel $\mathbf{x}$ in the input depth map. Addition-

ally, we require that their relative distance be at least r in order to avoid minimizing cost trivially by matching to oneself, and that $|\mathcal{S}'_{\mathbf{x}}| \geq |\mathcal{S}_{\mathbf{x}}| \geq 3$ in order to benefit from greater point density or from sub-pixel point shifts at equal density, and for reasons discussed below. Given a pixel $\mathbf{x}$ and a 3D rigid body motion g, we compute the matching cost $E_{\mathbf{x}}(g)$ as a convex combination of the 'backward' cost $E^{\mathrm{b}}_{\mathbf{x}}(g)$ (5.1) and the 'forward' cost $E^{\mathrm{f}}_{\mathbf{x}}(g)$ (5.2) according to

$$E_{\mathbf{x}}(g) = \begin{cases} \alpha \cdot E^{\mathrm{b}}_{\mathbf{x}}(g) + \alpha' \cdot E^{\mathrm{f}}_{\mathbf{x}}(g) & \text{if valid} \\ \infty & \text{otherwise} \end{cases}, \tag{5.3}$$

where $\alpha \in [0, 1]$ and $\alpha' = 1 - \alpha$. It is for α to control the relative influence of the 'backward' cost and 'forward' cost that the two constituent costs are normalized with respect to the size of $\mathcal{S}_{\mathbf{x}}$ and $\mathcal{S}'_{\mathbf{x}}$, respectively. The absence of this normalization would introduce a bias towards matches where $|\mathcal{S}_{\mathbf{x}}| \approx |\mathcal{S}'_{\mathbf{x}}|$, which has a tendency to occur when $\mathbf{P}_{\mathbf{x}}$ and $\mathbf{P}'_{\mathbf{x}}$ are situated at similar depth.

5.2.2 Dense 6 DoF Matching via PatchMatch

We introduce a new 3D variant of the PatchMatch algorithm of Barnes et al. [BSFG09,BSGF10] in the aim of assigning to each pixel $\mathbf{x}$ of the input depth map a 6 DoF 3D rigid body motion, mapping $\mathcal{S}_{\mathbf{x}}$ to a valid matching patch $\mathcal{S}'_{\mathbf{x}}$ at equal or lesser depth with respect to the camera. A common thread between variants of PatchMatch—in which ours is no exception—is a random (or semi-random) initialization step followed by i iterations of propagation and refinement. We explain each step in greater detail in the remainder of this section. An example of a projected displacement field obtained using our 3D variant of PatchMatch is visualized in Figure 5.5.

Semi-Random Initialization. In contrast to PatchMatch variants that carry out initialization using altogether random states, we adopt a semi-random initialization strategy. In our experiments, we found this led to faster convergence when dealing with our high-dimensional state space. Specifically, for each pixel $\mathbf{x}$ we randomly select another pixel $\mathbf{x}'$ of the input depth map such that the depth assigned to $\mathbf{x}'$ is less than or equal to that of $\mathbf{P}_{\mathbf{x}}$, giving us a translation vector (3 DoF). We then compute the rotation minimizing arc length between the patch normal vectors at $\mathbf{P}_{\mathbf{x}}$ and $\mathbf{P}_{\mathbf{x}'}$, respectively, (2 DoF) and choose a random angular perturbation around the normal vector of $\mathbf{P}_{\mathbf{x}'}$ (1 DoF). We pack these elements into a rigid body motion. An illustration of this process is provided in Figure 5.6. A normal vector for each point $\mathbf{P}_{\mathbf{x}}$ is precomputed via RANSAC [FB81] plane fitting over the 3D points in the 3D point patch $\mathcal{S}_{\mathbf{x}}$ (and is the reason why we require that $|\mathcal{S}_{\mathbf{x}}| \geq 3$ in Section 5.2.1), which is made to point towards the camera center.

Spatial Propagation. In keeping with the traversal schedule used in classical PatchMatch (cf. Barnes et al. [BSFG09, BSGF10]), we traverse the pixels $\mathbf{x}$ of our input depth map in scanline order—upper left to lower right for even iterations, lower right to upper left for odd—and adopt the rigid body motion assigned to a neighboring pixel if doing so yields an equal or lower cost. Note that as a consequence, we propagate over pixels for which $|\mathcal{S}_{\mathbf{x}}| < 3$, which we treat as so-called flying pixels, since such pixels are always assigned infinite cost by $E_{\mathbf{x}}(g)$ in (5.3). The

(a) Input (shown as disparity map). (b) Projected 3D displacements.

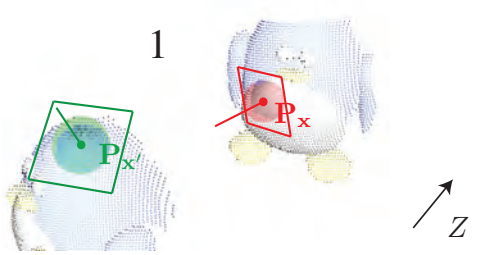
Figure 5.5: Visualization of projected 3D displacements of the output of our dense matching. (a) A filled variant of the disparity map of the Middlebury cones data set as input to our depth SR algorithm (after conversion to a depth map). We use this filled variant of the disparity map in the evaluation because the algorithms against which we compared ourselves in Section 5.3.1 were run on it; our algorithm can easily handle background values by simply ignoring them. (b) A visualization of projected 3D displacements of the output of our dense correspondence search using conventional 2D flow coloring (cf. Figure 4.2). Note that the flow field is spatially coherent.

geometric rationale underlying spatial propagation in our 6 DoF setting can be understood by observing that if two objects are related by a rigid body motion, then any corresponding pair of 3D point patches is related (modulo noise or sampling) by precisely the same motion.

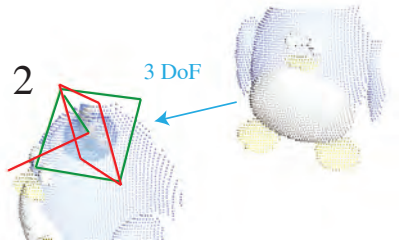
Refinement. Immediately following propagation at a given pixel $\mathbf{x}$, we independently carry out k iterations of additional initialization and of perturbation of the translational and rotational components of the assigned motion $g_{\mathbf{x}}$ (cf. Section 2.3), adopting the initialization or perturbation if doing so yields an equal or lower cost. Translational perturbation (3 DoF) consists of checking whether hopping from $\mathbf{P}'_{\mathbf{x}}$ to one of its k-NN points $\mathbf{P}_{\mathbf{x}'}$—which we obtain by again making use of a kd-tree—yields an equal or lower cost. Rotational perturbation, which we carry out in a range that decreases with every iteration k, consists of random rotation around the normal at $\mathbf{P}_{\mathbf{x}}$ (1 DoF) and of random perturbation of the remaining two degrees of freedom of the rotation. We carry out and evaluate all three types of perturbations independently.

5.2.3 Patch Upscaling and Merging

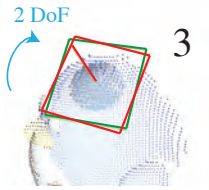
Having assigned a motion $g_{\mathbf{x}} \in SE(3)$ to each pixel $\mathbf{x}$ of the input depth map, we generate an SR depth map by merging interpolated depth values of the 'backward'-transformed points $g_{\mathbf{x}}^{-1}(\mathcal{S}'_{\mathbf{x}})$ of each valid matched patch. We begin, for each $\mathbf{x}$, by (i) determining—with the help of contour polygonalization—the spatial extent of $\mathcal{S}_{\mathbf{x}}$ at the target resolution, giving an 'overlay mask'



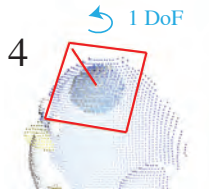
(a) Randomly pick a point $\mathbf{P}_{\mathbf{x}'}$ at equal or lesser depth with respect to that of $\mathbf{P}_{\mathbf{x}}$.



(b) Translate accordingly.

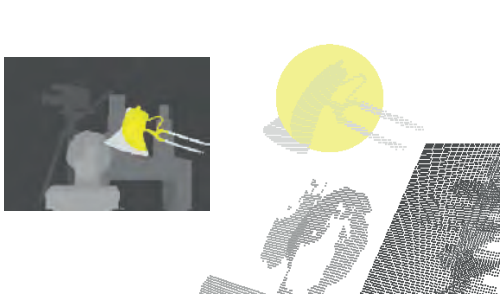


(c) Rotate the source normal vector onto the destination normal vector.

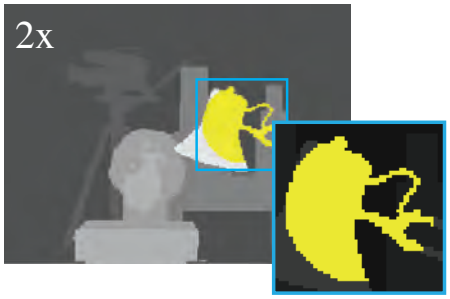


(d) Randomly rotate around the destination normal.

Figure 5.6: Semi-random initialization of a 6 DoF rigid body motion g at a pixel $\mathbf{x}$, with the objective of restricting initialization to plausible states (most importantly meaning never initializing to empty space). (a) Randomly pick a point $\mathbf{P}_{\mathbf{x}'}$ at equal or lesser depth with respect to that of the point $\mathbf{P}_{\mathbf{x}}$ encoded at $\mathbf{x}$. (b) Obtain the 3 DoF translation from $\mathbf{P}_{\mathbf{x}'} - \mathbf{P}_{\mathbf{x}}$. (c) Obtain 2 DoF of the 3 DoF rotation from the rotation between the local normal vectors. (d) Obtain the remaining 1 DoF of the 3 DoF rotation by random rotation around the normal.



(a) Mask at input resolution.



(b) 2x NN mask.

(c) 2x overlay mask.

Figure 5.7: Overlay masks. (a) A patch of points $\mathcal{S}_{\mathbf{x}}$ in the input point cloud and its corresponding pixel mask in the raster of the input depth map (depicted in yellow). (b) Nearest neighbor upscaling of the depth map and mask by a factor of 2 (giving the target resolution). (c) Corresponding polygon approximation (polygonalization) of the nearest neighbor upscaled mask, which we term the 'overlay mask' corresponding to $\mathbf{x}$. In the merging step, it is only the pixels $\hat{\mathbf{x}}$ at the target resolution of the overlay mask of $\mathbf{x}$ that the 'backward'-transformed points $g_{\mathbf{x}}^{-1}(\mathcal{S}'_{\mathbf{x}})$ of the matched patch are allowed to influence.

over which we then (ii) generate an 'overlay patch' by interpolating depth values from the points $g_{\mathbf{x}}^{-1}(\mathcal{S}'_{\mathbf{x}})$. Next, we (iii) populate the SR depth map by merging the interpolated depth values of overlapping overlay patches, with the influence of each valid overlay patch weighted as a function of patch similarity. Finally, we (iv) clean the SR depth map in a post-processing step, removing small holes that might have arisen at object boundaries as a consequence of polygonalization. The pseudocode for steps (i-iii) is provided in Algorithm 5.1; the remainder of this section is dedicated to detailing each of the steps (i-iv).

Overlay Masks. The 2D pixels $\mathbf{x}$ of the input depth map to which the 3D points of $\mathcal{S}_{\mathbf{x}}$ project define the spatial extent of $\mathcal{S}_{\mathbf{x}}$ at the input resolution (cf. Figure 5.7). It is only these pixels—at the input resolution—that we wish the 'backward'-transformed points $g_{\mathbf{x}}^{-1}(\mathcal{S}'_{\mathbf{x}})$ of the matched closer patch to influence in generating an output depth map, since it is over these pixels that we computed the matching cost. Upscaling the mask by the SR factor using NN interpolation gives a mask at the target resolution, but introduces disturbing jagged edges. Accordingly, we carry out

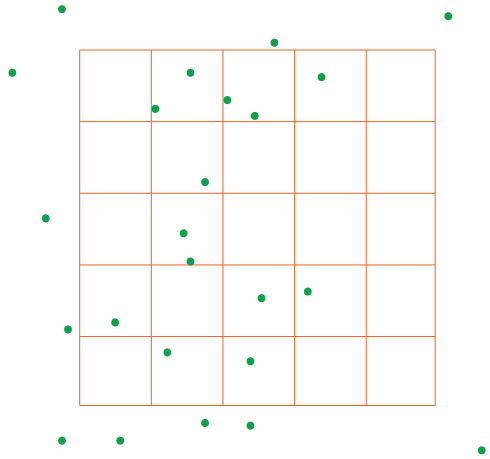
Algorithm 5.1 Patch Upscaling and Merging

initialize $\Omega_{\hat{\mathbf{x}}}$ and $Z_{\hat{\mathbf{x}}}$ to 0 for all pixels $\hat{\mathbf{x}}$ at the *target* resolution
for all pixels $\mathbf{x}$ at the *input* resolution (i.e., in the input depth map) **do**
  if $|\mathcal{S}_\mathbf{x}| \geq 3$ **then**
    $\omega_\mathbf{x} \leftarrow \exp\big(-\gamma \cdot E^{\mathrm{b}}_\mathbf{x}(g_\mathbf{x})\big)$
    if $E^{\mathrm{b}}_\mathbf{x}(g_\mathbf{x}) > \beta$ **then**
      $g_\mathbf{x} \leftarrow$ identity motion $g_I = (\mathtt{I}, \mathbf{0})$
    end if
    for all pixels $\hat{\mathbf{x}}$ at the *target* resolution in $\mathrm{overlayMask}^\mathbf{x}$ **do**
      $\mathrm{overlayPatch}^\mathbf{x}_{\hat{\mathbf{x}}} \leftarrow$ interpolated Z-value from $g_\mathbf{x}^{-1}(\mathcal{S}'_\mathbf{x})$ at $\hat{\mathbf{x}}$
      $Z_{\hat{\mathbf{x}}} \leftarrow Z_{\hat{\mathbf{x}}} + \omega_\mathbf{x} \cdot \mathrm{overlayPatch}^\mathbf{x}_{\hat{\mathbf{x}}}$
      $\Omega_{\hat{\mathbf{x}}} \leftarrow \Omega_{\hat{\mathbf{x}}} + \omega_\mathbf{x}$
    end for
  end if
end for
for all pixels $\hat{\mathbf{x}}$ at the *target* resolution **do**
  if $\Omega_{\hat{\mathbf{x}}} > 0$ **then**
    $Z_{\hat{\mathbf{x}}} \leftarrow Z_{\hat{\mathbf{x}}} / \Omega_{\hat{\mathbf{x}}}$
  end if
end for

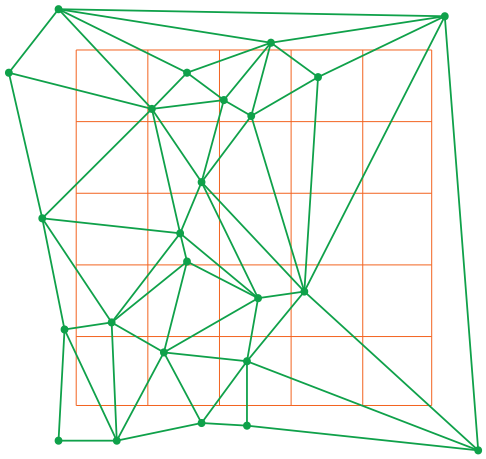
a polygon approximation [PD73] of this NN upscaled mask, constrained such that approximated contours be at a distance of at most the SR factor—which corresponds to a single pixel at the input resolution—from the NN upscaled contours. We ignore recovered polygonalized contours whose area is less than or equal to the square of the SR factor, thereby removing flying pixels. This polygonalized mask—to which we refer as the *overlay mask* of $\mathbf{x}$—consists of all pixels $\hat{\mathbf{x}}$ at the target resolution that fall into one of the remaining polygonalized contours but fall into no contour that is nested inside another, in order to handle holes like in the lamp in Figure 5.7.

Overlay Patches. We interpolate, for the pixels $\hat{\mathbf{x}}$ at the target resolution of the overlay mask corresponding to the input pixel $\mathbf{x}$, depth values from the 'backward'-transformed points $g_\mathbf{x}^{-1}(\mathcal{S}'_\mathbf{x})$ of the closer patch $\mathcal{S}'_\mathbf{x}$. This gives what we term the *overlay patch* of $\mathbf{x}$, and it is in this sense that we perform patch 'upscaling'. Since the points $g_\mathbf{x}^{-1}(\mathcal{S}'_\mathbf{x})$ are not guaranteed to project to a regular grid the way $\mathcal{S}'_\mathbf{x}$ do, we compute depth values for the pixels of the overlay mask by interpolating over the depth values of the points $g_\mathbf{x}^{-1}(\mathcal{S}'_\mathbf{x})$ using barycentric coordinates with respect to a Delaunay triangulation of the projections of those transformed points to image space. This amounts to a form of interpolation over an irregular grid [SBM95]. Our interpolation scheme is illustrated in Figure 5.8.

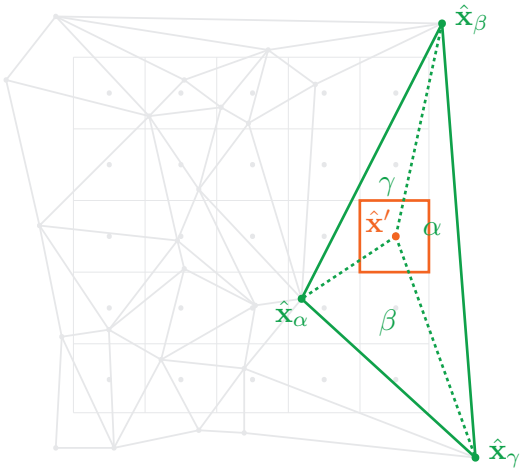
Merging of Overlay Patches. The SR depth map is computed by working out, for each pixel $\hat{\mathbf{x}}$ at the target resolution, a weighted average of the corresponding interpolated depth values from the overlapping overlay patches. The weight $\omega_\mathbf{x}$ of the interpolated depth value at each $\hat{\mathbf{x}}$ in the



(a) Projection of $g_{\mathbf{x}}^{-1}(\mathcal{S}'_{\mathbf{x}})$ to image space.

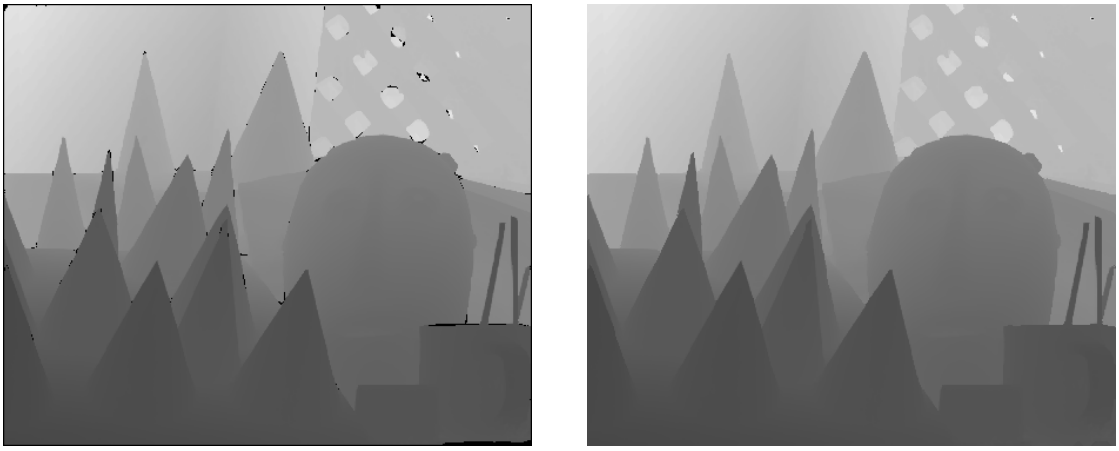


(b) Delaunay triangulation.



(c) Barycentric coordinates (α, β, γ) of the centered pixel $\hat{\mathbf{x}}' = \hat{\mathbf{x}} + (0.5, 0.5)^\top$ with respect to the triangle $\hat{\mathbf{x}}_\alpha \hat{\mathbf{x}}_\beta \hat{\mathbf{x}}_\gamma$ enclosing $\hat{\mathbf{x}}'$.

Figure 5.8: Interpolating a depth $Z_{\hat{\mathbf{x}}}$ for a pixel $\hat{\mathbf{x}}$ at the target resolution for the overlay patch corresponding to the input pixel $\mathbf{x}$ given the 'backward'-transformed points $g_{\mathbf{x}}^{-1}(\mathcal{S}'_{\mathbf{x}})$ of the closer patch $\mathcal{S}'_{\mathbf{x}}$. Having computed the Delaunay triangulation of the projections at the target resolution of the points $g_{\mathbf{x}}^{-1}(\mathcal{S}'_{\mathbf{x}})$, the interpolated depth $Z_{\hat{\mathbf{x}}} = \alpha \cdot Z_{\hat{\mathbf{x}}_\alpha} + \beta \cdot Z_{\hat{\mathbf{x}}_\beta} + \gamma \cdot Z_{\hat{\mathbf{x}}_\gamma}$ is computed as the weighted sum of the depths $Z_{\hat{\mathbf{x}}_\alpha}, Z_{\hat{\mathbf{x}}_\beta}, Z_{\hat{\mathbf{x}}_\gamma}$ of the points in $g_{\mathbf{x}}^{-1}(\mathcal{S}'_{\mathbf{x}})$ projecting to the pixels $\hat{\mathbf{x}}_\alpha, \hat{\mathbf{x}}_\beta, \hat{\mathbf{x}}_\gamma$ at the target resolution that enclose the centered pixel $\hat{\mathbf{x}}' = \hat{\mathbf{x}} + (0.5, 0.5)^\top$ as a triangle in the Delaunay triangulation. The weights $(\alpha, \beta, \gamma), \alpha + \beta + \gamma = 1$, are the Barycentric coordinates of $\hat{\mathbf{x}}'$ with respect to the enclosing triangle $\hat{\mathbf{x}}_\alpha \hat{\mathbf{x}}_\beta \hat{\mathbf{x}}_\gamma$. Note that projecting to the target resolution effectively calls for scaling the focal length and principal point in the camera calibration matrix `K` by the SR factor.



(a) Before post-processing. (b) After post-processing.

Figure 5.9: Post-processing for 2x SR on the cones data set (shown as depth map). (a) The output of the merging stage can contain holes (indicated in black), owing to the fact that there is no guarantee that each pixel at the target resolution will be covered by some polygonalized overlay mask. (b) Post-processing by iterative morphological dilation.

overlay patch assigned to the pixel $\mathbf{x}$ is given by

$$\omega_{\mathbf{x}} = \exp\Big(-\gamma \cdot E^{\mathrm{b}}_{\mathbf{x}}(g_{\mathbf{x}}) \Big), \tag{5.4}$$

where $\gamma \in \mathbb{R}^+$ controls the falloff to 0. If $E^{\mathrm{b}}_{\mathbf{x}}(g_{\mathbf{x}}) > \beta$, $\beta \in \mathbb{R}^+$, we instead use the overlay patch at $\mathbf{x}$ given by the identity motion $g_I = (\mathtt{I}, \mathbf{0})$—which amounts to simply using $\mathcal{S}_{\mathbf{x}}$ to generate the overlay patch at $\mathbf{x}$—in the aim of preventing patches for which no good match was found from undergoing heavy degradation. We check against $E^{\mathrm{b}}_{\mathbf{x}}(g_{\mathbf{x}})$ from (5.1) since it gives an indication of how satisfied the input points are with the match without penalizing the addition of new detail from $\mathcal{S}'_{\mathbf{x}}$. As in Section 5.2.2, if $|\mathcal{S}_{\mathbf{x}}| < 3$ then we consider $\mathbf{x}$ a flying pixel, and set $\omega_{\mathbf{x}} = 0$.

Post-processing. Since our polygon approximation guarantees only that the outlines of the polygon be within the SR factor of the outlines of the NN upscaled mask, it is possible that no overlay mask cover a given pixel $\hat{\mathbf{x}}$ at the target resolution. Such holes can be filled using morphological dilation carried out iteratively, with the dilation affecting only pixels identified as holes (cf. Figure 5.9). Another possible cause for holes is if pixels within an overlay mask could not be interpolated owing to the spatial distribution of the projected points. In that event, we dilate within the overlay mask with highest weight, again only over pixels identified as holes. Note that no post-processing was performed in the output in Figure 5.1. Occasional gentle overlap of overlay masks can cause streaking artifacts; if in the *input* point cloud fewer than 3 points lie within the radius r of the point $\mathbf{P}_{\hat{\mathbf{x}}}$ assigned to a pixel $\hat{\mathbf{x}}$ at the target resolution, we likewise treat $\hat{\mathbf{x}}$ as a hole and fill it using dilation in the manner outlined above.

5.3 Evaluation

We evaluate our method using depth data from stereo, ToF, laser scans and structured light. We carry out a quantitative evaluation in Section 5.3.1, and provide a qualitative evaluation in the section thereafter. Unless otherwise stated, we performed no preprocessing. In all our experiments, we carried out 5 iterations of PatchMatch, with $k = 3$. Setting appropriate parameters r, α, β, and γ is largely intuitive upon visualization of the input point cloud, and depends on the scale, density, and relative depth of point features one aims to capture. In Section 5.3.1, all algorithm parameters were kept identical across Middlebury and laser scan tests, respectively. We detail, in Figures 5.19-5.26, the fixed radius algorithm parameters used for the Middlebury data sets.

5.3.1 Quantitative Evaluation

Following the example of Mac Aodha et al. [MCNB12] we provide a quantitative evaluation of our technique on cones, teddy, tsukuba and venus of the Middlebury stereo data set (cf. Scharstein and Szeliski [SS03]). For Middlebury tests, we ran our algorithm on filled ground truth data—the same used in Mac Aodha et al.–downscaled by NN interpolation by a factor of 2 and 4 and subsequently super resolved by the same factor, respectively, which we compared to ground truth. In Table 5.1, we provide percent error scores—giving the percentage of pixels for which the absolute difference in disparity exceeds 1—for Middlebury. Table 5.2 shows root mean squared error (RMSE) scores. Note that although popular in the depth SR literature, RMSE scores over depth or disparity maps are dominated by misassignments at the boundaries of objects separated by large depth differences; given two data sets with equal percent error, a data set where boundaries are gently blurred will have lower RMSE than one with boundaries that are sharp. Even so, our RMSE scores fare very competitively with those of alternative techniques. In percent error, we are the top performer among example-based methods, and on a few occasions outperform the image-guided techniques. Among depth SR methods that leverage a color or intensity image at the target resolution, we compared against Diebel and Thrun [DT05] and Yang et al. [YYDN07]; among techniques that make use of an external database we compared against Mac Aodha et al., and against Yang et al. [YWHM10] and Freeman and Liu [FL11] from the image SR literature. Additionally, we compared against the 'single-image' image SR approach of Glasner et al. [GBI09]. For completeness, we compared against NN upscaling to provide a rough baseline, although it introduces jagged edges and does nothing to improve the apparent depth measurement accuracy. Table 5.2 additionally gives RMSE scores for three depth maps obtained from laser scans detailed in Mac Aodha et al., which were downscaled and subsequently super resolved by a factor of 4. For the laser scans we compared to the original resolution since ground truth data was not available. All RMSE and percent error scores were computed on 8 bit disparity maps. The data sets—with the exception of results on the algorithm of Glasner et al. [GBI09]—and the code used in carrying out the quantitative evaluation are from Mac Aodha et al. [MCNB12].[2]

[2]The RMSE scores published in Mac Aodha et al. [MCNB12] were subject to a subtle image resizing issue. Details and updated numbers are available at `http://visual.cs.ucl.ac.uk/pubs/depthSuperRes/supp/index.html`.

Percent Error. Let Ω denote the set of valid pixels in the ground truth disparity map, which is to say pixels for which a valid disparity is available. Moreover, let $\mathrm{disp}_{\hat{\mathbf{x}}}$ and $\mathrm{disp}_{\hat{\mathbf{x}}^{\mathrm{GT}}}$ denote the disparity at the pixel $\hat{\mathbf{x}}$ at the target resolution in a recovered—meaning, in our context, super resolved—disparity map and in a ground truth (GT) disparity map, respectively. The percent error is an error measure borrowed from the stereo literature (e.g., [SS02]), and gives the percentage of pixels $\hat{\mathbf{x}} \in \Omega$ for which the absolute difference $\left|\mathrm{disp}_{\hat{\mathbf{x}}} - \mathrm{disp}_{\hat{\mathbf{x}}}^{\mathrm{GT}}\right|$ exceeds 1:

$$\text{percent error} = \frac{1}{|\Omega|} \sum_{\hat{\mathbf{x}} \in \Omega} \begin{cases} 0 & \text{if } \left|\mathrm{disp}_{\hat{\mathbf{x}}} - \mathrm{disp}_{\hat{\mathbf{x}}}^{\mathrm{GT}}\right| \leq 1 \\ 1 & \text{otherwise} \end{cases} . \tag{5.5}$$

Note that percent error is applicable only where a disparity map is available.

Root Mean Squared Error (RMSE). Let Ω, $\mathrm{disp}_{\hat{\mathbf{x}}}$, and $\mathrm{disp}_{\hat{\mathbf{x}}^{\mathrm{GT}}}$ denote the same as for percent errror. The RMSE between a recovered disparity map and a GT disparity map—reusing the notation introduced above—is computed by

$$\text{disparity RMSE} = \sqrt{\frac{1}{|\Omega|} \sum_{\hat{\mathbf{x}} \in \Omega} \left(\mathrm{disp}_{\hat{\mathbf{x}}} - \mathrm{disp}_{\hat{\mathbf{x}}}^{\mathrm{GT}}\right)^2}. \tag{5.6}$$

The RMSE between a recovered depth map and a GT depth map is computed analogously,

$$\text{depth RMSE} = \sqrt{\frac{1}{|\Omega|} \sum_{\hat{\mathbf{x}} \in \Omega} \left(Z_{\hat{\mathbf{x}}} - Z_{\hat{\mathbf{x}}}^{\mathrm{GT}}\right)^2}, \tag{5.7}$$

where $Z_{\hat{\mathbf{x}}}$ and $Z_{\hat{\mathbf{x}}}^{\mathrm{GT}}$ denote the recovered—again, in our context meaning super resolved—and GT depth at pixel $\hat{\mathbf{x}}$, respectively.

5.3.2 Qualitative Evaluation

In Figure 5.10 we show results on a data set of two similar egg cartons situated at different depths, obtained using the stereo algorithm of Bleyer et al. [BRR11], and follow in Figures 5.11-5.14 with results using 2x nearest neighbor upscaling and 2x SR with our algorithm, the algorithm of Glasner et al. [GBI09], and that of Mac Aodha et al. [MCNB12], respectively. Input to NN upscaling and to the three SR algorithms is the 2x downscaled version of the disparity map in Figure 5.10. Our result is visually superior to that of our competitors, and is the only one to succeed in removing noise. Note the disturbing patch artifacts for Mac Aodha et al. In Figure 5.15, we consider a noisy ToF data set from [MCNB12]. We see that although our depth map appears pleasing, it in fact remains gently noisy if shaded as a mesh, owing to the great deal of noise in the input. However, if we apply the same bilateral filtering as Mac Aodha et al. [MCNB12], our result when shaded—although not as smooth over the vase—preserves edges better (e.g., at the foot) without introducing square patch artifacts. Note that Glasner et al. do not succeed in removing visible noise in their depth map, and introduce halo artifacts at the boundaries. Figure 5.16 provides a comparison over the noiseless, yet quantized Middlebury Cones data set. Note that although Glasner et al. [GBI09] perform well in RMSE, their method produces poor object boundaries, as can additionally be seen in Figures 5.19-5.26.

	Y1	FL	G	MA	Ours	NN	DT	Y2
Middlebury 2x								
Cones	61.617	6.266	4.697	2.935	**2.018**	1.713	3.800	2.346
Teddy	54.194	4.660	3.137	2.311	**1.862**	1.548	2.786	1.918
Tsukuba	5.566	3.240	3.234	2.235	**1.644**	1.240	2.745	1.161
Venus	46.985	0.790	0.940	0.536	**0.377**	0.328	0.574	0.250
Middlebury 4x								
Cones	63.742	15.077	8.790	6.541	**3.271**	3.121	7.452	4.582
Teddy	55.080	12.122	6.806	5.309	**4.234**	3.358	6.865	4.079
Tsukuba	7.649	10.030	6.454	4.780	**2.932**	2.197	5.118	2.565
Venus	47.053	3.348	1.770	**0.856**	3.245	0.609	1.236	0.421

Table 5.1: Percent error scores. Y1 = Yang et al. [YWHM10] (Y1), FL = Freeman and Liu [FL11], G = Glasner et al. [GBI09], MA = Mac Aodha et al. [MCNB12], NN = nearest neighbor upscaling, DT = Diebel and Thrun [DT05], Y2 = Yang et al. [YYDN07]. Our method is the top performer among example-based methods and on a few occasions outperforms Diebel and Thrun [DT05] and Yang et al. [YYDN07]. Results provided for Yang et al. [YWHM10] suffer from incorrect absolute intensities. Cell colors indicate ranking among the five methods, from best to worst: green, light green, yellow, orange, red. Gray cells are shown for comparison but are not included in the ranking.

5.4 Discussion

A major drawback of reasoning in terms of spheres of a single fixed radius r is that the sphere inlier count then tends to vary as a function of depth, as illustrated in Figures 5.17 and 6.2. To be supplied with a radius r for which spheres at large depth from the camera contain enough inliers for matching to be sensible—while at the same time not allowing spheres at small depth to contain so many points that patch upscaling and merging smoothes away interesting detail—is accordingly a key assumption of the algorithm. An even more important issue in what concerns our numerical results, however, is that patches that contain few inliers are prone to be discarded in the polygonalization step, as illustrated in Figure 5.18. Of course, the extent to which it is even possible to find a radius r that is neither too big nor too small is a function of the depth range of the given scene, and varies in general with every input depth map. Proposing a way to adaptively set the radius $r_{\mathbf{x}}$ of the sphere at each pixel $\mathbf{x}$ in terms of a fixed radius r_{pix} expressed instead in *pixel units* is one of the contributions of Hornáček et al. [HFR14], the details of which we defer to Chapter 6 and Appendix A. In addition to making sphere inlier counts more uniform, proceeding in this manner has the advantage of making the choice of radius more intuitive. In Tables 5.3 and 5.4, we compare our reported results from Tables 5.1 and 5.2, respectively, on Middlebury (i) for fixed radius $r = 1.35$ (in the manner of the published algorithm) with (ii) our results having fixed the radius of each sphere adaptively to $r_{\text{pix}} = 5$ and $r_{\text{pix}} = 10$. Additionally, on the intuition that the error measures considered in Section 5.3.1 are sensitive especially to errors at major object boundaries, we provide RMSE and percent error scores having fixed each

	Y1	FL	G	MA	Ours	NN	DT	Y2
Middlebury 2x								
Cones	2.027	1.447	**0.867**	1.127	0.994	1.094	0.740	0.756
Teddy	1.420	0.969	**0.596**	0.825	0.791	0.815	0.527	0.510
Tsukuba	0.705	0.617	**0.482**	0.601	0.580	0.612	0.401	0.393
Venus	0.992	0.332	**0.209**	0.276	0.257	0.268	0.170	0.167
Middlebury 4x								
Cones	2.214	1.536	1.483	1.504	**1.399**	1.531	1.141	0.993
Teddy	1.572	1.110	1.065	**1.026**	1.196	1.129	0.801	0.690
Tsukuba	0.840	0.869	0.832	0.833	**0.727**	0.833	0.549	0.514
Venus	1.012	0.367	0.394	**0.337**	0.450	0.368	0.243	0.216
Scan 4x								
Scan 21	0.030	0.019	1.851	**0.017**	0.021	0.018	N/A	N/A
Scan 30	0.035	**0.017**	1.865	**0.017**	0.018	0.016	N/A	N/A
Scan 42	0.054	0.075	1.764	0.045	**0.030**	0.040	N/A	N/A

Table 5.2: Root mean squared error (RMSE) scores. Yang et al. [YWHM10] (Y1) and Freeman and Liu [FL11] (FL) are image SR methods and Mac Aodha et al. [MCNB12] (MA) a depth SR method, all of which require an external database. Diebel and Thrun [DT05] (DT) and Yang et al. [YYDN07] (Y2) are depth SR methods that use an image at the target resolution. Glasner et al. [GBI09] (G) is an image SR technique that uses patches from within the input image. For most data sets, our method is competitive with the top performer. Laser scan tests on the image-guided techniques were not possible for want of images at the target resolution. Best score is indicated in bold for the example-based methods, which we consider our main competitors. Cell colors indicate ranking among the five methods, from best to worst: green, light green, yellow, orange, red. Gray cells are shown for comparison but are not included in the ranking.

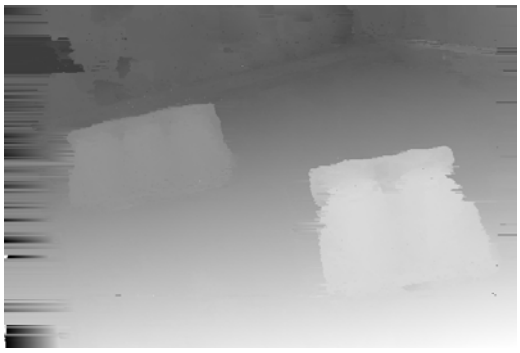
rigid body motion $g_{\mathbf{x}}$ to the identity motion $g_I = (\mathtt{I}, \mathbf{0})$, thereby placing the onus in the patch upscaling and merging step *solely on mask polygonalization*. Note that while setting each $g_{\mathbf{x}}$ to the identity motion keeps major object boundaries sharp and accordingly has little impact on numerical results, proceeding in such a manner fails to improve detail on object surfaces themselves (beyond perhaps removing shot noise), as exemplified in Figure 5.27.

5.5 Conclusion

Inspired by the work of Glasner et al. [GBI09] on 'single-image' super resolution for color and intensity images, we presented a tailored 'single-image' depth super resolution algorithm, reasoning in terms of only the information contained in the single input depth map. We introduced a new 3D variant of the PatchMatch algorithm [BSFG09, BSGF10] for recovering a dense matching between pairs of closer-further corresponding 3D point patches related by 6 DoF 3D rigid body motions, and presented a technique for upscaling and merging matched patches that predicts sharp object boundaries at the target resolution. In our evaluation, we showed our results

(a) Rectified left image.

(b) Rectified right image.

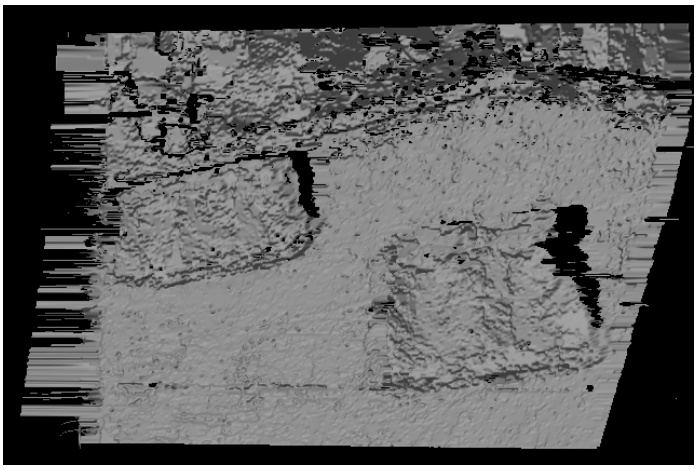


(c) Disparity map.

Figure 5.10: The egg carton data set stereo pair. A disparity map is computed for the rectified left image using the algorithm of Bleyer et al. [BRR11]. The output disparity map is downscaled by a factor of 2 (which can be seen scaled back up by 2x nearest neighbor upscaling in Figure 5.11a) before being provided as input to our algorithm and the algorithms of Glasner et al. [GBI09] and Mac Aodha et al. [MCNB12].

to be highly competitive with methods that make use of ancillary data such as a color image at the target resolution or a database of high-resolution depth exemplars.

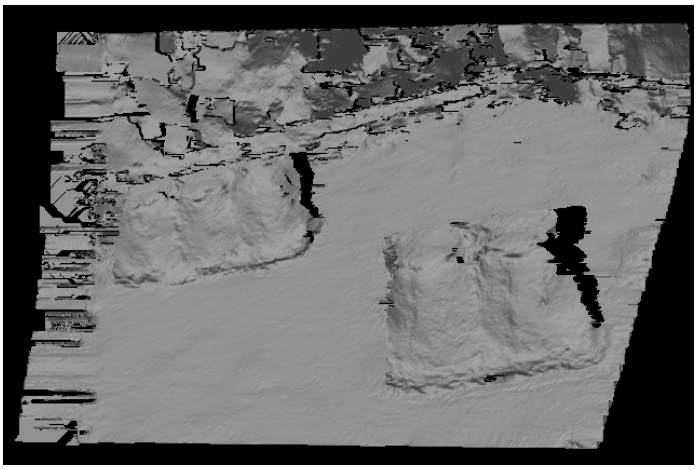
(a) Output as disparity map.



(b) Output as shaded mesh.

Figure 5.11: 2x nearest neighbor upscaling of the 2x downscaled disparity map of the egg carton data set, which we use as input to our algorithm and to the algorithms of Glasner et al. [GBI09] and Mac Aodha et al. [MCNB12].

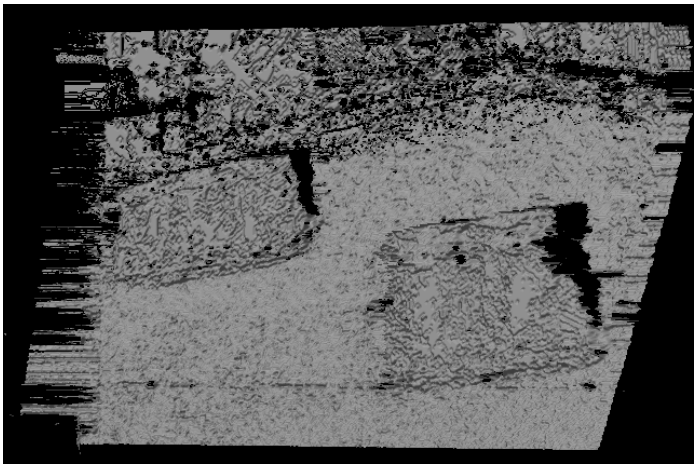
(a) Output as disparity map.



(b) Output as shaded mesh.

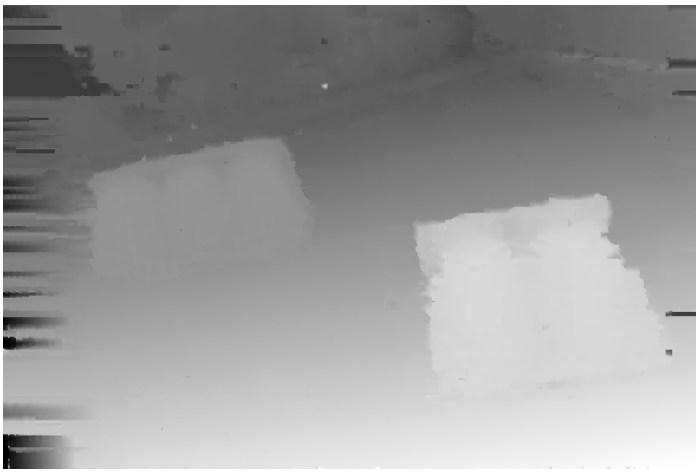
Figure 5.12: Our 2x SR result on the egg carton data set. Our algorithm succeeds in removing noise better than our competitors, while contours in our disparity map come closest to the contours of the original disparity map.

(a) Output as disparity map.

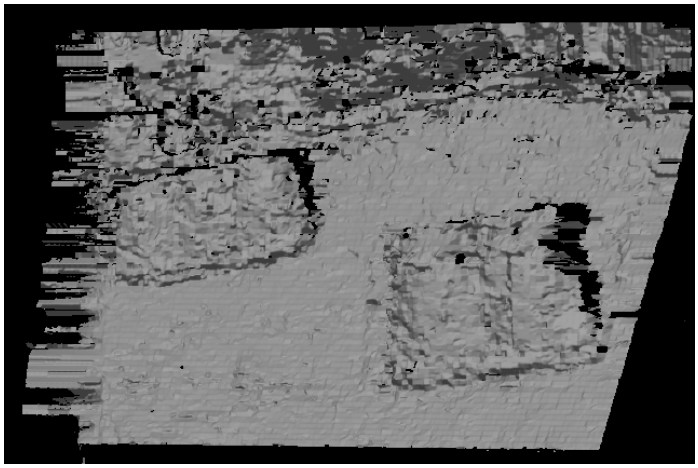


(b) Output as shaded mesh.

Figure 5.13: 2x SR result of Glasner et al. [GBI09] on the egg carton data set. While edges in the disparity map are smooth, the algorithm fails to remove noise.



(a) Output as disparity map.

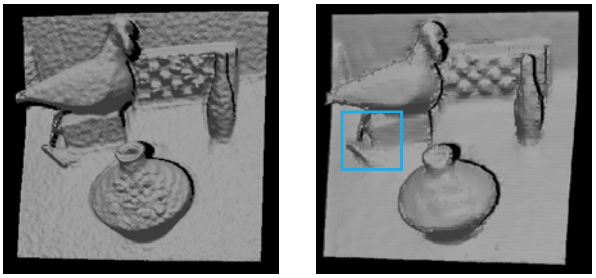


(b) Output as shaded mesh.

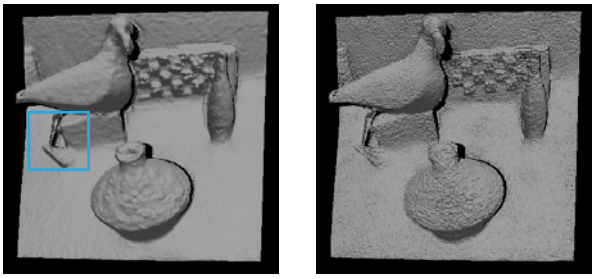
Figure 5.14: 2x SR result of Mac Aodha et al. [MCNB12] on the egg carton data set (preprocessed). While edges in the disparity map are more smooth than by using nearest neighbor upscaling, the algorithm introduces disturbing patch artifacts clearly visible in the shaded mesh.

(a) Nearest neighbor. (b) Our result. (c) Mac Aodha et al. [MCNB12] (PP).

(d) Glasner et al. [GBI09]. (e) Yang et al. [YWHM10]. (f) Freeman and Liu [FL11].

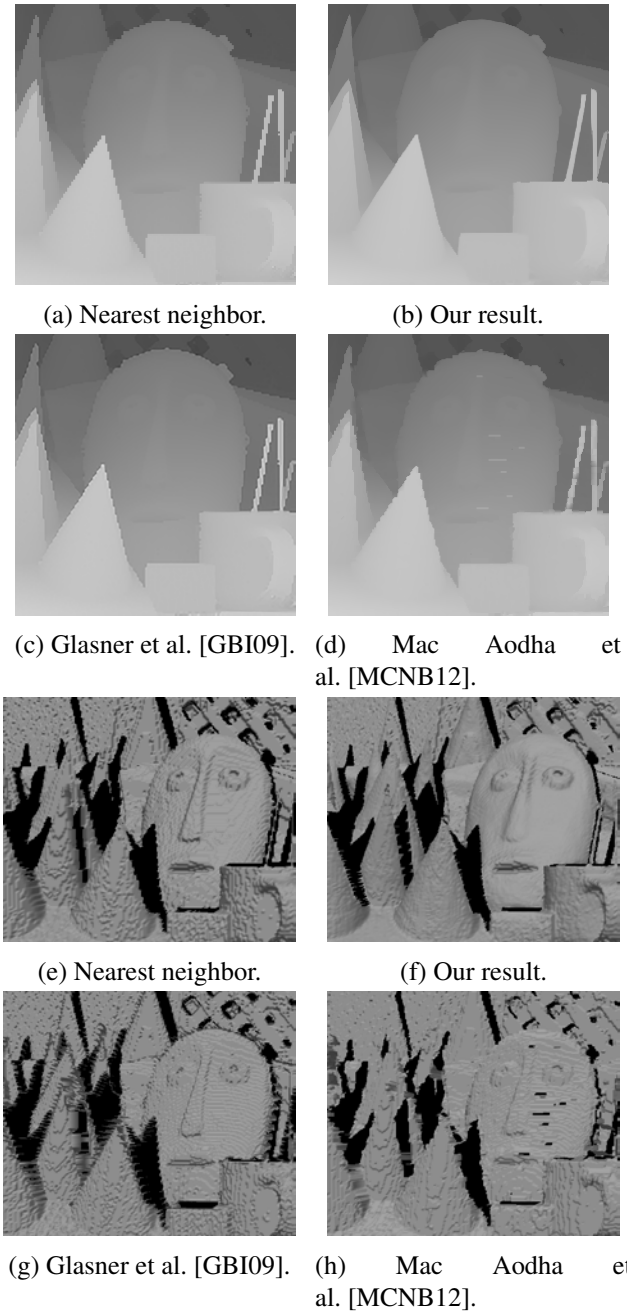


(g) Nearest neighbor (PP). (h) Mac Aodha et al. [MCNB12] (PP).



(i) Our result (PP). (j) Our result (not PP).

Figure 5.15: Results on the ToF dove data set. Above, we provide zoom ins on a region of interest of the noisy PMD CamCube 2.0 ToF data set shown in Figure 5.2b for 4x nearest neighbor upscaling in (a) and 4x SR otherwise. A depth map zoom in for Mac Aodha et al. was available only with bilateral preprocessing (window size 5, spatial deviation 3, range deviation 0.1). Below, we show shaded meshes for the preprocessed result of Mac Aodha et al. and for our method with and without the same preprocessing ((h) is not aligned with the other meshes because the rendering is not ours). Note that although we in (i) perform worse than (h) on the vase, we preserve fine detail better and do not introduce square patch artifacts. PP = preprocessed.



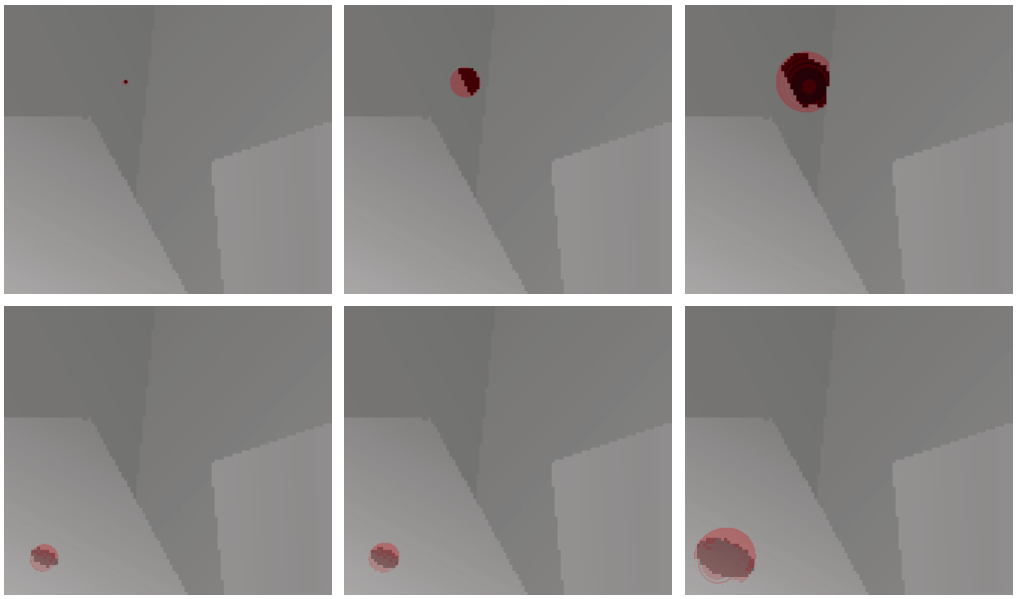
(a) Nearest neighbor. (b) Our result.

(c) Glasner et al. [GBI09]. (d) Mac Aodha et al. [MCNB12].

(e) Nearest neighbor. (f) Our result.

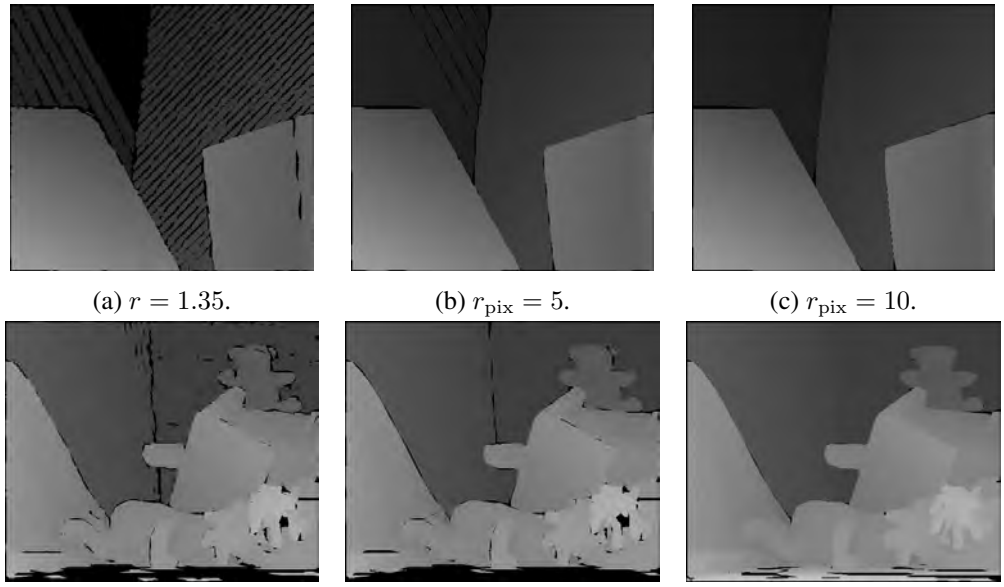
(g) Glasner et al. [GBI09]. (h) Mac Aodha et al. [MCNB12].

Figure 5.16: Results on the Middlebury cones data set. Above, zoom ins on a region of interest of the noiseless, though quantized Middlebury Cones data set. 2x SR was carried out (in our case, using the parameters from the quantitative evaluation) on the 2x nearest neighbor downscaling of the original, depicted in (a). Our method has the sharpest edges. Below, the corresponding shaded meshes. Our method performs the best smoothing even after quantization (particularly at the cones), although it lightly smooths away the nose for the parameters used.



(a) $r = 1.35$. (b) $r_{\text{pix}} = 5$. (c) $r_{\text{pix}} = 10$.

Figure 5.17: Apparent spatial extent of 3D point patches on venus. Note that for fixed radius $r = 1.35$, the sphere at large depth (top row) contains only a single inlier; at small depth (bottom row), the spatial extent of the sphere is almost the same as for $r_{\text{pix}} = 5$. In general, a challenge of reasoning in terms of a single fixed radius r is to ensure that spheres at large depth from the camera contain enough inliers for matching to be sensible, while at the same time not allowing spheres at small depth to contain so many points that patch upscaling and merging smoothes away interesting detail.



(a) $r = 1.35$. (b) $r_{\mathrm{pix}} = 5$. (c) $r_{\mathrm{pix}} = 10$.

(d) $r = 1.35$. (e) $r_{\mathrm{pix}} = 5$. (f) $r_{\mathrm{pix}} = 10$.

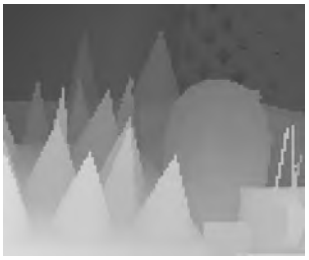
Figure 5.18: Before post-processing for 4x SR on the venus and teddy data sets. Unfilled areas—indicated in black, and subsequently to be filled by post-processing—arise because polygonalization discards pixels that form a mask that at the source resolution is only one pixel high or wide. Note that it is due to these large unfilled areas that performance is poor for 4x SR on venus and teddy for $r = 1.35$.

	$r_{\text{pix}} = 5$	$r_{\text{pix}} = 5, g_I$	$r_{\text{pix}} = 10$	$r_{\text{pix}} = 10, g_I$	$r = 1.35$
Middlebury 2x					
Cones	**1.926**	**1.702**	**2.018**	**1.772**	2.018
Teddy	3.555	3.344	**1.845**	**1.658**	1.862
Tsukuba	**1.641**	**1.472**	**1.634**	**1.520**	1.644
Venus	**0.340**	**0.239**	**0.336**	**0.269**	0.377
Middlebury 4x					
Cones	**3.162**	**2.541**	**3.294**	**2.934**	3.271
Teddy	**4.667**	**4.667**	**3.218**	**3.190**	4.234
Tsukuba	**2.901**	**2.464**	**2.877**	**2.647**	2.932
Venus	**0.542**	**0.542**	**0.223**	**0.218**	3.245

Table 5.3: Supplemental percent error scores for our approach, with $r_{\text{pix}} = 5$ and with $r_{\text{pix}} = 10$, such that each $g_{\mathbf{x}}$ be recovered using our algorithm and each $g_{\mathbf{x}}$ be set to the identity motion g_I, respectively. Green cell color indicates that the result performs at least as well in percent error as our result for fixed radius $r = 1.35$ from Table 5.1; red indicates that the performance is worse. Bold indicates that, disregarding our result for fixed radius $r = 1.35$, the result outperforms all other competitors in Table 5.1.

	$r_{\text{pix}} = 5$	$r_{\text{pix}} = 5, g_I$	$r_{\text{pix}} = 10$	$r_{\text{pix}} = 10, g_I$	$r = 1.35$
Middlebury 2x					
Cones	1.017	1.113	0.988	1.091	0.994
Teddy	0.986	1.006	0.800	0.784	0.791
Tsukuba	0.556	0.610	0.571	0.597	0.580
Venus	0.252	0.239	0.221	0.237	0.257
Middlebury 4x					
Cones	**1.405**	**1.402**	**1.315**	**1.332**	1.399
Teddy	**1.279**	**1.279**	**1.013**	**1.003**	1.196
Tsukuba	**0.719**	0.749	**0.736**	**0.770**	0.727
Venus	**0.258**	**0.258**	**0.220**	**0.227**	0.450

Table 5.4: Supplemental root mean squared error (RMSE) scores for our approach, with $r_{\text{pix}} = 5$ and with $r_{\text{pix}} = 10$, such that each $g_{\mathbf{x}}$ be recovered using our algorithm and each $g_{\mathbf{x}}$ be set to the identity motion g_I, respectively. Green cell color indicates that the result performs at least as well in RMSE as our result for fixed radius $r = 1.35$ from Table 5.2; red indicates that the performance is worse. Bold indicates that, disregarding our result for fixed radius $r = 1.35$, the result outperforms all other competitors in Table 5.2.



(a) Nearest neighbor.

(b) Our result for fixed radius $r = 1.35$ (published alg.).

(c) Our result for $r_{\text{pix}} = 5$.

(d) Our result for $r_{\text{pix}} = 10$.

(e) Our result for $r_{\text{pix}} = 5$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.

(f) Our result for $r_{\text{pix}} = 10$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.

(g) Glasner et al. [GBI09].

(h) Mac Aodha et al. [MCNB12].

Figure 5.19: 4x SR results on cones, with focal length $f = 150$, baseline $B = 10$, $\alpha = 0.5$, $\beta = 0.025$, $\gamma = 10$ and disparities of the 4x NN downscaled original disparity map divided by a factor of 4. Given this choice of focal length and baseline, sphere radius at minimum, median, and maximum depth Z for $r_{\text{pix}} = 1$ is $0.181818, 0.314961, 0.655738$, respectively.

(a) Nearest neighbor.

(b) Our result for fixed radius $r = 1.35$ (published alg.).

(c) Our result for $r_{\text{pix}} = 5$.

(d) Our result for $r_{\text{pix}} = 10$.

(e) Our result for $r_{\text{pix}} = 5$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.

(f) Our result for $r_{\text{pix}} = 10$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.

(g) Glasner et al. [GBI09].

(h) Mac Aodha et al. [MCNB12].

Figure 5.20: 4x SR results on teddy, with focal length $f = 150$, baseline $B = 10$, $\alpha = 0.5$, $\beta = 0.025$, $\gamma = 10$ and disparities of the 4x NN downscaled original disparity map divided by a factor of 4. Given this choice of focal length and baseline, sphere radius at minimum, median, and maximum depth Z for $r_{\text{pix}} = 1$ is $0.190476, 0.322581, 0.666667$, respectively.

(a) Nearest neighbor.

(b) Our result for fixed radius $r = 1.35$ (published alg.).

(c) Our result for $r_{\text{pix}} = 5$.

(d) Our result for $r_{\text{pix}} = 10$.

(e) Our result for $r_{\text{pix}} = 5$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.

(f) Our result for $r_{\text{pix}} = 10$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.

(g) Glasner et al. [GBI09].

(h) Mac Aodha et al. [MCNB12].

Figure 5.21: 4x SR results on tsukuba, with focal length $f = 150$, baseline $B = 10$, $\alpha = 0.5$, $\beta = 0.025$, $\gamma = 10$ and disparities of the 4x NN downscaled original disparity map divided by a factor of 4. Given this choice of focal length and baseline, sphere radius at minimum, median, and maximum depth Z for $r_{\text{pix}} = 1$ is $0.178571, 0.5, 0.5$, respectively.

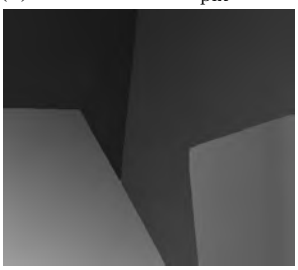
(a) Nearest neighbor.

(b) Our result for fixed radius $r = 1.35$ (published alg.).

(c) Our result for $r_{\text{pix}} = 5$.

(d) Our result for $r_{\text{pix}} = 10$.

(e) Our result for $r_{\text{pix}} = 5$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.



(f) Our result for $r_{\text{pix}} = 10$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.

(g) Glasner et al. [GBI09].

(h) Mac Aodha et al. [MCNB12].

Figure 5.22: 4x SR results on venus, with focal length $f = 150$, baseline $B = 10$, $\alpha = 0.5$, $\beta = 0.025$, $\gamma = 10$ and disparities of the 4x NN downscaled original disparity map divided by a factor of 4. Given this choice of focal length and baseline, sphere radius at minimum, median, and maximum depth Z for $r_{\text{pix}} = 1$ is $0.254777, 0.677966, 1.66667$, respectively. Note that for $r = 1.35$, around maximum depth spheres contain only a single inlier, causing polygonalization to discard the corresponding singleton patches of points and thereby leading to the corresponding areas of the SR output in (b) to be filled purely by post-processing.

(a) Nearest neighbor.

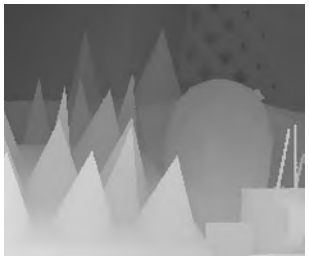
(b) Our result for fixed radius $r = 1.35$ (published alg.).

(c) Our result for $r_{\text{pix}} = 5$.

(d) Our result for $r_{\text{pix}} = 10$.

(e) Our result for $r_{\text{pix}} = 5$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.

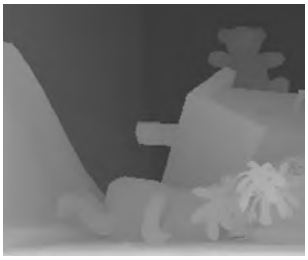
(f) Our result for $r_{\text{pix}} = 10$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.



(g) Glasner et al. [GBI09].

(h) Mac Aodha et al. [MCNB12].

Figure 5.23: 2x SR results on cones, with focal length $f = 300$, baseline $B = 10$, $\alpha = 0.5$, $\beta = 0.025$, $\gamma = 10$ and disparities of the 4x NN downscaled original disparity map divided by a factor of 4. Given this choice of focal length and baseline, sphere radius at minimum, median, and maximum depth Z for $r_{\text{pix}} = 1$ is $0.0909091, 0.15625, 0.327869$, respectively.



(a) Nearest neighbor.

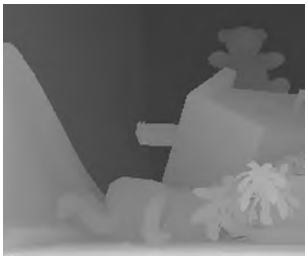
(b) Our result for fixed radius $r = 1.35$ (published alg.).

(c) Our result for $r_{\text{pix}} = 5$.

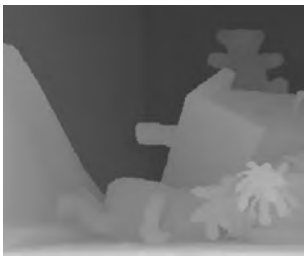
(d) Our result for $r_{\text{pix}} = 10$.

(e) Our result for $r_{\text{pix}} = 5$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.

(f) Our result for $r_{\text{pix}} = 10$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.



(g) Glasner et al. [GBI09].



(h) Mac Aodha et al. [MCNB12].

Figure 5.24: 2x SR results on teddy, with focal length $f = 300$, baseline $B = 10$, $\alpha = 0.5$, $\beta = 0.025$, $\gamma = 10$ and disparities of the 2x NN downscaled original disparity map divided by a factor of 2. Given this choice of focal length and baseline, sphere radius at minimum, median, and maximum depth Z for $r_{\text{pix}} = 1$ is $0.0947867, 0.16129, 0.344828$, respectively.

(a) Nearest neighbor.

(b) Our result for fixed radius $r = 1.35$ (published alg.).

(c) Our result for $r_{\text{pix}} = 5$.

(d) Our result for $r_{\text{pix}} = 10$.

(e) Our result for $r_{\text{pix}} = 5$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.

(f) Our result for $r_{\text{pix}} = 10$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.

(g) Glasner et al. [GBI09].

(h) Mac Aodha et al. [MCNB12].

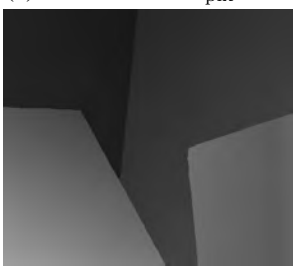
Figure 5.25: 2x SR results on tsukuba, with focal length $f = 300$, baseline $B = 10$, $\alpha = 0.5$, $\beta = 0.025$, $\gamma = 10$ and disparities of the 4x NN downscaled original disparity map divided by a factor of 4. Given this choice of focal length and baseline, sphere radius at minimum, median, and maximum depth Z for $r_{\text{pix}} = 1$ is $0.0892857, 0.25, 0.25$, respectively.

(a) Nearest neighbor.

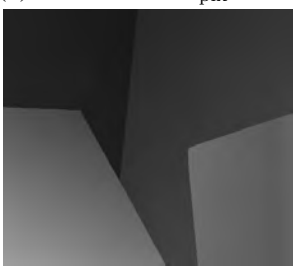
(b) Our result for fixed radius $r = 1.35$ (published alg.).

(c) Our result for $r_{\text{pix}} = 5$.

(d) Our result for $r_{\text{pix}} = 10$.



(e) Our result for $r_{\text{pix}} = 5$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.

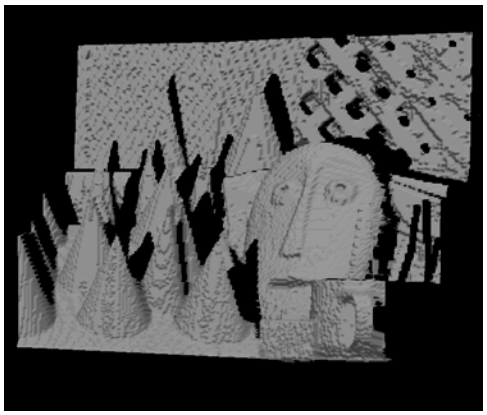


(f) Our result for $r_{\text{pix}} = 10$ with $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.

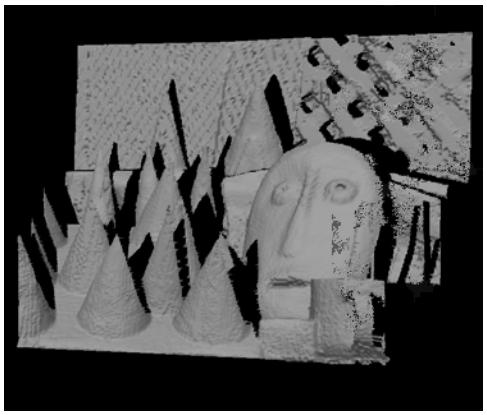
(g) Glasner et al. [GBI09].

(h) Mac Aodha et al. [MCNB12].

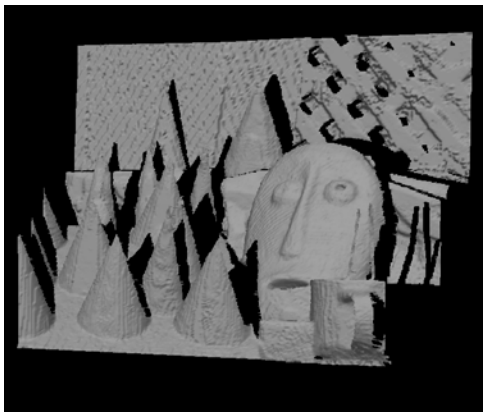
Figure 5.26: 2x SR results on venus, with focal length $f = 300$, baseline $B = 10$, $\alpha = 0.5$, $\beta = 0.025$, $\gamma = 10$ and disparities of the 4x NN downscaled original disparity map divided by a factor of 4. Given this choice of focal length and baseline, sphere radius at minimum, median, and maximum depth Z for $r_{\text{pix}} = 1$ is $0.127389, 0.338983, 0.833333$, respectively.



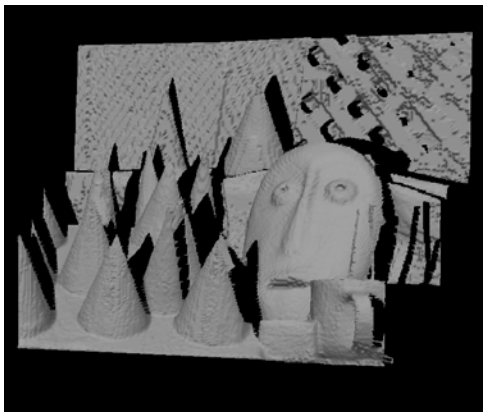
(a) 2x nearest neighbor.



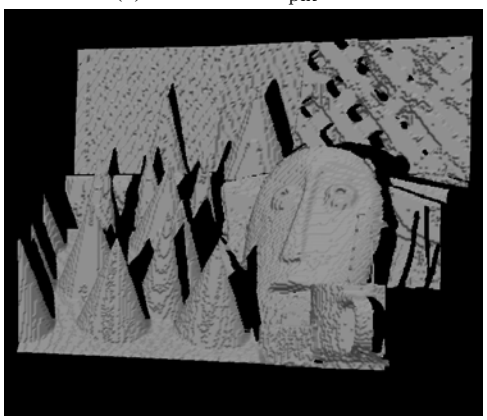
(b) 2x SR with $r = 1.35$.



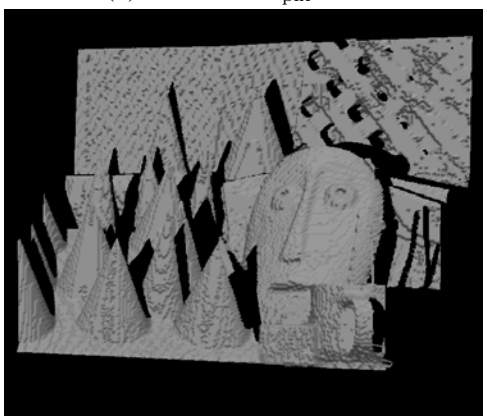
(c) 2x SR with $r_{\text{pix}} = 5$.



(d) 2x SR with $r_{\text{pix}} = 10$.



(e) 2x SR with $r_{\text{pix}} = 5$ and $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.



(f) 2x SR with $r_{\text{pix}} = 10$ and $g_{\mathbf{x}} = g_I, \forall \mathbf{x}$.

Figure 5.27: Meshes for our 2x SR on cones at various radii. Note that for our results with $g_{\mathbf{x}}$ fixed to the identity motion g_I for all pixels $\mathbf{x}$, object surfaces do not appear to undergo any improvement relative to NN upscaling.

CHAPTER 6

RGB-D Scene Flow

In this chapter, we take a new approach to computing dense scene flow between a pair of consecutive RGB-D frames. We exploit the availability of depth data by seeking correspondences with respect to patches specified not as the pixels inside traditional square windows in image space, but—as in Chapter 5—in terms of the 3D points that are the respective inliers of spheres in world space. Our primary contribution is to show that by reasoning in terms of such patches of points undergoing 6 DoF rigid body motions in 3D, we succeed in obtaining compelling scene flow results at displacements large and small without relying on either of two simplifying assumptions that pervade much of the earlier literature: brightness constancy or local surface planarity. As a consequence of our approach, our output is a dense field of 3D rigid body motions, in contrast to the 3D translations that are the norm in scene flow. In Section 6.1, we introduce our methodology and place our algorithm in the context of earlier work. We then proceed to detail out algorithm in Section 6.2, and show, in Section 6.3, attractive flow results on challenging synthetic and real-world scenes that push the practical limits of the aforementioned assumptions. In Section 6.4 we attempt to emphasize the key elements of the algorithm in what concerns performance, and end with concluding remarks in Section 6.5.

6.1 Introduction

The growing consumer-level availability of RGB-D data—in particular since the introduction of the inexpensive Microsoft Kinect camera—has made solving computer vision problems by jointly exploring cues in color and depth an increasingly practical pursuit. We present a substantially new way of computing the dense 3D motion field between a pair of consecutive RGB-D frames of a (perhaps non-rigidly) moving scene, making neither of the traditional assumptions of brightness constancy or local surface planarity. The assumption that corresponding pixels have the same color is perhaps the most common simplifying assumption in scene flow, and breaks down as displacements become large. An alternative it to aggregate intensity information over patches of pixels. Traditionally, however, patch-based methods have relied on motion mod-

els that assume local surface planarity, which imposes limits on the size of such patches over geometry that is not planar.
In stark contrast to previous patch-based scene flow techniques, ours exploits the availability of 3D points in RGB-D data by reasoning in terms of so-called '3D point patches'. Our **main contributions** derive from how we reason about patches. By identifying the points constituting our patches as the inliers of *spheres* and relating such patches using 6 DoF 3D rigid body motions, we are able to: (i) overcome the local surface planarity assumption, allowing for larger patches over non-planar surfaces than are possible with traditional motion models; (ii) introduce a 6 DoF consistency check for the flow recovered in both directions; (iii) introduce a patchwise silhouette check to help reason about alignments in occlusion areas; and (iv) introduce an intuitive local rigidity prior to promote smoothness. Finally, a consequence of our approach is that (v) our output is a dense field of 6 DoF 3D rigid body motions, rather than one of 3D translation vectors as is the norm in scene flow.

6.1.1 Related Work

The term 'scene flow' is due to Vedula et al. [VBR+05], who introduced it as a dense field of 3D translation vectors for each point in the scene. Traditionally, scene flow has referred to techniques that take RGB images as input and recover 3D structure in addition to 3D motion. Among these techniques, we briefly mention in this section only work against which we evaluate our algorithm in Section 6.3; we refer the reader to a broader treatment in Section 4.4. RGB scene flow methods against which we evaluate our algorithm comprise Huguet and Devernay [HD07] and Basha et al. [BMK13]. Huguet and Devernay and Basha et al. both use a variational technique. RGB-D scene flow methods in our evaluation comprise Hadfield and Bowden [HB13], Herbst et al. [HRF13], and Quiroga et al. [QDC13]. The approach of Hadfield and Bowden operates instead on pairs of RGB-D frames, and uses a particle filter; Herbst et al. and Quiroga et al. both compute RGB-D scene flow using a variational technique. Each one of these methods invokes the assumption of brightness constancy.

6.2 Algorithm

We begin by obtaining a dense 6 DoF correspondence field in both directions (frame 1 to frame 2, frame 2 to frame 1), using a new variant of the PatchMatch algorithm building on the work of Barnes et al. [BSFG09, BSGF10], Hornáček et al. [HRGR13] (described in Chapter 5), and Bleyer et al. [BRR11]. We detail this step in Section 6.2.2. In Section 6.2.1, we present our 3D point patches and the matching cost we aim to minimize, from which the majority of our contributions derive. In Section 6.2.3, we detail the second step of our algorithm, optimized using α-expansion [BVZ01], which refines the correspondence fields by handling occlusions and promoting 6 DoF smoothness.

6.2.1 3D Point Patches

We carry out our correspondence search by reasoning in terms of so-called '3D point patches' undergoing 6 DoF 3D rigid body motions as in Hornáček et al. [HRGR13], but extend the

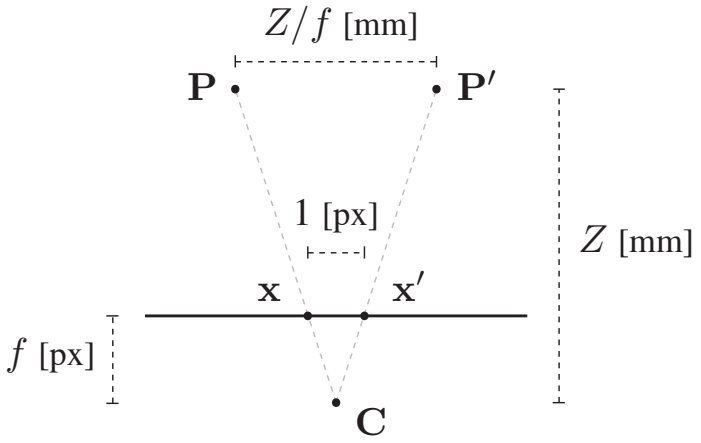


Figure 6.1: The Euclidean distance Z/f in world space between two points $\mathbf{P}, \mathbf{P}' \in \mathbb{R}^3$ both situated at depth Z—with respect to the camera center $\mathbf{C}$—and projecting to neighboring pixels $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^2$ in image space, respectively, obtained by similar triangles.

matching cost presented in their formulation to include the influence of 2D image gradients recoverable from the available RGB. Image gradients allow for matching salient texture features without relying on brightness constancy, while our 3D point patch formulation allows for better alignment of such gradients over non-planar surfaces than is possible using traditional motion models that assume local surface planarity (cf. Section 2.2). Additionally, our formulation allows us to compare depth between potential correspondences in a manner that borrows from the ICP literature (cf. Rusinkiewicz and Levoy [RL01]), which would also not be possible with traditional image space motion models. Recognizing that object boundaries encoded in depth can be noisy or poorly aligned with RGB, we additionally integrate an optional adaptive support weighting scheme in our matching cost. Finally, we overcome a shortcoming of the formulation in [HRGR13] that effectively tied the number of inlier points constituting a 3D point patch to its corresponding sphere's depth (cf. Figure 6.2), enabling us to ensure a more uniform matching quality across the scene.

Formal Definition. Let $g_{\mathbf{x}} = (\mathtt{R}_{\mathbf{x}}, \mathbf{t}_{\mathbf{x}})$ denote a 6 DoF rigid body motion in 3D to be assigned to the pixel $\mathbf{x} = (x, y)^\top$ by our algorithm. Given one of the two input views as the *source* view, and the other as the *destination* view, let $\mathcal{P} \subset \mathbb{R}^3$ denote the set of 3D points encoded in the depth map of the source view, and the *3D point patch* $\mathcal{S}_{\mathbf{x}}$ denote the subset of points in $\mathcal{P}$ situated within a radius $r_{\mathbf{x}}$ of the point $\mathbf{P}_{\mathbf{x}} = Z_{\mathbf{x}} \cdot \mathtt{K}^{-1}(\mathbf{x}^\top, 1)^\top \in \mathbb{R}^3$, where $Z_{\mathbf{x}}$ is the depth encoded at the pixel $\mathbf{x}$ and $\mathtt{K}$ is the 3×3 camera calibration matrix (cf. Section 2.4). Our goal is to assign a rigid body motion $g_{\mathbf{x}}$ to each valid pixel $\mathbf{x}$, mapping the 3D point patch $\mathcal{S}_{\mathbf{x}}$ in the source view to its counterpart in the destination view. A pixel $\mathbf{x}$ is deemed *valid* only if a depth value is encoded at $\mathbf{x}$ in the input depth map. In practice, we carry out spatial queries using a kd-tree [ML14].

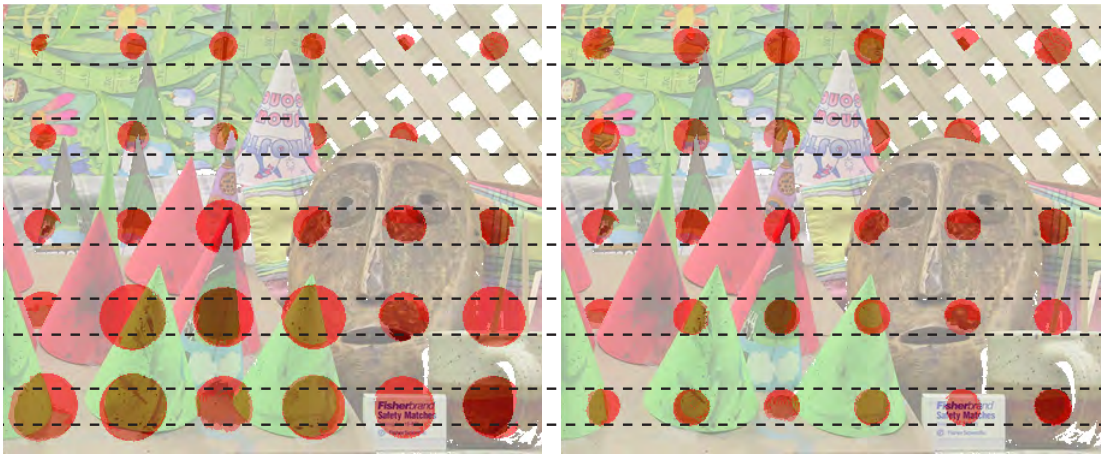


Figure 6.2: Apparent spatial extent of 3D point patches on cones. Left: fixed sphere radius r renders apparent spatial extent a function of sphere depth. Right: proposed per-pixel radius $r_{\mathbf{x}}$, obtained as a function of the depth $Z_{\mathbf{x}}$ of $\mathbf{P}_{\mathbf{x}}$. In this manner, spheres can be expected to contain a more uniform number of inlier points than for fixed r, thereby providing more uniform matching quality across the scene.

Radius $r_{\mathbf{x}}$. It can be shown by similar triangles that the distance in world space between two points both situated at depth Z projecting to neighboring pixels in image space is given by Z/f (cf. Figure 6.1; an alternative derivation is provided in Appendix A), where f is the camera's focal length in units of pixels. Accordingly, the number of points constituting a 3D point patch given a fixed radius r can on average be expected to decrease as depth increases, and with it so too can confidence in the strength of any matching score based on such points (cf. Figure 6.2). In order to alleviate this problem of reasoning in terms of a fixed radius r, we assign to each pixel $\mathbf{x}$ a tailored radius $r_{\mathbf{x}}$, obtained as a function of the depth $Z_{\mathbf{x}}$ encoded at $\mathbf{x}$ in the depth map, such that all resulting spheres appear to have the same size from the viewpoint of the camera:

$$r_{\mathbf{x}} = r_{\text{pix}} \cdot Z_{\mathbf{x}}/f, \tag{6.1}$$

where r_{pix} is a fixed radius *in pixels* and f is the camera's focal length, likewise in units of pixels. The quantity $Z_{\mathbf{x}}/f$ gives the distance in world space corresponding to a one-pixel displacement in image space. In this manner, each sphere appears to have equal size from the viewpoint of the camera, and thus the spheres can be expected to contain a more uniform number of inlier points than for fixed r.

Matching Cost. Let I, I' and G, G' be the source and destination color and gradient images, respectively, and let π, π' denote projection (cf. Section 2.4) into the source and destination views. We compute gradient similarity by projection and interpolation, promoting sub-pixel accuracy with respect to salient texture edges in image space:

$$E_{\mathbf{x}}^{\text{gr}}(g) = \sum_{\mathbf{P} \in \mathcal{S}_{\mathbf{x}}} w\big(\mathbf{x}, \pi(\mathbf{P})\big) \cdot \big\| G(\pi(\mathbf{P})) - G'(\pi'(g(\mathbf{P})))\big\|_2^2, \tag{6.2}$$

where $w(\mathbf{x}, \mathbf{x}') = \exp\big(-\|(I(\mathbf{x}) - I(\mathbf{x}'))\|_2/\gamma\big)$ implements a form of adaptive support weighting (cf. Yoon and Kweon [YK05]), which is valuable when object boundaries in the depth map are noisy or poorly aligned with RGB. We compute L2 color distance for adaptive support weighting in the CIE L*a*b color space. Let $\mathrm{NN}_{\mathcal{S}}(\mathbf{P})$ denote the nearest neighbor point to $\mathbf{P}$ in the set $\mathcal{S} \subset \mathbb{R}^3$. We compute 3D point similarity by

$$E^{\mathrm{pt}}_{\mathbf{x}}(g) = \sum_{\mathbf{P} \in \mathcal{S}_{\mathbf{x}}} w\big(\mathbf{x}, \pi(\mathbf{P})\big) \cdot \big\|g(\mathbf{P}) - \mathrm{NN}_{\mathcal{P}'}\big(g(\mathbf{P})\big)\big\|_2^2, \tag{6.3}$$

where $\mathcal{P}'$ denotes the set of points encoded in the depth map of the destination view. Reasoning in terms of nearest neighbors in 3D for point similarity allows for handling shot noise and invalid pixels without special treatment, which would not be possible by projection and interpolation. Moreover, it allows for a natural delineation of boundaries at depth discontinuities, insofar as object boundaries encoded in the input depth maps are of reasonable quality. We compute the final matching cost according to

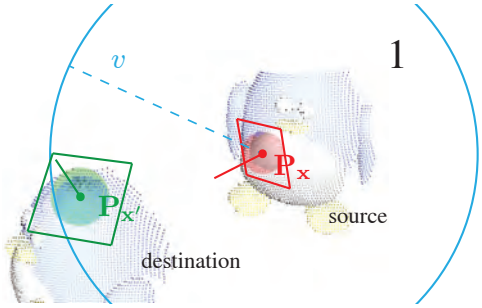
$$E_{\mathbf{x}}(g) = \begin{cases} E^{\mathrm{pt}}_{\mathbf{x}}(g) + \alpha \cdot E^{\mathrm{gr}}_{\mathbf{x}}(g) & \text{if } \mathbf{x} \text{ is valid} \\ \infty & \text{otherwise} \end{cases}, \tag{6.4}$$

where α is a fixed weight.

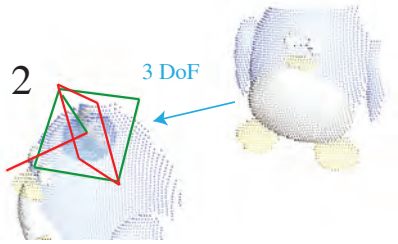
6.2.2 Dense 6 DoF Matching via PatchMatch

We turn to PatchMatch (cf. Barnes et al. [BSFG09, BSGF10]) to carry out our dense 6 DoF matching, building primarily upon the PatchMatch variant introduced in Hornáček et al. [HRGR13] for depth super resolution, and upon that introduced in Bleyer et al. [BRR11] for stereo. Our goal is to assign a rigid body motion $g_{\mathbf{x}}$ to each valid pixel $\mathbf{x}$, mapping the 3D point patch $\mathcal{S}_{\mathbf{x}}$ in the source view to its counterpart in the destination view. We begin with a semi-random initialization step, assigning a first guess to each valid pixel. Next, for i iterations, we visit each $\mathbf{x}$, carrying out (i) spatial propagation, (ii) j additional semi-random initializations, (iii) k refinements, and (iv) view propagation. In each of the steps (i-iv), a candidate rigid body motion is adopted at $\mathbf{x}$ if doing so yields equal or lesser matching cost. For the first of the two views, we visit its pixels in scanline order, upper left to lower right for even iterations, lower right to upper left for odd. A contribution of ours—applicable to stereo as well as to scene flow—is to promote convergence via view propagation by traversing the pixels of the second view *in the opposite order*, *in parallel* with the first view. We describe the individual steps in greater detail below.

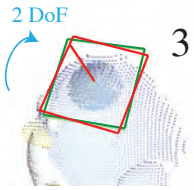
Semi-Random Initialization. For each valid pixel $\mathbf{x}$ in the source view, we randomly pick a point $\mathbf{P}_{\mathbf{x}'}$ in the destination view within a search radius v of $\mathbf{P}_{\mathbf{x}}$, giving a 3D translation vector $\mathbf{P}_{\mathbf{x}'} - \mathbf{P}_{\mathbf{x}}$ (3 DoF). Additionally, we obtain candidate 3D translations by carrying out SURF feature matching [BTG06] in image space. We obtain the remaining 3 DoF by computing the rotation minimizing arc length between the surface normal vector at $\mathbf{P}_{\mathbf{x}}$ and that at $\mathbf{P}_{\mathbf{x}'}$ (2 DoF), and choosing a random around-normal angular perturbation (1 DoF). An illustration of this process is provided in Figure 6.3. A normal vector for each point $\mathbf{P}_{\mathbf{x}}$ is precomputed via RANSAC [FB81] plane fitting in a local neighborhood of points, and is made to point towards the camera center.



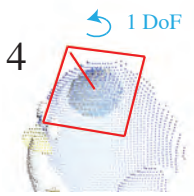
(a) Randomly pick a point $\mathbf{P}_{\mathbf{x}'}$ in the destination view within a search radius v of $\mathbf{P}_{\mathbf{x}}$.



(b) Translate accordingly.



(c) Rotate the source normal vector onto the destination normal vector.



(d) Randomly rotate around the destination normal.

Figure 6.3: Semi-random initialization of a 6 DoF rigid body motion g at a pixel $\mathbf{x}$, with the objective of restricting initialization to plausible states (most importantly meaning never initializing to empty space). (a) Randomly pick a point $\mathbf{P}_{\mathbf{x}'}$ in the destination view within a search radius v of the point $\mathbf{P}_{\mathbf{x}}$ encoded at $\mathbf{x}$ in the source view. (b) Obtain the 3 DoF translation from $\mathbf{P}_{\mathbf{x}'} - \mathbf{P}_{\mathbf{x}}$. (c) Obtain 2 DoF of the 3 DoF rotation from the rotation between the local normal vectors. (d) Obtain the remaining 1 DoF of the 3 DoF rotation by random rotation around the normal.

Spatial Propagation. Given a traversal from upper left to lower right, we consider at $\mathbf{x}$ the rigid body motions $g_{\mathbf{x}_\mathrm{n}}, g_{\mathbf{x}_\mathrm{w}}$, where $\mathbf{x}_\mathrm{n} = (x, y-1)^\top, \mathbf{x}_\mathrm{w} = (x-1, y)^\top$, and adopt a motion if doing so yields equal or lesser matching cost. Analogously, for a traversal from lower right to upper left, we consider $g_{\mathbf{x}_\mathrm{s}}, g_{\mathbf{x}_\mathrm{e}}$, where $\mathbf{x}_\mathrm{s} = (x, y+1)^\top, \mathbf{x}_\mathrm{e} = (x+1, y)^\top$. As in Hornáček et al. [HRGR13], the geometric rationale behind spatial propagation in our 6 DoF setting can be understood by observing that if two objects are related by a rigid body motion, then any corresponding pair of 3D point patches is related (modulo noise or sampling) by precisely the same motion.

Refinement. We aim of to improve upon the assigned motion $g_\mathbf{x}$ by considering a gentle perturbation of $g_\mathbf{x}$ (cf. Section 2.3). We build our candidates as in the semi-random initialization step, but try different combinations of changing the translation by hopping to a neighboring point in the destination view (3 DoF), altering the rotation to reflect a different destination normal (2 DoF), or modifying the around-normal rotation (1 DoF). With each try, we adopt the resulting candidate motion if doing so gives equal or lesser matching cost, and progressively reduce the allowed range of deviation from the current best.

View Propagation. As a last step when visiting a pixel $\mathbf{x}$, we project $g_\mathbf{x}(\mathbf{P}_\mathbf{x})$ to the nearest integer pixel $\mathbf{x}'$ in the destination view, where we evaluate the inverse $g_\mathbf{x}^{-1}$ of $g_\mathbf{x}$ provided that $\mathbf{x}'$ is valid. We adopt $g_\mathbf{x}^{-1}$ at $\mathbf{x}'$ if doing so gives equal or lesser matching cost. Since we carry out our variant of PatchMatch in parallel in both directions while traversing pixels in opposite order, by the time a pixel is reached in one view the most recent match available from the other has already been propagated. In contrast, Bleyer et al. [BRR11] treat views sequentially, with the effect of carrying out their form of view propagation *after* a full iteration is completed.

6.2.3 Occlusion Handling and Regularization

The dense correspondence search algorithm from Section 6.2.2 is reasonable only over points visible in both views, and can be expected to fail in areas where occlusions arise. We introduce a 6 DoF consistency check in the aim of distinguishing good matches from bad. In a first occlusion handling step, we assign to each valid pixel $\mathbf{x}$ that failed the check for $g_\mathbf{x}$ the motion $g_{\mathbf{x}'}$, such that $\mathbf{P}_{\mathbf{x}'}$ is the nearest neighbor point to $\mathbf{P}_\mathbf{x}$ corresponding to the subset of pixels that passed, drawn from the same view. Next, we carry out regularization of the motion field, which we reduce to a labeling problem optimized using α-expansion [BVZ01] over unary and pairwise potentials. Recognizing that over pixels that failed the consistency check, our unary potentials cannot rely on any criterion derived from our matching cost, we introduce a silhouette check to promote at least patchwise edge alignment from the viewpoint of the camera. We introduce an intuitive local rigidity prior as our pairwise potential. In contrast to Section 6.2.2, we carry out all steps in this section sequentially: first for the first view, then for the second.

Consistency Check. For a motion g under consideration at a pixel $\mathbf{x}$—by analogy to the left-right consistency check over disparity in stereo (e.g., Bleyer et al. [BRR11])—we project $g(\mathbf{P}_\mathbf{x})$ to the nearest integer pixel $\mathbf{x}'$ in the destination view and fail our check if $\mathbf{x}'$ is not valid or if

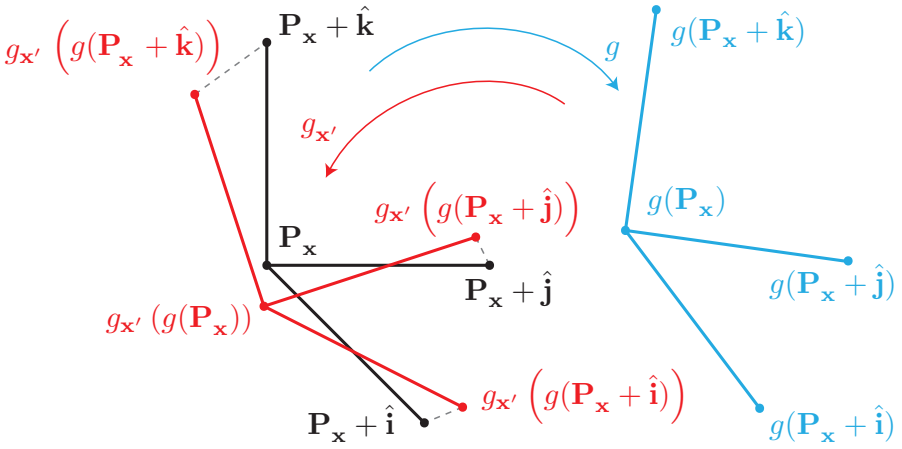


Figure 6.4: 6 DoF consistency check for g at $\mathbf{x}$. We project $g(\mathbf{P}_\mathbf{x})$ to the nearest integer pixel $\mathbf{x}'$ in the destination view, where we obtain the assigned motion $g_{\mathbf{x}'}$. We transform a unit tripod centered on the point $\mathbf{P}_\mathbf{x}$ (black) forward by g (blue) and then backward by $g_{\mathbf{x}'}$ (red). The check fails if any one of the distances in world space indicated by the three dashed lines exceeds a threshold.

the distance in image space between $\mathbf{x}$ and the projection of $g_{\mathbf{x}'}(g(\mathbf{P}_\mathbf{x}))$ back into the source view exceeds 1 pixel. However, such a check is by itself unsatisfactory: the distance in world space corresponding to a pixel displacement in image space grows as depth increases, and such a check ignores the rotational components of $g, g_{\mathbf{x}'}$. Accordingly, we additionally fail the check if $\exists \mathbf{v} \in \{\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}\}$ such that

$$\left\|(\mathbf{P}_\mathbf{x}+\mathbf{v}) - g_{\mathbf{x}'}\big(g(\mathbf{P}_\mathbf{x}+\mathbf{v})\big)\right\|_2 > \delta, \tag{6.5}$$

where $\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}$ are the standard (unit) basis vectors for $\mathbb{R}^3$. We set δ to Z_{med}/f (cf. Section 6.2.1), where Z_{med} is the median depth across both views, however this does not preclude alternative schemes for choosing a threshold. An illustration of the geometry of (6.5) is provided in Figure 6.4. Finally, we fail the check if $|\mathcal{S}_\mathbf{x}| < n$ or $|\mathcal{S}_{\mathbf{x}'}| < n$ in order to ensure a minimum matching support.

Initial Labeling. Each valid pixel $\mathbf{x}$ that passed the consistency check for $g_\mathbf{x}$ is assigned its own unique label l. For each valid pixel $\mathbf{x}$ that failed the consistency check, we instead assign to $\mathbf{x}$ the label assigned to the pixel $\mathbf{x}'$ such that $\mathbf{P}_{\mathbf{x}'} = \text{NN}_{\tilde{\mathcal{P}}}(\mathbf{P}_\mathbf{x})$, where $\tilde{\mathcal{P}}$ denotes the set of points corresponding to the set $\tilde{\mathcal{X}}$ of valid pixels that *passed* the consistency check for the view under consideration.

Regularization. We formulate our regularization as an energy minimization problem, taking the form:

$$E(\mathbf{l}) = \sum_{\mathbf{x}\in\mathcal{X}} D_\mathbf{x}(l_\mathbf{x}) + \sum_{\{\mathbf{x},\mathbf{x}'\}\in\mathcal{N}} V_{\{\mathbf{x},\mathbf{x}'\}}(l_\mathbf{x}, l_{\mathbf{x}'}), \tag{6.6}$$

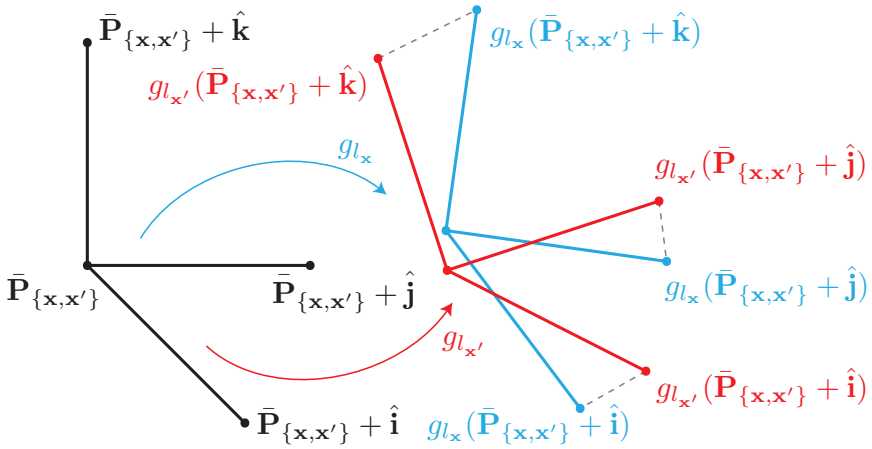


Figure 6.5: 6 DoF local rigidity prior. Given a pixel $\mathbf{x}$ and a neighboring pixel $\mathbf{x}'$, $\{\mathbf{x}, \mathbf{x}'\} \in \mathcal{N}$, and the rigid body motions $g_{l_{\mathbf{x}}}, g_{l_{\mathbf{x}'}}$ corresponding to the labels $l_{\mathbf{x}}, l_{\mathbf{x}'}$, the prior returns the sum of squares of the distances in world space indicated by the three dashed lines.

where $\mathcal{X}$ denotes the set of pixels in the image and $\mathcal{N}$ the set of its 4-connected pixel neighbors, and where $D_{\mathbf{x}}(l_{\mathbf{x}})$ denotes the unary potential at $\mathbf{x}$ and $V_{\{\mathbf{x},\mathbf{x}'\}}(l_{\mathbf{x}}, l_{\mathbf{x}'})$ the pairwise potential between $\mathbf{x}, \mathbf{x}'$. For pixels $\mathbf{x}$ that are not valid, we set $D_{\mathbf{x}}(l_{\mathbf{x}}) = 0$. Otherwise, if the consistency check was *passed* at $\mathbf{x}$ for the motion $g_{\mathbf{x}}$ assigned in our dense matching stage, $D_{\mathbf{x}}(l_{\mathbf{x}})$ takes the form $D^{\mathrm{p}}_{\mathbf{x}}(l_{\mathbf{x}})$:

$$D^{\mathrm{p}}_{\mathbf{x}}(l_{\mathbf{x}}) = \begin{cases} 0 & \text{if } g_{l_{\mathbf{x}}} \text{ passes consistency check at } \mathbf{x} \\ \rho & \text{otherwise} \end{cases}, \tag{6.7}$$

where ρ reflects the trust we lend to the assigned motions that satisfied the consistency check, recognizing that they were the strongest matches that PatchMatch succeeded in finding. In future work, we plan to address the merits of setting $D^{\mathrm{p}}_{\mathbf{x}}(l_{\mathbf{x}})$ to our matching cost when the check is passed; here, we opt instead for an approximation with the advantage of speed. If the consistency check was *failed* at $\mathbf{x}$ for $g_{\mathbf{x}}$, $D_{\mathbf{x}}(l_{\mathbf{x}})$ instead takes the form $D^{\mathrm{f}}_{\mathbf{x}}(l_{\mathbf{x}})$:

$$D^{\mathrm{f}}_{\mathbf{x}}(l_{\mathbf{x}}) = \begin{cases} 0 & \text{if } g_{l_{\mathbf{x}}} \text{ passes silhouette check at } \mathbf{x} \\ \kappa & \text{otherwise} \end{cases}, \tag{6.8}$$

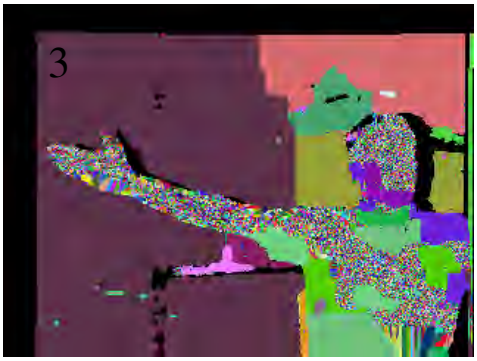
where κ weighs the influence of the silhouette check against the pairwise potentials. Our pairwise potentials reduce to 3D point SSD analogously to (6.3), but are computed over the axes of a unit tripod transformed according to the motions $g_{l_{\mathbf{x}}}, g_{l_{\mathbf{x}'}}$:

$$V_{\{\mathbf{x},\mathbf{x}'\}}(l_{\mathbf{x}}, l_{\mathbf{x}'}) = \beta_{\{\mathbf{x},\mathbf{x}'\}} \cdot \sum_{\mathbf{v} \in \{\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}\}} \left\| g_{l_{\mathbf{x}}}(\bar{\mathbf{P}}_{\{\mathbf{x},\mathbf{x}'\}} + \mathbf{v}) - g_{l_{\mathbf{x}'}}(\bar{\mathbf{P}}_{\{\mathbf{x},\mathbf{x}'\}} + \mathbf{v}) \right\|_2^2 . \tag{6.9}$$

We set $\beta_{\{\mathbf{x},\mathbf{x}'\}}$ to a fixed weight β if both $\mathbf{x}, \mathbf{x}'$ are valid and $\|\mathbf{P}_{\mathbf{x}} - \mathbf{P}_{\mathbf{x}'}\|_2 \leq r_{\mathrm{pix}} \cdot Z_{\mathrm{med}}/f$ (cf. Section 6.2.1), to 0 otherwise. This has the effect of regularizing only over pixels whose corresponding points could both be inliers of a sphere at depth Z_{med}. Our manner of proceeding does not preclude alternative schemes for choosing a maximum radius. We set $\bar{\mathbf{P}}_{\{\mathbf{x},\mathbf{x}'\}} = (\mathbf{P}_{\mathbf{x}} +$

(a) Consistency check (passing pixels in white).

(b) Initial labeling.



(c) Final output labeling.

Figure 6.6: Occlusion handling and regularization. (a) Having computed a dense correspondence field in both directions, determine which pixels pass the 6 DoF consistency check. (b) Each *passing* pixel $\mathbf{x}^{\mathrm{p}}$ is assigned its own unique label and each *failing* pixel $\mathbf{x}^{\mathrm{f}}$ is assigned that of the passing pixel whose point is closest in 3D to $\mathbf{P}_{\mathbf{x}^{\mathrm{f}}}$. (c) A final output labeling is obtained via α-expansion over 25 labels chosen at random from among the labels in the initial labeling, regularized with respect to our 6 DoF local rigidity prior (optimized by means of QPBO [LRRB10]).

$\mathbf{P}_{\mathbf{x}'})/2$, noting that the energy formulation in (6.6) implies that $V_{\{\mathbf{x},\mathbf{x}'\}} = V_{\{\mathbf{x}',\mathbf{x}\}}$, and that the local effect of the motions $g_{l_\mathbf{x}}, g_{l_{\mathbf{x}'}}$ on $\bar{\mathbf{P}}_{\{\mathbf{x},\mathbf{x}'\}}$ can be expected to approximate their effect on $\mathbf{P}_\mathbf{x}, \mathbf{P}_{\mathbf{x}'}$ when the two points are spatial neighbors in 3D belonging to the same object, which is precisely the target scenario. We illustrate the geometry of our local rigidity prior in Figure 6.5. We minimize our energy via the α-expansion algorithm [BVZ01], using QPBO [LRRB10] to compute the expansions. Labels l are drawn at random (without replacement) from the set of pixels for which the consistency check was satisfied in the initial labeling.

6.3 Evaluation

There exists at present no benchmark tailored to evaluating RGB-D scene flow. In Section 6.3.1, we accordingly give a quantitative evaluation following the example of Huguet and Devernay [HD07] by using the color images and ground truth disparity maps available with frames 2 and 6 of the Middlebury cones, teddy, and venus (cf. Scharstein and Szeliski [SS03]) stereo data sets, respectively, to compare against the known ground truth motion. In Section 6.3.2, we present qualitative results for the Middlebury data, and for challenging synthetic and real-world (Kinect) data sets.

6.3.1 Quantitative Evaluation

The 3D scene motion for the static cones, teddy, and venus data sets is due entirely to the motion of the camera and is purely translational in the X-direction of the camera coordinate frame, in the magnitude of the baseline relating the stereo pair (cf. Section 2.5). While that motion in 3D is simple, the matching problem is nevertheless confounded by occlusions and geometry of varying complexity. Let the vector $\big(u_\mathbf{x}, v_\mathbf{x}\big)^\top \in \mathbb{R}^2$ denote the 2D flow vector corresponding to the pixel $\mathbf{x}$ obtained by projecting the point $g_\mathbf{x}(\mathbf{P}_\mathbf{x})$ to image space, so that

$$\big(u_\mathbf{x}, v_\mathbf{x}\big)^\top = \pi\big(g_\mathbf{x}(\mathbf{P}_\mathbf{x})\big) - \mathbf{x}. \tag{6.10}$$

Similarly, the ground truth 2D flow vector $\big(u_\mathbf{x}^{\mathrm{GT}}, v_\mathbf{x}^{\mathrm{GT}}\big)^\top \in \mathbb{R}^2$ corresponding to the pixel $\mathbf{x}$ is obtained as in (6.10), but having replaced the rigid body motion $g_\mathbf{x}$ recovered by our algorithm with the ground truth motion $g^{\mathrm{GT}} \in SE(3)$ that all points in the left view undergo, expressed in matrix form—following (2.10)—as

$$\mathrm{T}_{\mathrm{E}}^{\mathrm{GT}} = \begin{bmatrix} 1 & 0 & 0 & -B \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6.11}$$

where B denotes the chosen baseline. Note that this gives the apparent 3D motion of all points encoded in the left view, having interpreted the pose of the right camera as identical with that of the left. We compare our results with respect to ground truth motion against Huguet and Devernay [HD07] and Basha et al. [BMK13]—who use only RGB images as input—in end point error (RMS-OF) and average angular error (AAE) over the 2D optical flow vectors obtained by projecting the output 3D displacements to image space. Following the example of

Hadfield and Bowden [HB13],[1] we additionally compute disparity change error (RMS-Vz) for the RGB-D scene flow techniques, namely for Hadfield and Bowden [HB13], Quiroga et al. [QDC13], and our method. All three metrics—whose respective definitions we defer until after this paragraph—are computed for the respective left view over the set Ω of valid pixels, which is to say over only those pixels for which a non-zero disparity value is encoded in the corresponding ground truth disparity map. For reference, we compute numbers for the RGB optical flow techniques of Brox and Malik [BM11] and Xu et al. [XJM12]. Our results placed our method as the top performer among scene flow algorithms considered in our quantitative evaluation. Numbers and additional explanation are provided in Table 6.1.

End Point Error (RMS-OF). The RMS-OF (more commonly referred to as EPE outside of the literature on scene flow) is simply the RMSE with respect to the Euclidean distance, over the pixels $\mathbf{x} \in \Omega$, between the recovered 2D flow $(u_\mathbf{x}, v_\mathbf{x})^\top$ and the ground truth 2D flow $(u_\mathbf{x}^{\mathrm{GT}}, v_\mathbf{x}^{\mathrm{GT}})^\top$:

$$\text{RMS-OF} = \sqrt{\frac{1}{|\Omega|} \sum_{\mathbf{x} \in \Omega} \left\| \left(u_\mathbf{x}, v_\mathbf{x}\right)^\top - \left(u_\mathbf{x}^{\mathrm{GT}}, v_\mathbf{x}^{\mathrm{GT}}\right)^\top \right\|_2^2}. \tag{6.12}$$

Disparity Change Error (RMS-Vz). Let $Z : \mathbb{R}^3 \to \mathbb{R}$ denote the function that returns the depth Z of a point $(X, Y, Z)^\top$. The disparity change error is given by the RMSE with respect to the absolute difference, over the pixels $\mathbf{x} \in \Omega$, between the disparity corresponding to $g_\mathbf{x}(\mathbf{P}_\mathbf{x})$ and that corresponding to $g^{\mathrm{GT}}(\mathbf{P}_\mathbf{x})$:

$$\begin{aligned} \text{RMS-Vz} &= \sqrt{\frac{1}{|\Omega|} \sum_{\mathbf{x} \in \Omega} \left| \frac{Bf}{Z\left(g_\mathbf{x}(\mathbf{P}_\mathbf{x})\right)} - \frac{Bf}{Z\left(g^{\mathrm{GT}}(\mathbf{P}_\mathbf{x})\right)} \right|} \\ &= \sqrt{\frac{1}{|\Omega|} \sum_{\mathbf{x} \in \Omega} \left| \frac{Bf}{Z\left(g_\mathbf{x}(\mathbf{P}_\mathbf{x})\right)} - \frac{Bf}{Z_\mathbf{x}} \right|}. \end{aligned} \tag{6.13}$$

Average Angular Error (AAE). We compute the AAE in precisely the same manner as in Huguet and Devernay [HD07], which is to say according to its form as provided in their publicly available[2] source code:

$$\text{AAE} = \frac{1}{|\Omega|} \cdot \frac{180}{\pi} \sum_{\mathbf{x} \in \Omega} \left| \arctan\left(\frac{u_\mathbf{x} \cdot v_\mathbf{x}^{\mathrm{GT}} - u_\mathbf{x}^{\mathrm{GT}} \cdot v_\mathbf{x}}{u_\mathbf{x} \cdot u_\mathbf{x}^{\mathrm{GT}} + v_\mathbf{x}^{\mathrm{GT}} \cdot v_\mathbf{x}} \right) \right|. \tag{6.14}$$

	B	HD	HB	Q	Ours	BM	X
Cones							
RMS-OF	0.58	1.10	1.24	0.57	**0.54**	2.83	1.66
RMS-Vz	N/A	N/A	0.06	0.05	**0.02**	1.75†	1.15†
AAE	**0.39**	0.69	1.01	0.42	0.52	0.39	0.21
Teddy							
RMS-OF	0.57	1.25	0.83	0.69	**0.35**	3.20	1.7
RMS-Vz	N/A	N/A	0.03	0.04	**0.01**	0.47†	0.5†
AAE	1.01	0.51	0.83	0.71	**0.15**	0.39	0.28
Venus							
RMS-OF	**0.16**	0.31	0.36	0.31	0.26	0.72	0.3
RMS-Vz	N/A	N/A	0.02	**0.00**	0.02	0.14†	0.22†
AAE	1.58	0.98	1.03	1.26	**0.53**	1.28	1.43

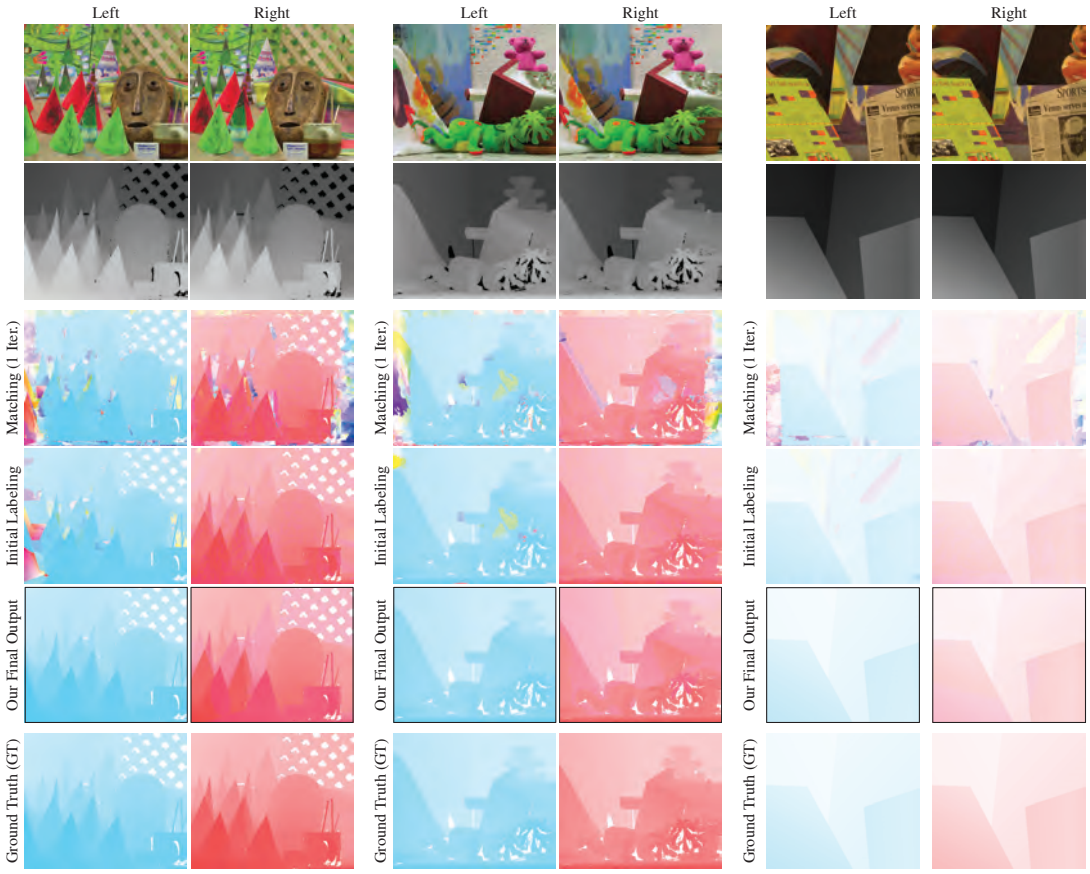
Table 6.1: Quantitative evaluation on RMS-OF (end point error), RMS-Vz (disparity change error), and AAE (average angular error). The topmost two methods are RGB optical flow algorithms; the next two are RGB scene flow algorithms that compute 3D translational flow and depth jointly. The remaining three are RGB-D scene flow techniques. Results were computed over all valid pixels, here meaning that GT disparity is nonzero and the pixel is marked as unoccluded in the Middlebury GT occlusion map. † indicates that RMS-Vz was computed by estimating 3D translational flow by interpolating depth encoded at the start and end points given its 2D flow vector. Accordingly, for the two optical flow techniques, we also ignored pixels at which the recovered 2D flow pointed to pixels with GT disparity of 0, since no end point depth could be interpolated. For Hadfield and Bowden, we additionally deemed pixels for which no flow was recovered as invalid. Cell colors indicate ranking among the five methods, from best to worst: green, light green, yellow, orange, red. Gray cells are shown for comparison but are not included in the ranking.

6.3.2 Qualitative Evaluation

We visualize the recovered correspondence fields by projecting the 3D displacements to 2D flow vectors, and coloring those vectors in the conventional manner (cf. Figure 4.2). In Figure 6.7, we show our intermediate and final results, contrasted with ground truth 2D flow colorings. In Figure 6.8, we show analogous results for large displacement Kinect data sets of varying complexity, and compare against the results of the RGB-D scene flow techniques of Hadfield and Bowden [HB13], Herbst et al. [HRF13], and Quiroga et al. [QDC13]. In Figure 6.9 we visualize our results—as in Figure 1.1—by flowing all 3D points in both frames for our final results on the data set in Figure 6.9 to intermediate points in time, demonstrating the visual credibility of our recovered motions. Finally, in Figure 6.10 we provide our results for four pairs

[1]Note that we borrowed their naming of end point error and disparity change error as RMS-OF and RMS-Vz, respectively. More commonly, end point error is referred to as EPE. In Chapter 7, we refer to end point error as EPE, which is the usual practice in the optical flow literature.

[2]Source code available at `http://devernay.free.fr/vision/varsceneflow/`.



(a) Cones. (b) Teddy. (c) Venus.

Figure 6.7: Results on Middlebury cones, teddy, and venus. 2D optical flow coloring by projection of 3D displacements to image space, for the left (blue) and right (red) views. Numbers in Table 6.1 correspond to the left view.

of the penguins data set, showing outstanding flow results on complex non-planar geometry at large displacements.

6.3.3 Algorithm Parameters

Radius r_{pix} was set to 15 for all data sets. For our variant of PatchMatch, j was set to 3, k to 5. Number of iterations i was 2 for Kinect data, 1 otherwise. Search radius v was set to comfortably exceed the maximum displacement across the data sets. Recognizing that the relative magnitudes of the point and gradient similarity terms $E_{\mathbf{x}}^{\text{pt}}, E_{\mathbf{x}}^{\text{gr}}$ are a function of the distances between neighboring points, the weight α between $E_{\mathbf{x}}^{\text{pt}}$ and $E_{\mathbf{x}}^{\text{gr}}$ in (6.4) was set adaptively according to $0.001 \cdot (Z_{\text{med}}/f)^2$ (cf. Section 6.2.1), where Z_{med} is again the median depth across both views. We set γ for adaptive support weighting to 10 like Bleyer et al. [BRR11]. For regularization, κ was fixed to 1, and ρ and β were set to 10000 and 5 for Kinect data (giving a

(a) Couple. (b) Dance. (c) Wave.

Figure 6.8: Results on Kinect couple, dance, and wave.

(a) Hadfield and Bowden [HB13].

(b) Herbst et al. [HRF13]

(c) Quiroga et al. [QDC13]

(d) Our final output.

Figure 6.9: 3D flow visualization at a novel viewpoint of results on the Kinect wave data set. All points encoded in the *two* input RGB-D frames are flowed to a common point in time, in the same manner as in Figure 1.1.

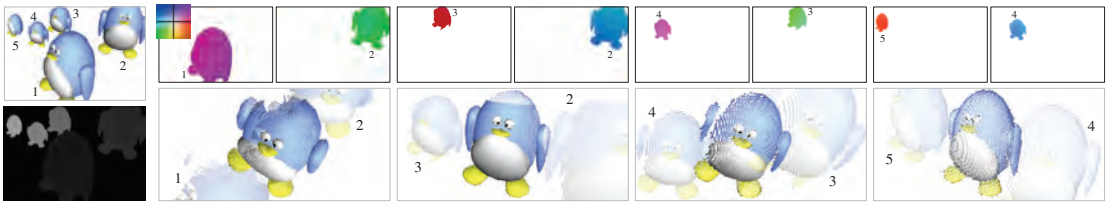


Figure 6.10: Large displacement synthetic penguins data set from Hornáček et al. [HRGR13] (featured in Chapter 6) for which assumptions of brightness constancy and local surface planarity (for discriminative patch size) fail pronouncedly. We run our algorithm on consecutive penguin pairs and visualize our results at a novel viewpoint by flowing all points, respectively, to an intermediate point in time. Original points shown with transparency for reference.

large degree of trust to the output of our PatchMatch variant, with gentle regularization), to 1 and 10000 otherwise (promoting a heavily regularized solution). Minimum sphere inlier count n was set to 10. Pixel masks for the silhouette check were dilated by 5 pixels for Kinect data (owing to poor edge quality in Kinect depth maps), 1 otherwise. We considered a total of 25 labels for α-expansion, independently for both views.

6.4 Discussion

The essence of PatchMatch is to grow (cf. Figure 1.5) and refine sparse correspondence seeds, where obtaining these seeds by *indiscriminate randomization* is shown in [BSFG09] to lead to attractive flow results in a 2 DoF translational setting. Whether this strategy scales to 6 DoF rigid body motions, however, is left an open question. As detailed in Section 6.2.2, the strategy we pursue to initialize our correspondence fields is to obtain the translational 3 DoF of a 6 DoF 3D rigid body motion by combining radius search with sparse feature matching, and the rotational 3 DoF by taking into account local surface normals. In Figures 6.11-6.20, we give 2D flow colorings for two iterations for each of the data sets presented in this chapter, comparing results given initialization (i) by radius search and sparse matching (i.e., in the manner of the published algorithm, as outlined in Section 6.2.2), (ii) by radius search alone, (iii) by sparse matches alone (at pixels where no sparse match is available, we initialize to a dummy state with infinite cost), and (iv) by indiscriminate randomization (translational 3 DoF by randomly choosing from all valid pixels in the destination view, with rotational 3 DoF chosen otherwise fully at random). Initialization by indiscriminate randomization clearly gives the slowest convergence, with the exception of the wave data set in Figure 6.11 and of the penguin data sets in Figures 6.17-6.20, for which the search space is restricted to a much smaller set of valid pixels, respectively, than is the case for Kinect and Middlebury. While radius queries may be conceptually attractive from a physical point of view and typically lead to faster convergence than by indiscriminate randomization, initializing only where sparse matches are available appears to give comparable results in most cases without the computational expense of densely computing radius queries using a kd-tree. The merits of combining radius search with sparse matching are most clearly visible in Figure 6.11, where combining the two gives faster convergence than for only radius search or sparse matching, respectively. Note in all cases the importance of the occlusion handling and regularization stage for obtaining the final results we present in Section 6.3, most notably for the penguin data sets in Figures 6.17-6.20, where on the whole the majority of pixels are occluded.

6.5 Conclusion

We presented a technique for densely computing continuous 6 DoF scene flow between a pair of consecutive RGB-D frames. Rather than rely on brightness constancy or local surface planarity as in most previous work in the literature on scene flow, we proceed instead to reason instead in terms of patches of 3D points identified as inliers of spheres, which we match with respect to 6 DoF 3D rigid body motions. We showed highly competitive results on the Middlebury cones, teddy, and venus data sets together with state of the art results on challenging real-world (Kinect) and synthetic scenes.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

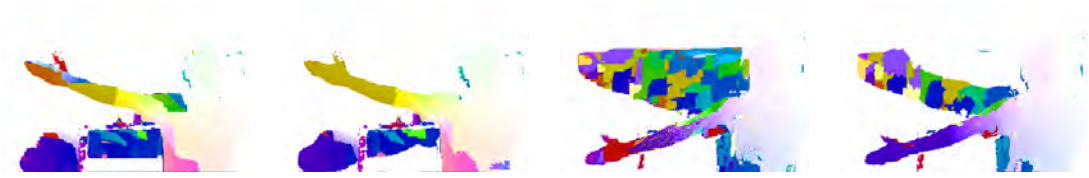
(a) Initialization by radius search and sparse matches (published algorithm).

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(b) Initialization by only radius search.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(c) Initialization by only sparse matches.



Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(d) Initialization by indiscriminate randomization.

Figure 6.11: Initialization schemes compared on wave.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

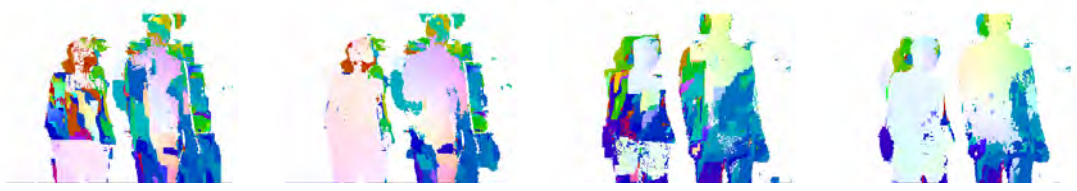
(a) Initialization by radius search and sparse matches (published algorithm).

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(b) Initialization by only radius search.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

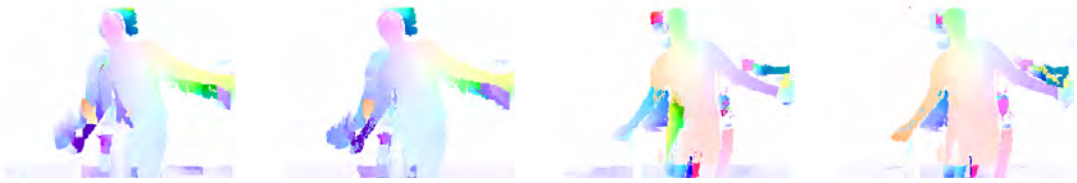
(c) Initialization by only sparse matches.



Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(d) Initialization by indiscriminate randomization.

Figure 6.12: Initialization schemes compared on couple.



Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(a) Initialization by radius search and sparse matches (published algorithm).

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(b) Initialization by only radius search.

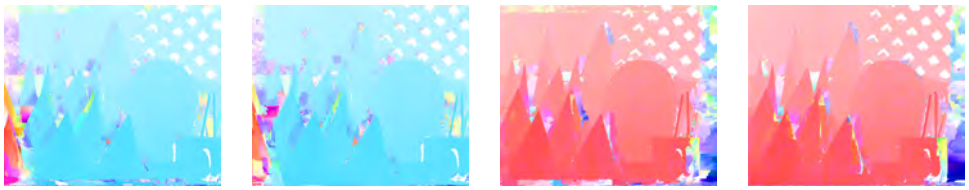
Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(c) Initialization by only sparse matches.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(d) Initialization by indiscriminate randomization.

Figure 6.13: Initialization schemes compared on dance.



Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

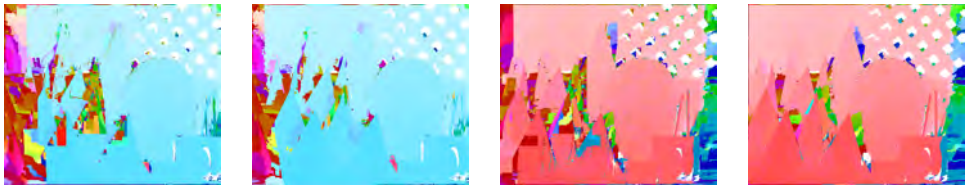
(a) Initialization by radius search and sparse matches (published algorithm).

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(b) Initialization by only radius search.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

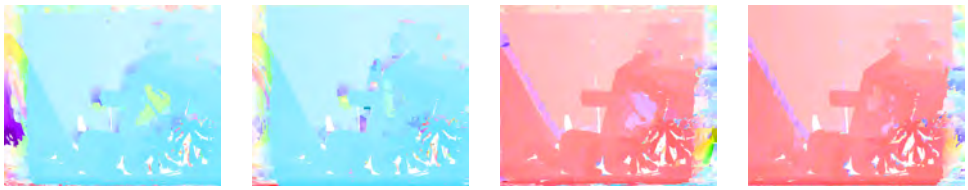
(c) Initialization by only sparse matches.



Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

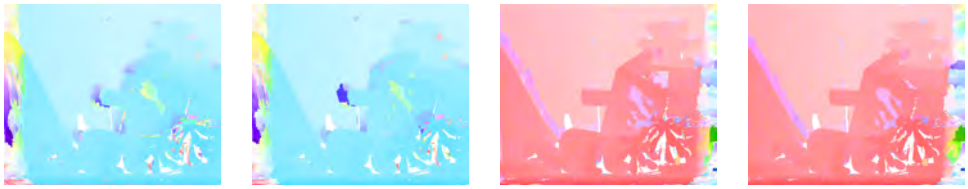
(d) Initialization by indiscriminate randomization.

Figure 6.14: Initialization schemes compared on cones.



Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(a) Initialization by radius search and sparse matches (published algorithm).

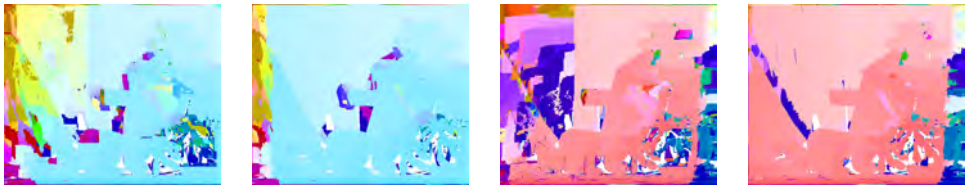


Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(b) Initialization by only radius search.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(c) Initialization by only sparse matches.



Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

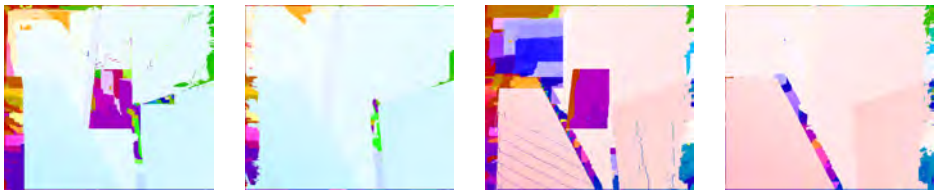
(d) Initialization by indiscriminate randomization.

Figure 6.15: Initialization schemes compared on teddy.

(a) Initialization by radius search and sparse matches (published algorithm).

(b) Initialization by only radius search.

(c) Initialization by only sparse matches.



(d) Initialization by indiscriminate randomization.

Figure 6.16: Initialization schemes compared on venus.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(a) Initialization by radius search and sparse matches (published algorithm).

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(b) Initialization by only radius search.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(c) Initialization by only sparse matches.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(d) Initialization by indiscriminate randomization.

Figure 6.17: Initialization schemes compared on penguins 1-2.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(a) Initialization by radius search and sparse matches (published algorithm).

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(b) Initialization by only radius search.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(c) Initialization by only sparse matches.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(d) Initialization by indiscriminate randomization.

Figure 6.18: Initialization schemes compared on penguins 2-3.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(a) Initialization by radius search and sparse matches (published algorithm).

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(b) Initialization by only radius search.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(c) Initialization by only sparse matches.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(d) Initialization by indiscriminate randomization.

Figure 6.19: Initialization schemes compared on penguins 3-4.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(a) Initialization by radius search and sparse matches (published algorithm).

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(b) Initialization by only radius search.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(c) Initialization by only sparse matches.

Frame 1, Iteration 1 Frame 1, Iteration 2 Frame 2, Iteration 1 Frame 2, Iteration 2

(d) Initialization by indiscriminate randomization.

Figure 6.20: Initialization schemes compared on penguins 4-5.

CHAPTER 7

Highly Overparameterized Optical Flow

2D motion in the image plane is ultimately a function of 3D motion in the scene. In this chapter, we propose to compute optical flow using what is ostensibly an extreme overparameterization: depth, surface normal, and frame-to-frame 3D rigid body motion at every pixel, giving a total of 9 DoF. The advantages of such an overparameterization are twofold: first, geometrically meaningful reasoning can be called upon in the optimization, reflecting possible 3D motion in the underlying scene; second, the 'fronto-parallel' assumption implicit in traditional pixel window-based matching is ameliorated because the parameterization determines a plane-induced homography at every pixel. Our main contribution is to show that optimization over this high-dimensional, continuous state space can be carried out using an adaptation of the recently introduced PatchMatch Belief Propagation (PMBP) energy minimization algorithm [BRFK12]. We begin in Section 7.1 by introducing our methodology and placing our algorithm in the context of earlier work. Next, we detail our algorithm in Section 7.2. In Section 7.3, we show that our flow fields compare favorably to the state of the art on a number of small- and large-displacement datasets. In Section 7.4 we attempt to emphasize the key elements of the algorithm with regard to performance, ending with concluding remarks in Section 7.5.

7.1 Introduction

One statement of the goal of optical flow computation is the recovery of a dense correspondence field between the *pixels* of a pair of images (cf. Section 4.2). Sun et al. [SRB10] argue that classical models such as Horn and Schunck [HS81] can achieve good performance when coupled to modern optimizers. They point out the key elements that contribute to quality of the solution, including image pre-processing, a coarse-to-fine (multiscale) scheme, bicubic interpolation, robust penalty functions, and median filtering, which they integrate into a new energy formulation. Xu et al. [XJM12] observe that while a large number of optical flow techniques

use a multiscale approach, pyramidal schemes can lead to problems in accurately detecting the large motion of fine structures. They propose to combine sparse feature detection with a classic pyramidal scheme to overcome this difficulty. Additionally, they selectively combine color and gradient in the similarity measure on a per pixel basis in the aim of improving robustness. Similarly, Brox and Malik [BM11] integrate SIFT feature matching [Low04] into a variational framework in order to guide the recovered flow towards large displacements.

Another way to define a correspondence is in terms of the similarity of *pixel windows* centered on each image pixel. Immediately, the size of the window becomes an important algorithm parameter: a small window offers little robustness to intensity variations such as those caused by lighting change, differences in camera response, or image noise; a large window can overcome these difficulties but most published work suffers from what we loosely term the 'fronto-parallel' (FP) assumption, according to which each pixel in the window is assumed to undergo the same 2D translation. The robustness of small-window models can be improved by means of priors over motion at neighboring pixels, but first-order priors themselves typically imply the fronto-parallel limitation, second-order priors are expensive to optimize for general energies [WTRF09] although efficient schemes exist for some cases [TPCB08]. Beyond the second order, higher-order priors impose quite severe limitations on the state spaces they can model. In the case of optical flow, the state space is essentially continuous, and certainly any discretization must be very dense.

An alternative strategy to relax the FP assumption is to *overparameterize* the 2 DoF motion field (cf. Section 2.2). Previous work in optical flow has considered 3 DoF similarity transformations [BSGF10], 6 DoF affine transformations [NBK08], or 6 DoF linearized 3D motion models [NBK08]. In the case of stereo correspondence, the 1 DoF disparity field has been overparameterized in terms of a 3 DoF surface normal and depth field [BRFK12, BRR11, LZ06]. With such models, even first order priors can be expressive (e.g., piecewise constant surface normal is equivalent to piecewise constant depth derivatives rather than piecewise constant depth). However, effective optimization of such models has required linearization of brightness constancy [NBK08] or has suffered from local optimality [LZ06]. Recently, however, algorithms based on PatchMatch [BSFG09, BSGF10] have been applied to 3 DoF (depth+normal) stereo matching [BRFK12, BRR11, HKJK13] and 6 DoF (3D rigid body motion) RGB-D scene flow [HFR14] (described in Chapter 6), and it is to this class of algorithms that ours belongs.

In this work, our main contribution is to employ an overparameterization not previously applied to the computation of optical flow, assigning a 9 DoF plane-induced homography to each pixel. In addition to relaxing the FP assumption, such a model allows for geometrically meaningful reasoning to be integrated in the optimization, reflecting possible 3D motion in the underlying scene. Vogel et al. [VSR13] recover scene flow over consecutive calibrated stereo pairs by jointly computing a segmentation of a keyframe and assigning to each segment a 9 DoF plane-induced homography, optimized using QPBO [LRRB10] over a set of proposal homographies. For optical flow from a pair of images without strictly enforcing epipolar geometry, we show that PatchMatch Belief Propagation (PMBP) of Besse et al. [BRFK12] can be adapted to optimize the high-dimensional, non-convex optimization problem of assigning a 9 DoF plane-induced homography to each pixel and that the resulting flow fields compare favorably to the state of the art on a number of datasets. The model parameterizes, at each pixel, a 3D plane undergoing rigid

body motion, and can be specialized for piecewise rigid motion, or indeed for a single global rigid motion [VBW08, WPB+08].

7.2 Algorithm

Let (I_1, I_2) be an ordered pair of images depicting a static or moving scene at different points in time and/or from different points of view, and let (G_1, G_2) be the analogous gradient images, each consisting of a total of p pixels. For one of the two views $i \in \{1, 2\}$, let $\mathbf{x}_s = (x_s, y_s)^\top$ denote such a pixel, indexed by $s \in \{1, \ldots, p\}$. Let $\mathcal{N}_s$ denote the set of indices of the 4-connected neighbors of $\mathbf{x}_s$ and $\mathcal{W}_s$ the set of indices of pixels in the patch centered on $\mathbf{x}_s$. At every pixel $\mathbf{x}_s$, rather than seek a 2D flow vector, we shall aim to obtain a state vector θ_s that determines a plane-induced homography $H(\theta_s)$ to explain the motion of the pixels $\mathbf{x}_t, t \in \mathcal{W}_s$. We solve for the flow field by minimizing an energy defined over such state vectors, comprising *data terms* E_s and *smoothness terms* $E_{s,t}$:

$$E(\theta_1, \ldots, \theta_p) = \sum_{s=1}^{p} E_s(\theta_s) + \sum_{\{s,t\} \in \mathcal{E}} E_{s,t}(\theta_s, \theta_t), \tag{7.1}$$

where $\mathcal{E}$ denotes the set $\bigcup_{s \in \{1,\ldots,p\}} \big\{\{s, t\} \mid t \in \mathcal{N}_s\big\}$ of all 4-connected neighbors. In the remainder of this section, we proceed first to introduce the parameterization and the data term, and follow by detailing the smoothness term.

7.2.1 Model and Data Term

Ignoring for the moment the details of the parameterization, let $I_i(\mathbf{x}_s)$ and $G_i(\mathbf{x}_s)$ denote the color and gradient, respectively, at pixel $\mathbf{x}_s$ in view i, $i \in \{1, 2\}$. Given a pixel $\mathbf{x}$ in floating point coordinates, we obtain $I_i(\mathbf{x}), G_i(\mathbf{x})$ by interpolation. Let $\tilde{\mathbf{x}} = (x_1, x_2, x_3)^\top \in \mathbb{P}^2$ denote a pixel in projective 2-space, and $\epsilon(\tilde{\mathbf{x}}) = (x_1/x_3, x_2/x_3)^\top \in \mathbb{R}^2$ its analogue in Euclidean 2-space. Let H_s be shorthand for $H(\theta_s)$ and $\mathtt{H}_s$ denote the 3×3 matrix form of H_s, and let $H_s * \mathbf{x} = \epsilon(\mathtt{H}_s(\mathbf{x}^\top, 1)^\top) \in \mathbb{R}^2$ be the pixel obtained by applying the homography H_s to the pixel $\mathbf{x}$. This lends itself to a data term that, at pixel $\mathbf{x}_s$ in view i—which we shall call the *source* view—sums over the pixels of the patch $\mathcal{W}_s$:

$$E_s(\theta_s) = \frac{1}{|\mathcal{W}_s|} \sum_{t \in \mathcal{W}_s} w_{s,t} \cdot \Big((1-\alpha) \big\| I_i(\mathbf{x}_t) - I_j(H_s * \mathbf{x}_t) \big\| + \alpha \big\| G_i(\mathbf{x}_t) - G_j(H_s * \mathbf{x}_t) \big\| \Big), \tag{7.2}$$

where $j \in \{1, 2\}, i \neq j$, indexes the *destination* view, $w_{s,t} = \exp(-\|I_i(\mathbf{x}_s) - I_i(\mathbf{x}_t)\|/\gamma)$ implements a form of adaptive support weighting (cf. Yoon and Kweon [YK05]), and $\alpha \in [0, 1]$ controls the relative influence of the color and gradient components of the data term. The data term is scaled by $1/|\mathcal{W}_s|$ in the aim of rendering the strength of the smoothness term in (7.1) invariant to the patch size.

Casting the standard FP model in these terms, one could define $\theta^{\text{FP}} = (\delta_x, \delta_y)$ in terms of the 2D flow vector $(\delta_x, \delta_y)^\top$ at pixel $\mathbf{x}_s$, and express the homography $H(\theta^{\text{FP}})$ in matrix form as

$$\mathtt{H}(\theta^{\text{FP}}) = \begin{bmatrix} 1 & 0 & \delta_x \\ 0 & 1 & \delta_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{7.3}$$

We take the parameterization a step further, parameterizing not only a 3D plane at each pixel, but also a 3D rigid body motion transforming the points in that plane. Let $\mathbf{n}_s$ denote the unit surface normal of a plane in 3D and Z_s the depth of the point of intersection of that plane with the back-projection $\mathbf{p}_s = \mathtt{K}^{-1}(\mathbf{x}_s^\top, 1)^\top$ of the pixel $\mathbf{x}_s$, where $\mathtt{K}$ is the 3×3 camera calibration matrix (cf. Section 2.4). The point of intersection is then given by $Z_s\mathbf{p}_s \in \mathbb{R}^3$. Let $g_s = (\mathtt{R}_s, \mathbf{t}_s)$ denote a 6 DoF rigid body motion in 3D, where $\mathtt{R}_s \in SO(3)$ and $\mathbf{t}_s \in \mathbb{R}^3$. We write our overparameterized motion model $H(\theta_s)$ in matrix form as

$$\mathtt{H}(\theta_s) = \mathtt{K}\left(\mathtt{R}_s + \frac{1}{Z_s\mathbf{n}_s^\top\mathbf{p}_s}\mathbf{t}_s\mathbf{n}_s^\top\right)\mathtt{K}^{-1}, \tag{7.4}$$

where $\theta_s = (Z_s, \mathbf{n}_s, \mathtt{R}_s, \mathbf{t}_s)$, for a total of 9 DoF. Setting $Z_s\mathbf{n}_s^\top\mathbf{p}_s = d_s$, we obtain the familiar *homography induced by the plane* (cf. Section 2.6), with plane $\boldsymbol{\pi}_s = (\mathbf{n}_s^\top, -d_s)^\top \in \mathbb{P}^3$. For static scenes undergoing only camera motion, $(\mathtt{R}_s, \mathbf{t}_s)$ determines the pose of the camera of the destination view, expressed in the camera coordinate frame of the source view (cf. Figure 2.7). More generally, such a homography lends itself to interpretation as $(\mathtt{R}_s, \mathbf{t}_s)$ applied to the point obtained by intersecting $\boldsymbol{\pi}_s$ with a pixel back-projection in the source view, and projecting the resulting point into the destination view (cf. Figure 7.1), with the pose of both cameras kept identical. On this interpretation, we may reason about scenes undergoing pure camera motion, pure object motion, or joint camera and object motion in the same conceptual framework, as illustrated in Figure 2.10.

Validity Check. Recognizing that a plane whose normal does not point toward the camera is meaningless, and one that is close to orthogonal to the look direction is of no practical use in obtaining matches, we additionally wish to flatly reject such *invalid* states without taking the time to compute the data term in (7.2). Accordingly, a homography $H(\theta_s)$ is deemed invalid if the source and destination normals $\mathbf{n}_s, \mathtt{R}_s\mathbf{n}_s$ do not both face toward the camera, if it encodes negative source or destination depth, or if $H(\theta_s) * \mathbf{x}_s$ lies outside the destination image. Homographies deemed invalid are assigned an infinite matching cost.

7.2.2 Smoothness Term

The role of the smoothness term $E_{s,t}$ is to encourage the action of the homographies parametrized by states θ_s, θ_t assigned to neighboring pixels to be similar. One approach to defining a such a smoothness term could be to define distances between the geometric quantities encoded in the state vectors, specifically depth, normal, and rigid body motion. Reasoning directly in terms of the similarity of the parameters of the model would introduce a number of algorithm tuning parameters, as the natural scales of variation of each parameter type are not commensurate. While

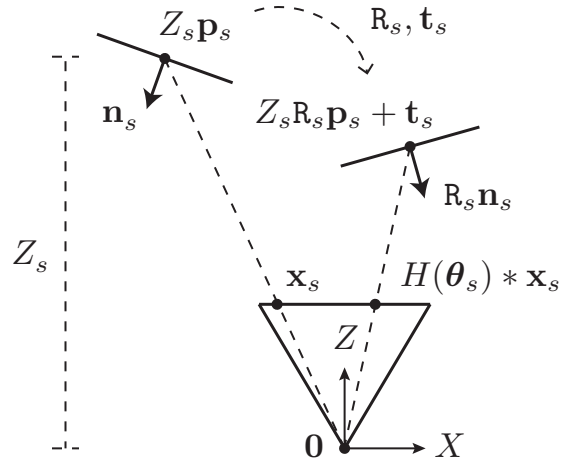


Figure 7.1: Depiction of the geometric interpretation of a homography $H(\theta_s)$, $\theta_s = (Z_s, \mathbf{n}_s, \mathtt{R}_s, \mathbf{t}_s)$, assigned to a pixel $\mathbf{x}_s$ as a 3D plane with unit normal $\mathbf{n}_s$ intersecting the back-projection of the pixel $\mathbf{x}_s$ at depth Z_s and undergoing the rigid body motion $(\mathtt{R}_s, \mathbf{t}_s)$. Applying $H(\theta_s)$ to an arbitrary pixel $\mathbf{x}_t$ has the effect of intersecting the back-projection of $\mathbf{x}_t$ with this plane to obtain a point $\mathbf{P}_t \in \mathbb{R}^3$, transforming $\mathbf{P}_t$ by the motion $(\mathtt{R}_s, \mathbf{t}_s)$ to obtain $\mathbf{P}'_t = \mathtt{R}_s\mathbf{P}_t + \mathbf{t}_s$, and finally projecting $\mathbf{P}'_t$ back to image space.

these could be determined using a training set, a large training set may be required. We instead focus our attention directly on the smoothness of the resulting 2D flow—since it is a smooth 2D flow field that we aim to obtain as output of our algorithm—and introduce a considerably more intuitive smoothness term:

$$E_{s,t}(\theta_s, \theta_t) = \lambda \cdot \min\left(\kappa, \left\|H_s * \mathbf{x}_s - H_t * \mathbf{x}_s\right\| + \left\|H_t * \mathbf{x}_t - H_s * \mathbf{x}_t\right\|\right), \tag{7.5}$$

where $\lambda \geq 0$ is a smoothness weight and $\kappa > 0$ is a truncation constant intended to add robustness to large state discontinuities, particularly with object boundaries in mind. This smoothness term has only two parameters (λ and κ) and is in units of pixels.

7.2.3 Energy Minimization

While it may be easy to formulate a realistic energy function, such a function is of little practical use if it cannot be minimized in reasonable time. Minimization of the energy in (7.1) is a non-convex optimization problem over a high-dimensional, continuous state space. The recently introduced PatchMatch Belief Propagation (PMBP) algorithm of Besse et al. [BRFK12] provides an avenue to optimizing over such a state space by leveraging PatchMatch [BSFG09, BSGF10] for exploiting the underlying spatial coherence of the state space by sampling from pixel neighbors (spatial propagation), and belief propagation [YFW00] for the explicit promotion of smoothness.

We adapt PMBP in the aim of assigning to each pixel $\mathbf{x}_s$ an optimal state θ_s, mapping the projectively warped patch centered on $\mathbf{x}_s$ in the source view to its analogue in the destination

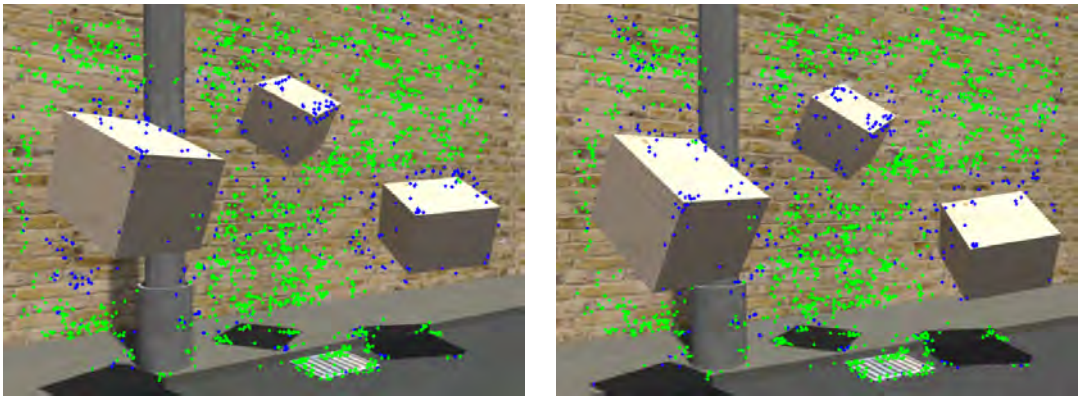
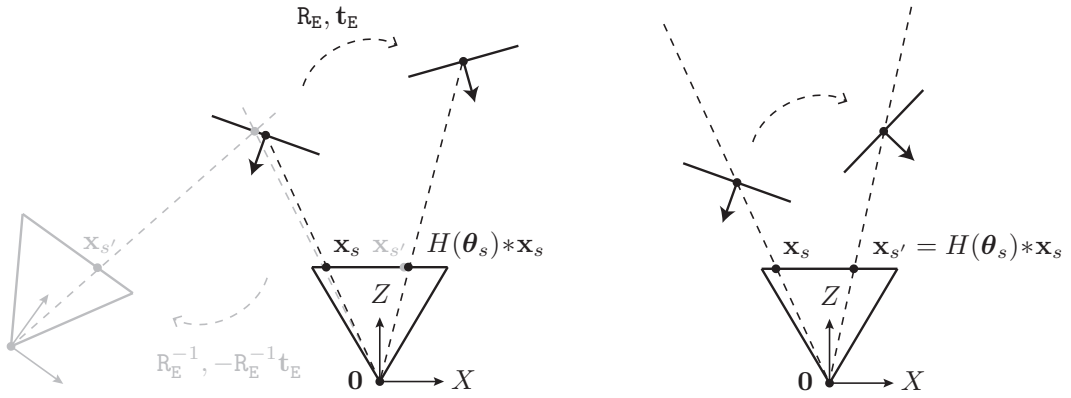


Figure 7.2: Inlier (green) and non-inlier (blue) ASIFT matches, with respect to the dominant 3D rigid body motion $(\mathtt{R}_\mathtt{E}, \mathbf{t}_\mathtt{E})$ recovered using the 5 point algorithm with RANSAC. Note that ordinary SIFT matching fails to capture the matches on the cubes (not shown).

view. Since our parameterization has a geometric interpretation in terms of rigidly moving planes in 3D, we are able to tailor PMBP to make moves that are sensible in 3D. We begin by (i) initializing the state space in a semi-random manner, making use of knowledge about the scene that we are able to recover from the input image pair (*initialization*). Next, for i iterations, we traverse each pixel $\mathbf{x}_s$ in scanline order, first (ii) attempting to propagate the states assigned to neighbors of $\mathbf{x}_s$ (*spatial propagation*) and then (iii) trying to refine the state vector (*random search*), in each case adopting a candidate state if doing so yields lower disbelief than the current assignment. We do this in both directions (view 1 to view 2, view 2 to view 1) *in parallel* and in opposite traversal orders, and as a last step when visiting $\mathbf{x}_s$ we additionally (iv) attempt to propagate the state at $\mathbf{x}_s$ from the source view to $H(\theta_s) * \mathbf{x}_s$ in the destination, rounded to the nearest integer pixel (*view propagation*); accordingly, by the time a pixel is reached in one view, the most recent match available from the other has already been considered.

Semi-Random Initialization. In order to promote convergence to correct local minima, we constrain our choice of initializing state vectors using knowledge we are able to recover from the input image pair. We estimate the dominant rigid body motion of the scene by feeding pairs of keypoint matches obtained using ASIFT[1] [MY09] to the 5 point algorithm [Nis04] with RANSAC [FB81], giving an essential matrix $\mathtt{E} = [\mathbf{t}_\mathtt{E}]_\times \mathtt{R}_\mathtt{E}$ that we subsequently decompose into a rigid body motion $(\mathtt{R}_\mathtt{E}, \mathbf{t}_\mathtt{E})$ (cf. Section 2.5 and Figure 7.2). One might consider iteratively recovering additional dominant rigid body motions by culling inlier matches and re-running the 5 point algorithm with RANSAC on the matches that remain, or consider alternative rigid motion segmentation techniques [DOIB12]. We triangulate the ASIFT matches that are inliers of the recovered dominant motion, giving seed points for which only the plane normal $\mathbf{n}_s$ remains a

[1]The publicly available ASIFT code carries out a form of epipolar filtering using the Moisan-Stival Optimized Random Sampling Algorithm (ORSA) [MS04b]. We remove this feature in order to obtain all matches recovered by the ASIFT matcher.



(a) Initialization from inlier matches. (b) Initialization from general matches.

Figure 7.3: Semi-random initialization. (a) Initialization from ASIFT match pairs $(\mathbf{x}_s, \mathbf{x}_{s'})$ that are inliers of a recovered dominant rigid body motion $(\mathtt{R}_\mathtt{E}, \mathbf{t}_\mathtt{E})$, with depth Z_s determined by triangulation and $\mathbf{n}_s$ as the only free parameter. (b) Initialization from general ASIFT match pairs $(\mathbf{x}_s, \mathbf{x}_{s'})$, constrained in that $\mathbf{x}_{s'} = H(\theta_s) * \mathbf{x}_s$; an alternative expression of this constraint is the requirement that $Z_s \mathtt{R}_s \mathbf{p}_s + \mathbf{t}_s$ project exactly to the pixel $\mathbf{x}_{s'}$.

free parameter (cf. Figure 7.3a). Since we wish to allow deviation from recovered dominant motions yet would like to leverage all of the available ASIFT matches, we additionally use the full set of ASIFT match pairs $(\mathbf{x}_s, \mathbf{x}_{s'})$ for seeding by estimating, for each pair, a tailored rigid body motion constrained by the requirement that $\mathbf{x}_{s'} = H(\theta_s) * \mathbf{x}_s$ (cf. Figure 7.3b), with depth Z_s in addition to normal $\mathbf{n}_s$ as free parameters. At pixels where more than one such seed is available, we choose one at random. For unseeded pixels, we set $(\mathtt{R}_s, \mathbf{t}_s)$ to one of the recovered dominant motions, with depth Z_s and normal $\mathbf{n}_s$ again free.

Spatial Propagation. In the usual manner of PatchMatch [BSFG09, BSGF10, BRFK12], we traverse the pixels of the source image in scanline order and consider, at the current pixel $\mathbf{x}_s$, the subset of states $\{\theta_t \mid t \in \mathcal{N}_s\}$ assigned to the 4-connected neighbors of $\mathbf{x}_s$ that have already been visited in the iteration, and adopt such a state θ_t if doing so gives lower disbelief than the current assignment. Note that owing to our parameterization, adopting the state $\theta_t = (Z_t, \mathbf{n}_t, \mathtt{R}_t, \mathbf{t}_t)$ at pixel $\mathbf{x}_s$ calls for recomputing the depth by intersecting the plane $\boldsymbol{\pi}_t$ with the back-projection of $\mathbf{x}_s$; the remaining components of the state vector θ_t are simply copied.

Random Search. We perturb, at random, either depth Z_s and normal $\mathbf{n}_s$ or the rigid body motion $(\mathtt{R}_s, \mathbf{t}_s)$ of the state vector θ_s currently assigned to the pixel $\mathbf{x}_s$. When $(\mathtt{R}_s, \mathbf{t}_s)$ is locked, we are effectively carrying out stereo matching. When $Z_s, \mathbf{n}_s$ are locked, we perturb the translational component of the motion with the effect of sampling within a 3D radius around $Z_s \mathtt{R}_s \mathbf{p}_s + \mathbf{t}_s$; perturbation of the rotational component serves in effect to change the normal of the transformed plane (cf. Figure 7.4 and Section 2.3). We carry out several such perturbations of the four components of the assigned state vector, reducing the search range with every try. We adopt a proposed perturbation if doing so gives lesser disbelief than the current assignment.

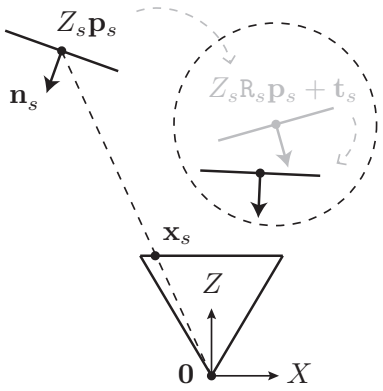


Figure 7.4: Refinement of the rigid body motion $(\mathtt{R}_s, \mathbf{t}_s)$ for plane parameters $Z_s, \mathbf{n}_s$ fixed. Perturbation of the translational component $\mathbf{t}_s$ is carried out with the effect of applying a translation to the current $\mathbf{P}'_s = Z_s\mathtt{R}_s\mathbf{p}_s + \mathbf{t}_s$ within a radius of $\mathbf{P}'_s$ in 3D (depicted by the dashed circle). Perturbation of the rotational component $\mathtt{R}_s$ serves effectively to rotate the transformed plane around the current $\mathbf{P}'_s$.

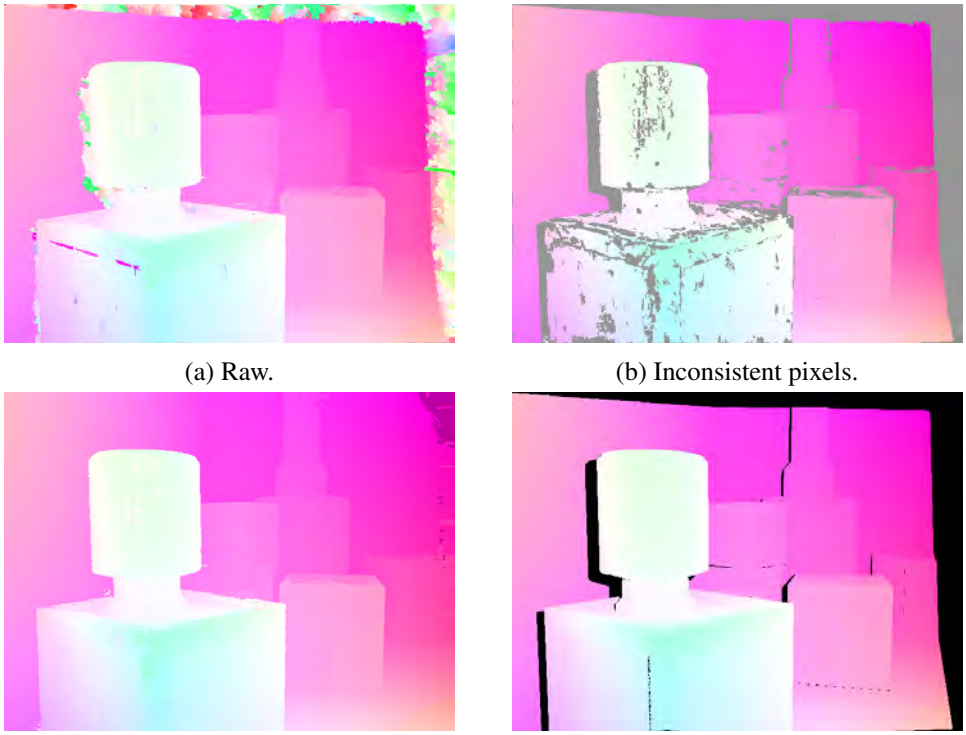
View Propagation. Most similarly to [HFR14] (as described in Section 6.2.2), which in turn builds upon [BRFK12,BRR11], as a last step when visiting a pixel $\mathbf{x}_s$ and given its assigned state vector $\theta_s = (Z_s, \mathbf{n}_s, \mathtt{R}_s, \mathbf{t}_s)$, we propose the inverted state $\theta'_s = (Z'_s, \mathbf{n}'_s, \mathtt{R}'_s, \mathbf{t}'_s)$ in the destination view. We compute θ'_s by $\mathbf{n}'_s = \mathtt{R}_s\mathbf{n}_s$, $\mathtt{R}'_s = \mathtt{R}_s^{-1}$, $\mathbf{t}'_s = -\mathtt{R}_s^{-1}\mathbf{t}$; the depth Z'_s is obtained by intersecting the transformed plane with the back-projection of $Z_s\mathtt{R}_s\mathbf{p}_s + \mathbf{t}_s$ projected to the nearest integer pixel, which is where in the destination view we then evaluate θ'_s. Geometrically, this amounts to considering the inverse rigid body motion applied to the transformed plane. Since we carry out our algorithm on both views in parallel and in opposite traversal orders, the most recent corresponding match available from the destination view has thus already been considered by the time $\mathbf{x}_s$ is reached.

7.2.4 Post-processing

In areas of the scene that are occluded in one of the two views, subject to the aperture problem, or poorly textured, our algorithm is likely to assign states that do not correspond to the correct flow (cf. Figure 7.5). If flow is computed in both directions, we can identify inconsistent state assignments by running a consistency check over 'forward' and 'backward' flow, labelling as inconsistent each pixel $\mathbf{x}_s$ that fails the following condition:

$$\left\|\mathbf{x}_s - H(\theta_s^{\mathrm{b}}) * \left(H(\theta_s^{\mathrm{f}}) * \mathbf{x}_s\right)\right\| \leq 1, \tag{7.6}$$

where θ_s^{f} determines the forward flow assigned in the source view to pixel $\mathbf{x}_s$, and θ_s^{b} the backward flow assigned in the destination view to the pixel $\theta_s^{\mathrm{f}} * \mathbf{x}_s$ rounded to the nearest integer coordinates. This generates a pixel mask that identifies pixels that subsequently undergo post-processing. For each $\mathbf{x}_s$ that failed the check, we first consider the pixels in a window around $\mathbf{x}_s$ that passed, adopting the homography of the pixel that is closest in appearance. Next, for



(a) Raw. (b) Inconsistent pixels.

(c) Post-processed. (d) Ground truth.

Figure 7.5: Post-processing, on the example of Crates1Htxtr2 from the UCL data set. Only the pixels that fail the consistency check (indicated in gray) undergo post-processing.

pixels $\mathbf{x}_s$ that still fail the check, we seek the nearest pixels above and below $\mathbf{x}_s$ that passed, and adopt the homography of the pixel closest in appearance. Finally, we proceed similarly for left and right.

7.3 Evaluation

We tested our method on the UCL optical flow data set [MHPB13] and on a subset of the Middlebury optical flow benchmark [BSL$^+$11] for which ground truth flow was available. Accordingly, we considered data sets exhibiting flow at small and large displacements (we set the threshold between the two at 25 pixels) and undergoing rigid, piecewise rigid, and non-rigid motions. A comparison over end point error (EPE)—computed according to (6.12)—is provided in Table 7.1 with respect to four competing methods. We ran our algorithm on all data sets in the table with a patch size of 21 $\times$ 21 for three iterations on a single particle. As in [BRFK12, BRR11], we set the weight α that balances the influence of gradient over color in (7.2) to 0.9, and γ in the adaptive support weighting to 10. The truncation constant κ of the smoothness term in (7.5) was set to 1 in all our experiments. Only a single dominant rigid body motion was recovered per data set, in the manner described in Section 7.2.3. Minimum depth was fixed to 0; maximum

depth per view was set to the maximum depth of triangulated matches that were inliers of the dominant motion. In the random search stage, maximum allowable deviation from the current rigid body motion was set to 0.01 for both the rotational (expressed in terms of quaternions) and translational components of the motion. Analogously to [BRFK12, BRR11], we set maximum flow per dataset. Camera calibration matrix $\mathtt{K}$ was fixed such that the focal length was 700 pixels and the principal point was in the image center.
Our method performs particularly well on the large displacement cases of the UCL dataset, and produces reasonable results for smaller displacements. Quantitative results show that our technique outperforms all four other methods in ca. 1/3 of the data sets (ca. 1/2 of the cases for large motion), while the end point error is lower than that of TV and LD in most of the cases. The color scheme used in Table 7.1 indicates that our approach is the one that is most frequently ranked in the first two positions (ca. 2/3 of the cases), when compared to the other four techniques. A visual comparison for four data sets by means of conventional 2D flow coloring (cf. Figure 4.2) with ground truth occlusion areas blacked out is given in Figure 7.6. The effect of the smoothness term can be seen in Figure 7.7, where we compare the resulting 2D flow for our algorithm with $\lambda = 0$ (no smoothness) and $\lambda = 0.005$ on the Middlebury dimetrodon data set. Additionally, we give the EPE results for $\lambda = 0$ and $\lambda = 0.01$ for all data sets in Table 7.1.

(Piecewise) Unrectified Stereo. For scenes undergoing only a single dominant rigid body motion, one could run our algorithm with no deviation allowed from the recovered dominant rigid body motion $(\mathtt{R}_\mathtt{E}, \mathbf{t}_\mathtt{E})$. We show precisely such a reconstruction for the Brickbox2t2 data set in Figure 7.8, providing a coloring of the recovered normal vectors, the depth map, and a colored point cloud rendered at a novel view. Locking the motion reduces our algorithm to an unrectified stereo matcher with slanted support windows, most closely akin to [BRFK12].
In order to give an impression of the limits of the approach, we recover the dominant rigid body motion on the street1Txtr1 data set in the manner described in Section 7.2.3 and obtain motions on the three independently moving cubes by manually supplying correspondences to the 5 point algorithm using RANSAC (cf. Figure 7.9). We show the result for allowing deviations from those four motions, and for allowing no deviation. We additionally show the resulting point cloud where no deviation is allowed. Note that the three cubes are not reconstructed with commensurate size; this is a consequence of each piecewise reconstruction being individually up to a scale ambiguity.

Radial Flow. Certain types of camera motions can be difficult to handle for flow methods that use a 2D parameterization. For instance, camera zoom induces a radial flow pattern around the viewing direction, which conflicts with a smoothness assumption that promotes neighboring flow vectors to be similar. However, our approach is flexible enough to recover the homographies induced by this motion, as illustrated in Figure 7.10.

Limitations. We kept the patch size identical across all our experiments, regardless of image size or scale. As in patch-based stereo techniques, our approach is sensitive to the aperture problem, and more generally to poorly textured surfaces. It is this problem of inadequate match discriminability that accounts for the comparatively poor performance of our algorithm for Robot,

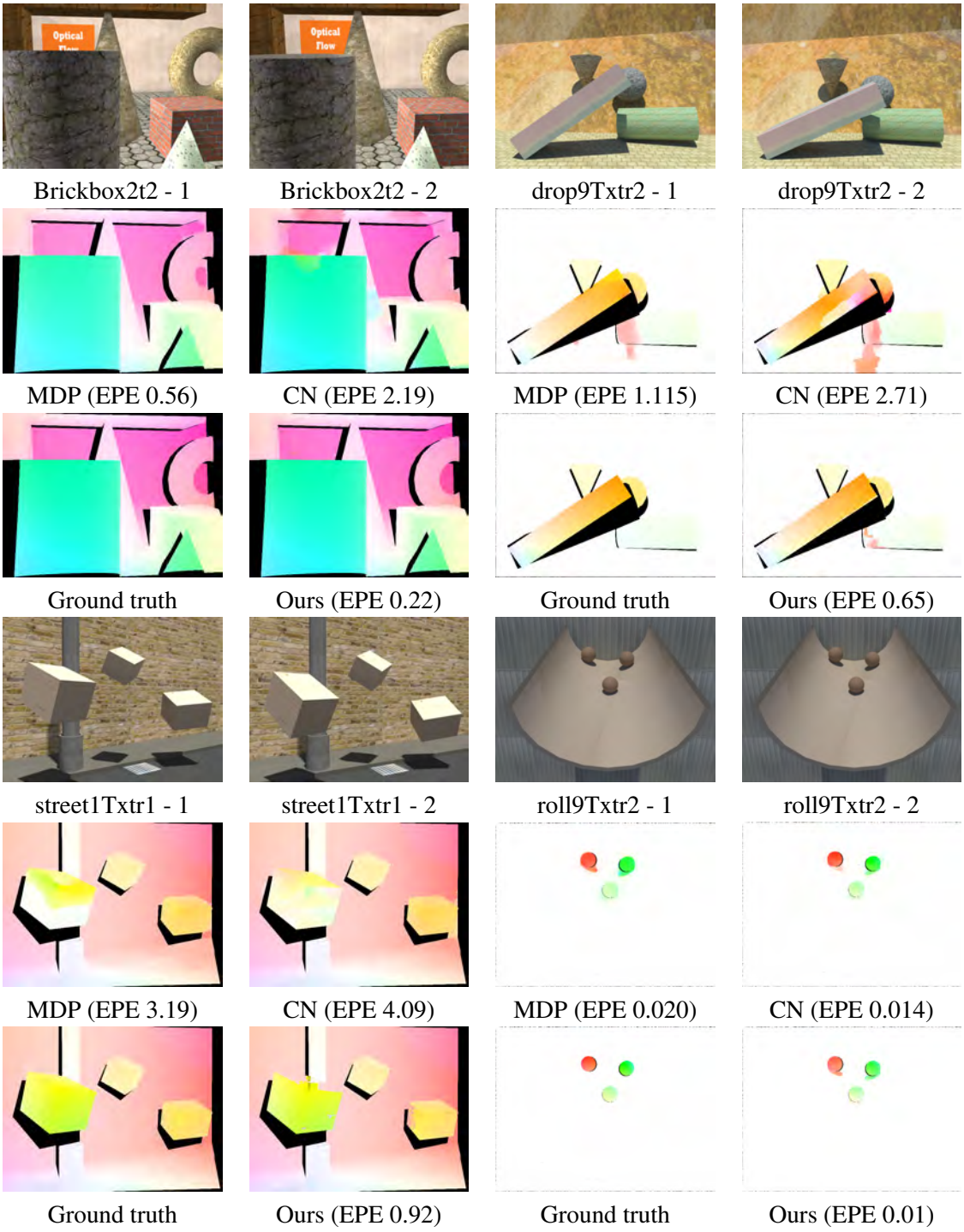
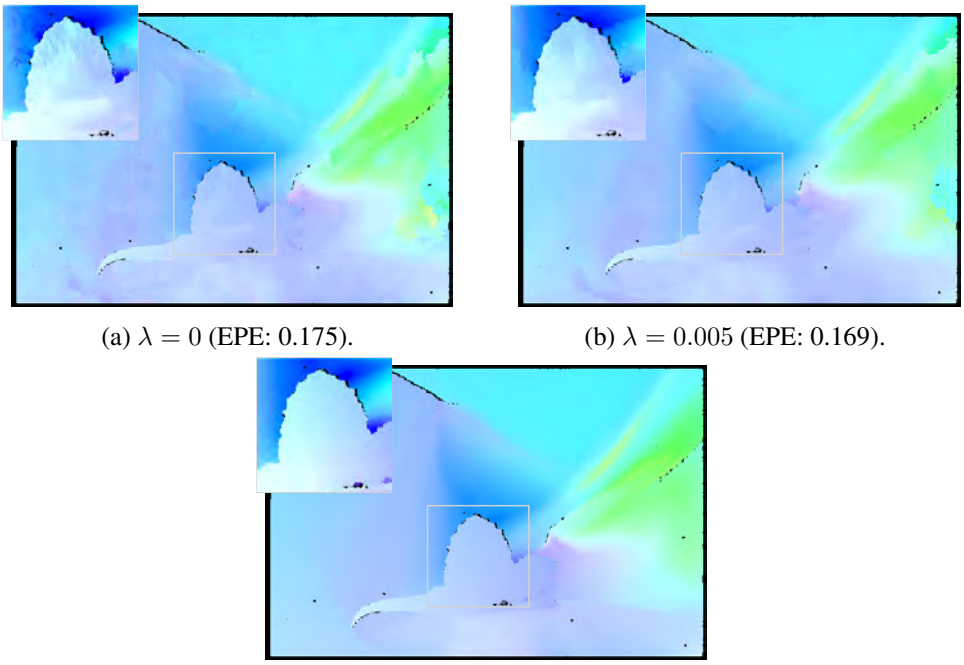


Figure 7.6: 2D flow colorings for a subset of the UCL optical flow data set. EPE = End Point Error. CN = Secrets of Optical Flow [SRB10]. MDP = Motion Detail Preserving Optical Flow [XJM12]. Results correspond to Table 7.1.

	TV	LD	CN	MDP	Ours$_{\lambda=0.005}$	Ours$_{\lambda=0}$	Ours$_{\lambda=0.01}$
UCL Large Displ.							
Crates1	3.46	3.10	3.15	**1.65**	2.37	2.62	2.9
Crates2	4.62	2.51	10.4	**1.35**	1.71	1.84	1.73
Mayan1	2.33	5.56	1.71	0.48	**0.16**	0.17	0.18
Robot	2.34	1.21	1.53	**0.7**	1.85	2.14	1.96
Crates1Htxtr2	1.11	0.54	1.64	**0.28**	0.29	0.39	0.3
Crates2Htxtr1	3.13	0.81	8.8	**0.37**	0.47	0.45	0.64
Brickbox1t1	1.09	2.6	0.22	0.2	**0.15**	0.16	0.15
Brickbox2t2	7.48	3.51	2.19	0.56	**0.22**	0.2	0.22
GrassSky0	2.1	1.04	1.3	0.47	**0.27**	0.3	0.27
GrassSky9	0.72	0.51	0.27	0.29	**0.25**	0.34	0.26
blow19Txtr2†	0.53	0.32	**0.19**	0.26	0.22	0.23	0.27
drop9Txtr2†	5.2	4.37	2.71	1.15	**0.65**	0.75	0.86
street1Txtr1†	3.65	2.66	4.09	3.19	**0.92**	1.72	1.45
UCL Small Displ.							
Mayan2	0.44	0.35	0.21	0.23	**0.17**	0.19	0.18
YosemiteSun†	0.31	**0.18**	0.23	3.79	0.33	0.35	0.38
GroveSun	0.58	0.48	**0.23**	0.43	0.24	0.24	0.23
Sponza1	1.01	**0.91**	1.1	1.08	2.75	2.84	2.8
Sponza2	0.53	**0.48**	1.6	1.77	2.61	2.58	2.61
TxtRMovement	3.17	0.36	**0.13**	0.19	1.71	1.7	1.72
TxtLMovement	1.52	0.6	**0.12**	0.23	1.73	1.76	1.76
blow1Txtr1†	0.09	0.08	**0.03**	0.05	0.04	0.04	0.04
drop1Txtr1†	0.12	0.08	0.05	0.06	**0.04**	0.04	0.04
roll1Txtr1†	0.004	**0.002**	**0.002**	**0.002**	**0.002**	0.002	0.002
roll9Txtr2†	0.04	0.02	**0.01**	0.02	**0.01**	0.01	0.01
Middlebury							
Dimetrodon†	0.211	0.117	**0.115**	0.153	0.169	0.174	0.17
Grove2	0.220	0.149	**0.091**	0.15	0.184	0.187	0.3
Grove3	0.745	0.657	**0.438**	0.53	0.517	0.455	0.97
Hydrangea†	0.196	0.178	**0.154**	0.164	0.222	0.207	0.234
RubberWhale†	0.135	0.120	**0.077**	0.09	0.114	0.12	0.125
Urban2	0.506	0.334	**0.207**	0.32	0.3	0.312	0.29
Urban3	1.132	0.600	**0.377**	0.42	0.905	1.27	1.03
Venus	0.408	0.433	**0.229**	0.28	0.342	0.342	0.434

Table 7.1: End point error (EPE) scores. TV = A Duality Based Approach for Realtime TV-L1 Optical Flow [ZPB07], LD = Large Displacement Optical Flow [BM11], CN = Secrets of Optical Flow [SRB10], MDP = Motion Detail Preserving Optical Flow [XJM12]. Cell colors indicate ranking among the five methods, from best to worst: green, light green, yellow, orange, red. Gray cells are shown for comparison but are not included in the ranking. † indicates that the scene is non-static.



(a) $\lambda = 0$ (EPE: 0.175).

(b) $\lambda = 0.005$ (EPE: 0.169).

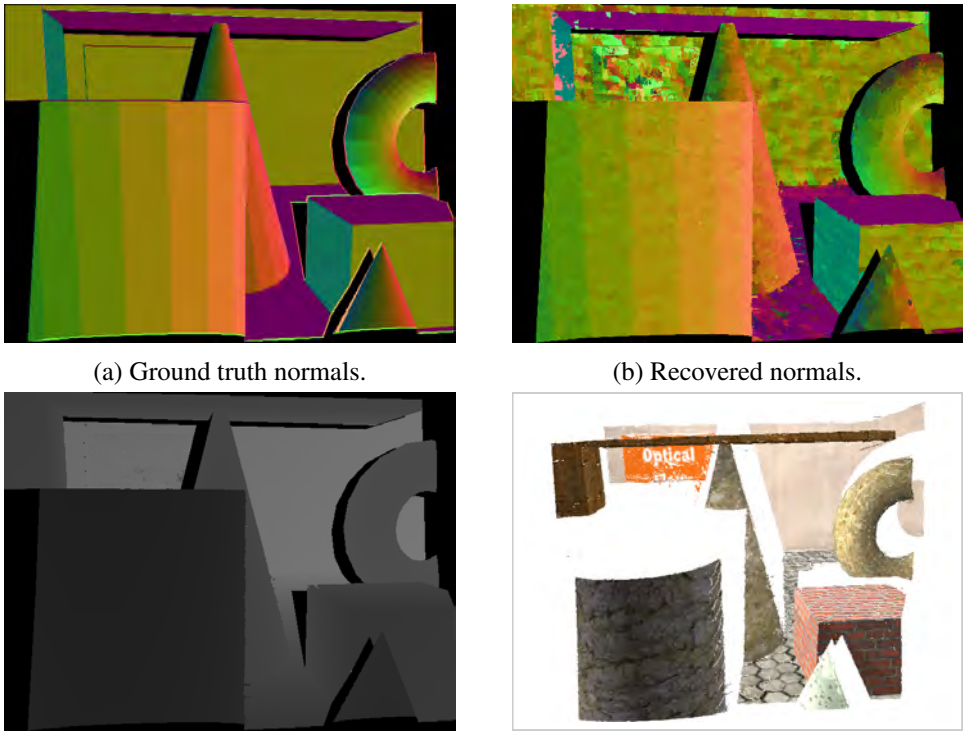
(c) Ground truth.

Figure 7.7: The effect of the smoothness term on dimetrodon for $\lambda = 0$ (no smoothness) and $\lambda = 0.005$. Inlay shown with contrast stretch; results best viewed zoomed in. 2D flow coloring and EPE without post-processing.

Sponza1, Sponza2, TxtRMovement, and TxtLMovement. An obvious way to alleviate this problem where applicable is to set the patch size appropriately. A direction for future work could be to develop a smoothness term that promotes not only smoothness of the 2D flow, but explicitly exploits the geometric interpretation of the parameterization to promote similarity of the 9 DoF states themselves.

7.4 Discussion

Similarly to the case of 6 DoF 3D rigid body motions (cf. Section 6.4), the value of restricting attention to 9 DoF plane-induced homographies that can be judged geometrically *plausible*—again in contrast to proceeding in a more strictly randomized fashion as in the manner of classical PatchMatch [BSFG09], whether for initialization or refinement—has the advantage of promoting convergence. In Figures 7.11-7.25, we compare results—having set $\lambda = 0.005$—for two iterations on Brickbox2t2 (rigid), street1Txtr1 (piecewise rigid), and Dimetrodon (non-rigid), (i) in the manner of the published algorithm (as outlined in Section 7.2), (ii) by initializing labels without sparse matches, such that $(\mathtt{R}_s, \mathbf{t}_s)$ be obtained by sampling randomly around $(\mathtt{R}_\mathtt{E}, \mathbf{t}_\mathtt{E})$, (iii) by independently perturbing *each* of the four components of $\theta_s = (Z_s, \mathbf{n}_s, \mathtt{R}_s, \mathbf{t}_s)$, (iv) without enforcing the validity check, which requires that source and destination 3D planes both



(a) Ground truth normals.

(b) Recovered normals.

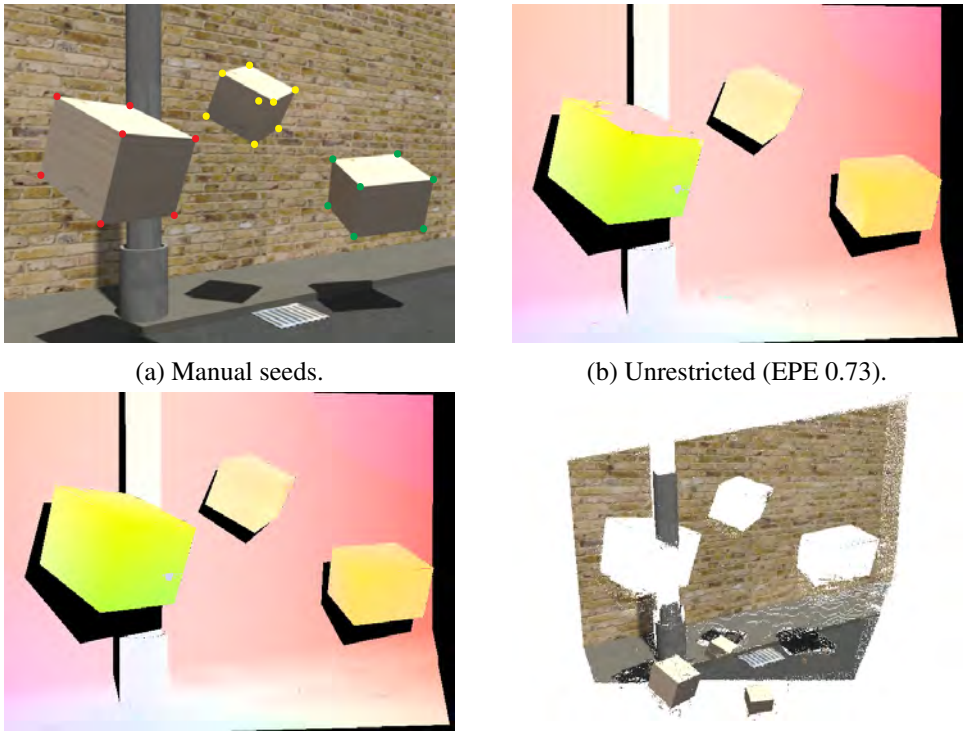
(c) Recovered depth.

(d) Colored point cloud at novel viewpoint.

Figure 7.8: Restriction to the recovered dominant rigid body motion $(\mathtt{R}_{\mathtt{E}}, \mathbf{t}_{\mathtt{E}})$ for the Brickbox2t2 data set. Estimation of the plane normals and depth on a static scene, and rendering as a colored point cloud.

point towards the camera, and (v) without view propagation. Note that in (ii) and (iii), sampling around a rigid body motion is carried out not as in Figure 7.4, but by randomly perturbing the respective rotation and translation *independently*; notably, (ii), (iii), (iv), and (v) each additionally aim illustrate the impact of a number of my more major personal technical contributions to the joint paper upon which this chapter is based [HBK+14], relative to the state prior to my involvement (cf. the note on authorship in Section 1.3).

While for each of the five cases considered on the example of Brickbox2t2 in Figures 7.11-7.15—with the exception of (iii) in Figure 7.13—the quality of recovered 2D flow appears almost indistinguishable to the human eye, the same cannot be said of the recovered depth and normals. On the example of street1Txtr1 in Figures 7.16-7.20, convergence is fastest for (i) in Figure 7.16; the algorithm fails to capture the motion of the three cubes within two iterations if propagation is carried out as in (iii) or in the absence of view propagation, as illustrated in Figures 7.18 and 7.20, respectively. In Figures 7.21-7.25, the example of the Dimetrodon data set likewise underscores the effectiveness in recovering 2D flow of our form of refinement over its more indiscriminately randomized form in (iii), as shown in Figure 7.23; in what concerns depth and normals, however, results appear unsatisfactory for each of the five cases. These comparatively poor depth and normals can perhaps be explained by the fact that object moving in the image



(a) Manual seeds. (b) Unrestricted (EPE 0.73).

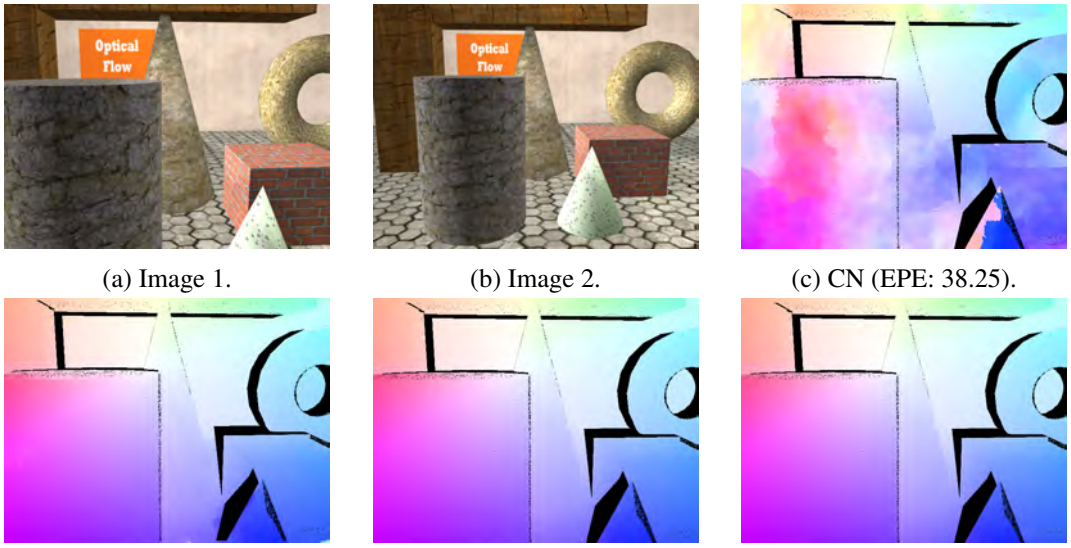
(c) Restricted (EPE: 0.468). (d) Restricted.

Figure 7.9: Result obtained on street1Txtr1 by seeding with the dominant motion obtained by the 5 point algorithm with RANSAC on all ASIFT matches and on the three additional sets of manually provided matches (indicated in red, yellow, and green), giving four motions in total. Results shown for deviation allowed from those four motions, and for no deviation allowed. Otherwise, we used the same parameter settings to compute our results as in Table 7.1.

plane can be explained in 3D as a small object situated close to the camera undergoing slow motion or as a large object far from the camera undergoing fast motion, a problem perhaps less pronounced in the case of (piecewise) rigid object motion because spatial propagation is carried out prior to refinement. Note that the apparent unimportance of initialization using sparse matches in the manner of our published algorithm relative to (ii)—where labels are initialized without sparse matches—is itself explained largely by the fact that our published algorithm's refinement strategy itself makes heavy use of sparse matches, thereby replacing many of the initial labels in (ii) with labels based in turn on sparse matches themselves.

7.5 Conclusion

We have presented a new optical flow technique that uses a geometrically motivated motion model and exploits that model to carry out the optimization in a manner that aims to focus attention on geometrically plausible motions. While the model lives in a continuous, high-



(a) Image 1. (b) Image 2. (c) CN (EPE: 38.25).

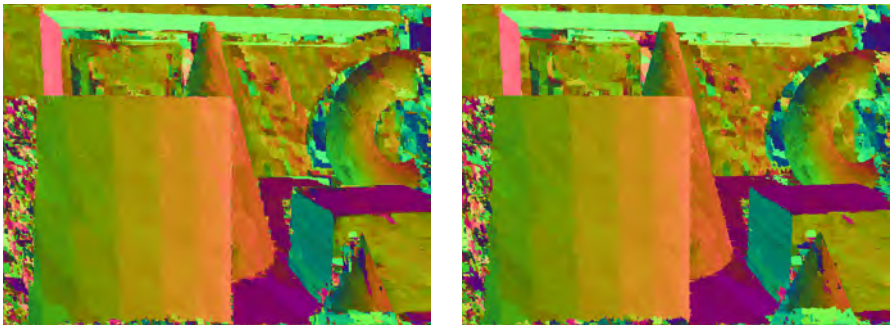
(d) MDP (EPE: 1.02). (e) Ours (EPE: 0.42). (f) Ground truth.

Figure 7.10: Result on a challenging case with large displacement camera zoom, causing a radial flow pattern. Note that this sequence is not part of the published UCL optical flow data set. We used the same parameter settings to compute our results as in Table 7.1.

dimensional space that would prove challenging to optimize using conventional methods, we show PMBP to be well suited for the task. We obtain 2D flow that compares favorably to other state-of-the-art techniques and manage to handle both small and large displacements. Our smoothness term helps promote smoothness of the obtained 2D flow fields. A side effect of our approach is that—provided rigid body motions are reasonable—depth can be directly extracted from the parameterization, which can in turn be used to construct a point cloud.

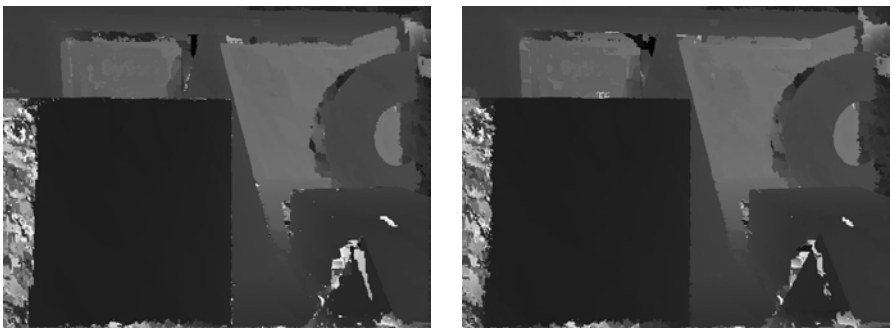
Frame 1, Iteration 1 Frame 1, Iteration 2

(a) 2D flow colorings.



Frame 1, Iteration 1 Frame 1, Iteration 2

(b) Normals.



Frame 1, Iteration 1 Frame 1, Iteration 2

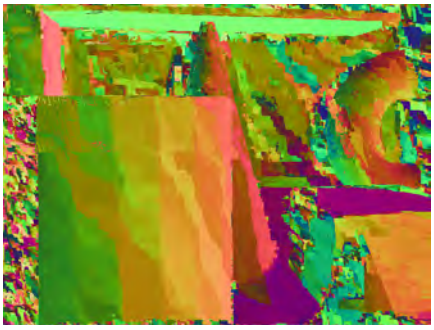
(c) Depth.

Figure 7.11: Brickbox2t2, published algorithm.

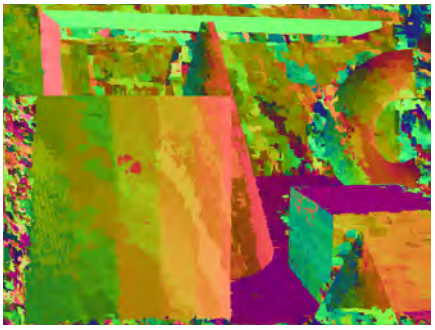
Frame 1, Iteration 1

Frame 1, Iteration 2

(a) 2D flow colorings.

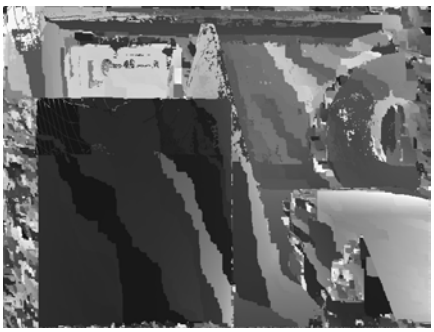


Frame 1, Iteration 1

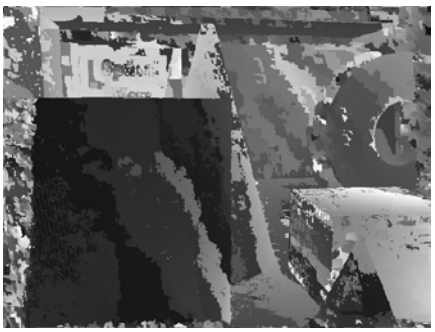


Frame 1, Iteration 2

(b) Normals.



Frame 1, Iteration 1



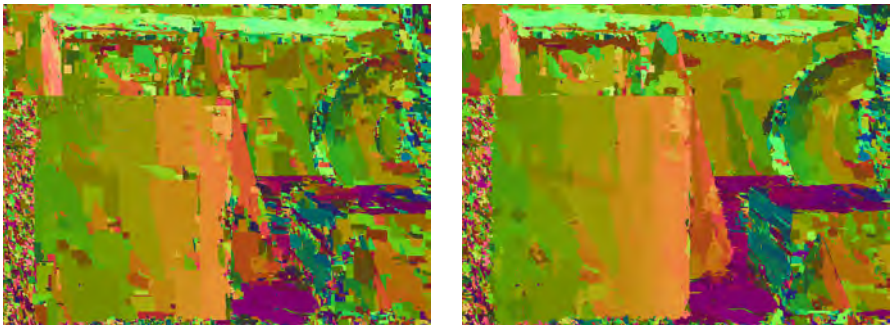
Frame 1, Iteration 2

(c) Depth.

Figure 7.12: Brickbox2t2, erstwhile initialization.

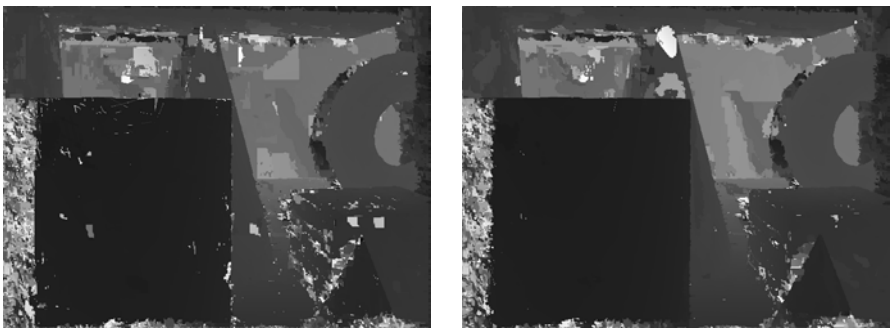
Frame 1, Iteration 1 Frame 1, Iteration 2

(a) 2D flow colorings.



Frame 1, Iteration 1 Frame 1, Iteration 2

(b) Normals.



Frame 1, Iteration 1 Frame 1, Iteration 2

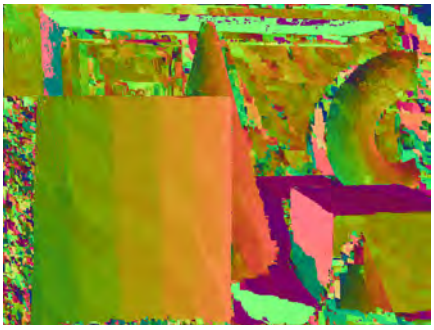
(c) Depth.

Figure 7.13: Brickbox2t2, erstwhile refinement.

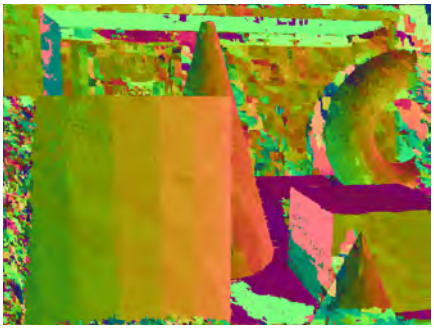
Frame 1, Iteration 1

Frame 1, Iteration 2

(a) 2D flow colorings.



Frame 1, Iteration 1



Frame 1, Iteration 2

(b) Normals.

Frame 1, Iteration 1

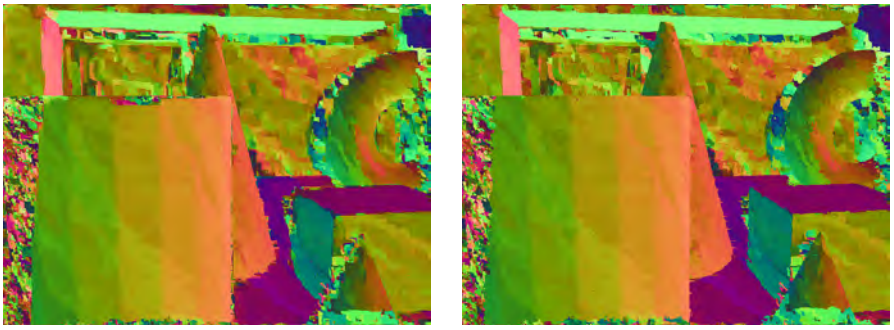
Frame 1, Iteration 2

(c) Depth.

Figure 7.14: Brickbox2t2, no validity check.

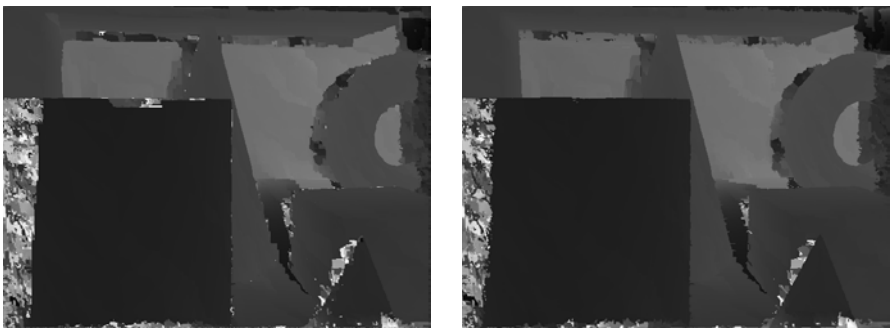
Frame 1, Iteration 1 Frame 1, Iteration 2

(a) 2D flow colorings.



Frame 1, Iteration 1 Frame 1, Iteration 2

(b) Normals.



Frame 1, Iteration 1 Frame 1, Iteration 2

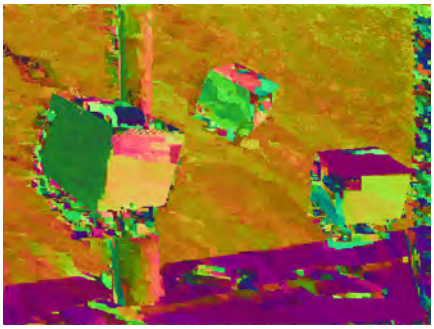
(c) Depth.

Figure 7.15: Brickbox2t2, no view propagation.

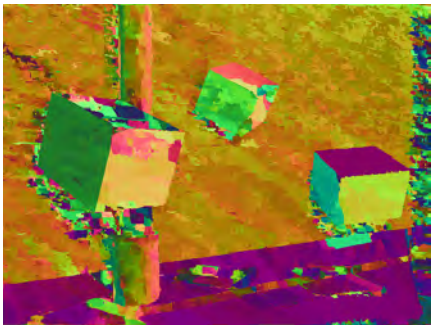
Frame 1, Iteration 1

Frame 1, Iteration 2

(a) 2D flow colorings.

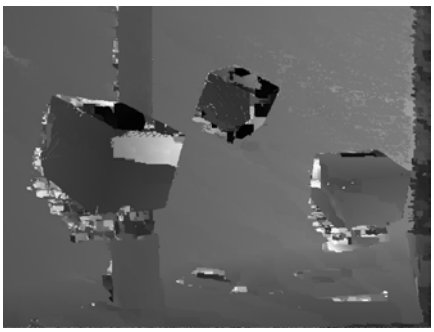


Frame 1, Iteration 1

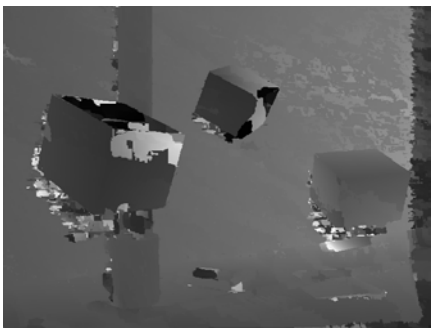


Frame 1, Iteration 2

(b) Normals.



Frame 1, Iteration 1



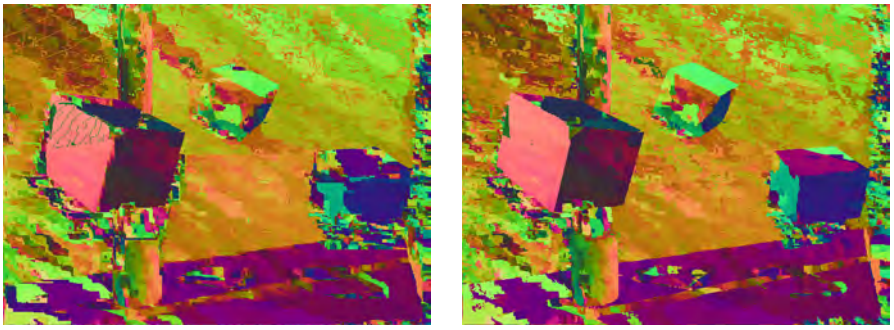
Frame 1, Iteration 2

(c) Depth.

Figure 7.16: street1Txtr1, published algorithm.

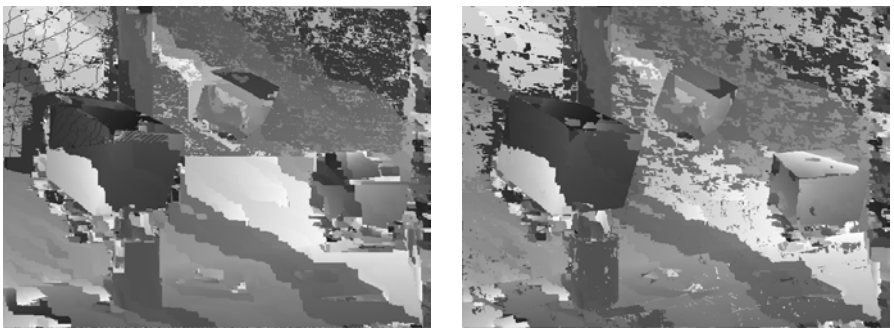
Frame 1, Iteration 1 Frame 1, Iteration 2

(a) 2D flow colorings.



Frame 1, Iteration 1 Frame 1, Iteration 2

(b) Normals.



Frame 1, Iteration 1 Frame 1, Iteration 2

(c) Depth.

Figure 7.17: street1Txtr1, erstwhile initialization.

Frame 1, Iteration 1

Frame 1, Iteration 2

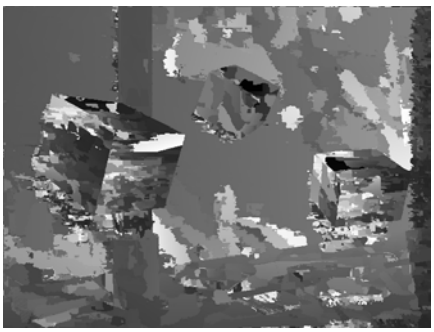
(a) 2D flow colorings.

Frame 1, Iteration 1

Frame 1, Iteration 2

(b) Normals.

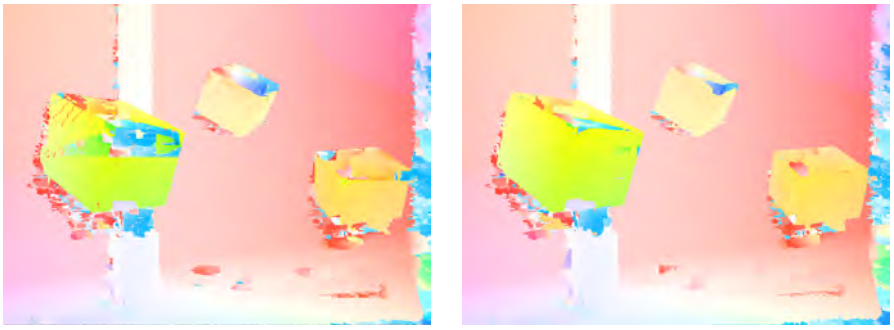
Frame 1, Iteration 1



Frame 1, Iteration 2

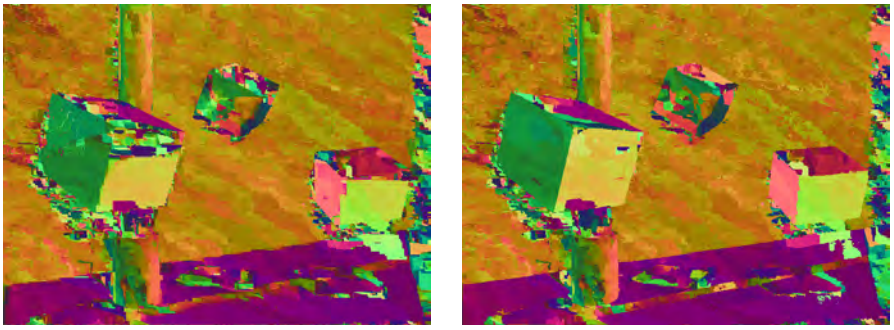
(c) Depth.

Figure 7.18: street1Txtr1, erstwhile refinement.



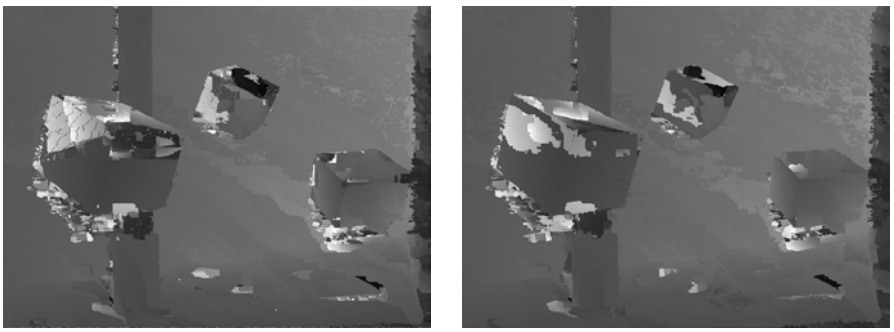
Frame 1, Iteration 1 Frame 1, Iteration 2

(a) 2D flow colorings.



Frame 1, Iteration 1 Frame 1, Iteration 2

(b) Normals.



Frame 1, Iteration 1 Frame 1, Iteration 2

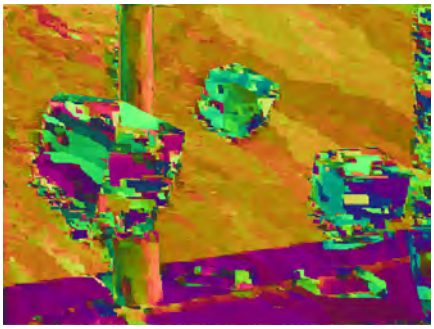
(c) Depth.

Figure 7.19: street1Txtr1, no validity check.

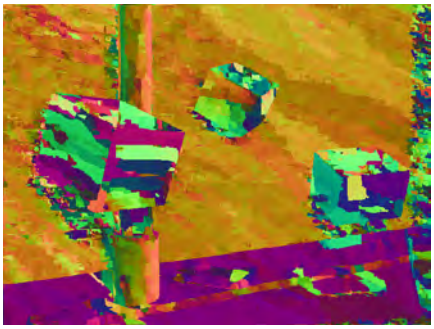
Frame 1, Iteration 1

Frame 1, Iteration 2

(a) 2D flow colorings.

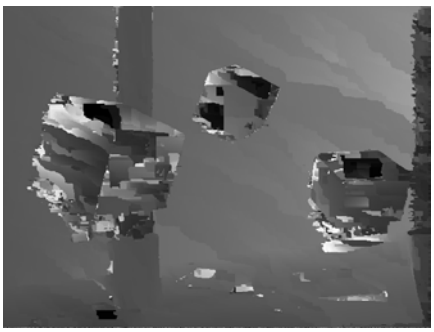


Frame 1, Iteration 1



Frame 1, Iteration 2

(b) Normals.

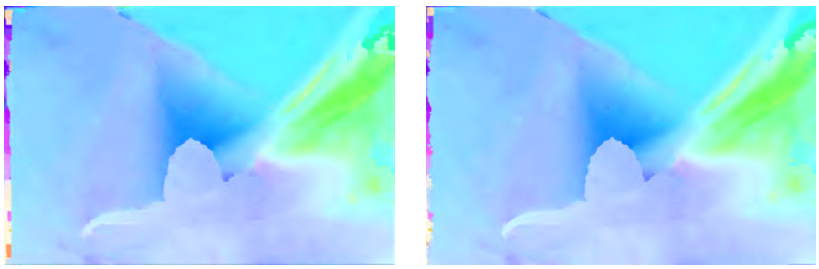


Frame 1, Iteration 1

Frame 1, Iteration 2

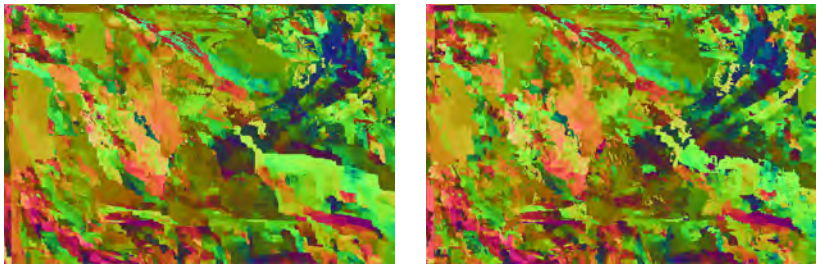
(c) Depth.

Figure 7.20: street1Txtr1, no view propagation.



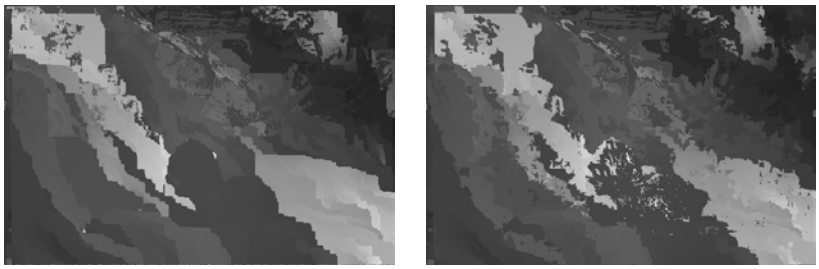
Frame 1, Iteration 1 Frame 1, Iteration 2

(a) 2D flow colorings.



Frame 1, Iteration 1 Frame 1, Iteration 2

(b) Normals.



Frame 1, Iteration 1 Frame 1, Iteration 2

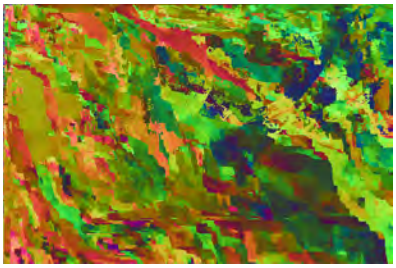
(c) Depth.

Figure 7.21: Dimetrodon, published algorithm.

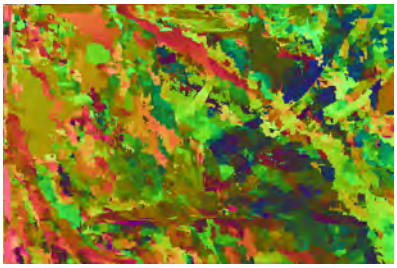
Frame 1, Iteration 1

Frame 1, Iteration 2

(a) 2D flow colorings.

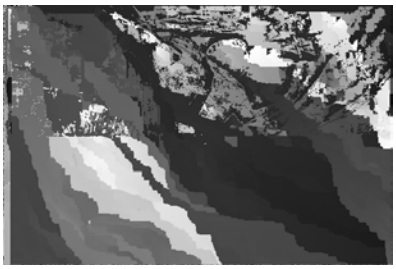


Frame 1, Iteration 1

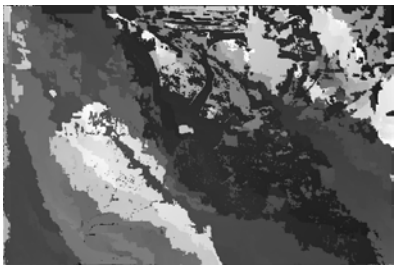


Frame 1, Iteration 2

(b) Normals.



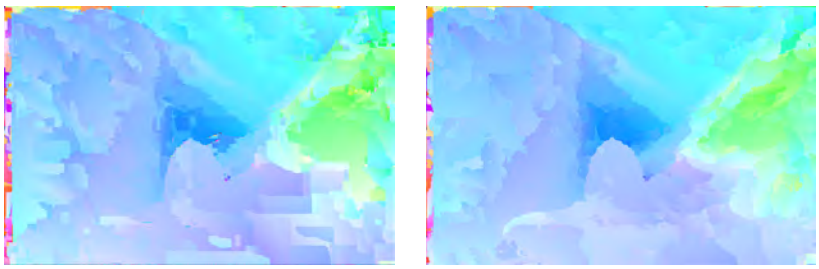
Frame 1, Iteration 1



Frame 1, Iteration 2

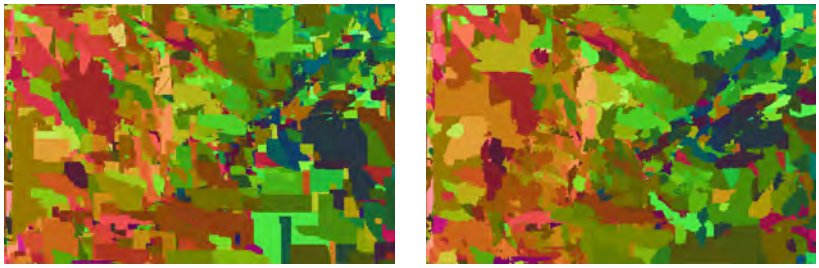
(c) Depth.

Figure 7.22: Dimetrodon, erstwhile initialization.



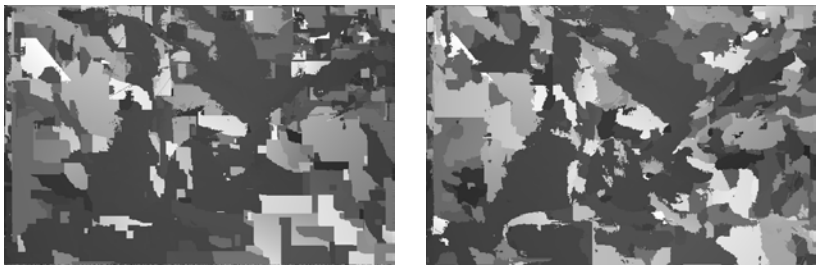
Frame 1, Iteration 1 Frame 1, Iteration 2

(a) 2D flow colorings.



Frame 1, Iteration 1 Frame 1, Iteration 2

(b) Normals.



Frame 1, Iteration 1 Frame 1, Iteration 2

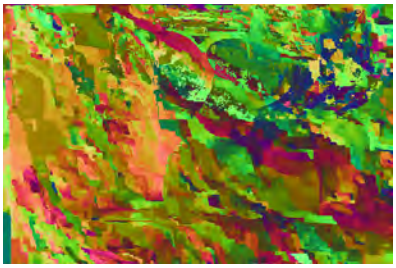
(c) Depth.

Figure 7.23: Dimetrodon, erstwhile refinement.

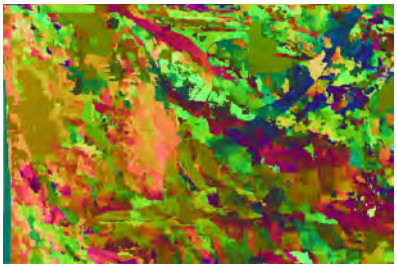
Frame 1, Iteration 1

Frame 1, Iteration 2

(a) 2D flow colorings.

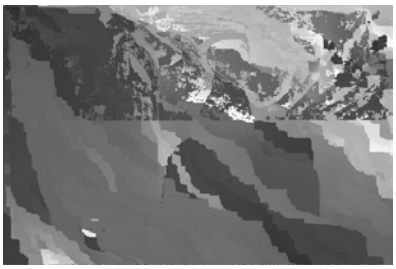


Frame 1, Iteration 1

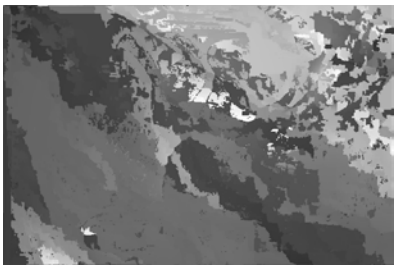


Frame 1, Iteration 2

(b) Normals.



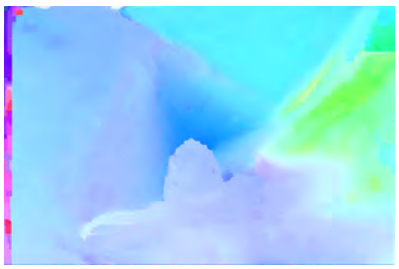
Frame 1, Iteration 1



Frame 1, Iteration 2

(c) Depth.

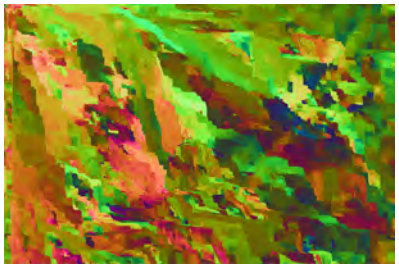
Figure 7.24: Dimetrodon, no validity check.



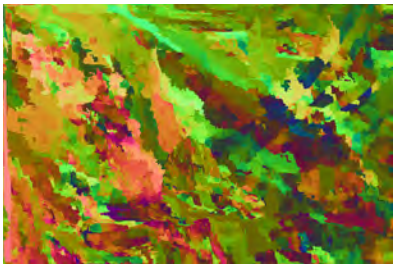
Frame 1, Iteration 1

Frame 1, Iteration 2

(a) 2D flow colorings.

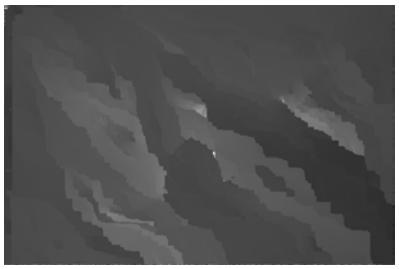


Frame 1, Iteration 1

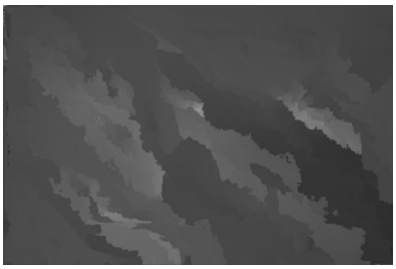


Frame 1, Iteration 2

(b) Normals.



Frame 1, Iteration 1



Frame 1, Iteration 2

(c) Depth.

Figure 7.25: Dimetrodon, no view propagation.

CHAPTER 8

Conclusions and Future Work

We showed tailored variants of PatchMatch to serve as effective optimizers for *dense large displacement* correspondence field estimation in *continuous, high dimensional* label spaces, where 2D motion in the image plane was cast terms of the *rigid motion of points* in 3D. Notably, where depth maps are available, we proposed *patches of the points* encoded in such depth maps as a way to circumvent the traditional assumption of local surface planarity. The essence of the effectiveness of PatchMatch is to *grow and refine* sparse correspondence seeds, whether obtained at random in the manner of classical PatchMatch [BSFG09] or calling on some form of prior knowledge; we showed obtaining *sparse correspondence seeds derived from sparse matching* in image space to be an effective strategy for promoting convergence with respect to the number of iterations of PatchMatch (cf. Sections 6.4 and 7.4). Proceeding in this manner can be thought of as part of a broader strategy of focusing attention to (geometrically) *plausible* labels. A list of secondary contributions per domain addressed in this thesis—depth SR, RGB-D scene flow, and optical flow—is provided below; in the section that follows, we additionally outline some potential directions for future work.

Depth SR. Our depth SR method differs from all previous depth SR methods in that ours makes *no use of ancillary data*, such as a color image at the target resolution, multiple aligned depth maps, or a database of high-resolution depth patches. We instead reason only in terms of matches across depth between patches of the points encoded in the single input depth map, consisting of the inliers of spheres and undergoing respective 6 DoF 3D rigid body motions. We show our results to be highly competitive with those of alternative techniques that do leverage ancillary data.

RGB-D Scene Flow. Building upon the spherical patches introduced in our work on depth SR, we succeed in showoing attractive scene flow results on challenging synthetic and real-world scenes that push the practical limits of the assumptions of brightness constancy and local surface planarity. An important novelty over the spherical patches of our depth SR work is to reason not in terms of a single fixed sphere radius r, but in terms of adaptive radii $r_{\mathbf{x}}$ at each

pixel $\mathbf{x}$; proceeding in this manner allows for ameliorating the problem of sphere inlier counts varying as a function of sphere depth, thereby in turn allowing for a more uniform matching quality than is possible for fixed r. As a consequence of our approach, our output is a dense field of 6 DoF 3D rigid body motions, in contrast to the 3D translation vectors that are the norm in the literature on scene flow.

Optical Flow. We show that a variant of PatchMatch presents itself as an effective optimizer for what is ostensibly an extreme *overparameterization* of 2 DoF optical flow in terms of 9 DoF plane-induced homographies, such that each pixel $\mathbf{x}$ be assigned a 3 DoF 3D plane and a 6 DoF 3D rigid body motion that the plane undergoes to describe motion in the image plane of the respective patch centered on $\mathbf{x}$. Proceeding in this manner has the advantage of reasoning about matching in a pair of RGB frames in terms of possible underlying 3D motions, recognizing that 2D motion in the image plane is ultimately a function of 3D motion in the scene.

8.1 Future Work

We broadly outline below a number of potential directions for future work in geometrically motivated, dense, continuous, and large displacement matching by means of growing and refining sparse correspondence seeds. These encompass improved handling of untextured areas, use of explicit motion segmentation, further investigation of the relaxation of the epipolar constraint for the reconstruction of non-rigid scenes, the leveraging of 3D features, and consideration for a 9 DoF smoothness term.

Untextured Areas. To carry out PatchMatch refinement (perturbation) in areas of low or indiscriminative texture (e.g., over an untextured wall or a cloudless blue sky) is certain to lead to getting one trapped in local minima situated far from the genuine correspondence. Instead, consider allowing *only propagation* in such areas (i.e., disallow refinement), in the aim of instead permitting refinement only in areas where it is possible to decide—by virtue of discriminative texture—whether or not the perturbation of a label yields a better match. Additionally, consider initializing untextured pixels with a dummy motion of infinite cost, and disallow perturbation of this dummy motion.

Motion Segmentation. For computing our 9 DoF optical flow (cf. Chapter 7), consider recovering additional 3D rigid body motions (and triangulating additional corresponding sparse seed points) using a motion segmentation algorithm. For computing our 6 DoF scene flow (cf. Chapter 6), consider feeding the output of our PatchMatch variant to a motion segmentation algorithm, and subsequently promoting or enforcing the resulting motions of the segmentation in a post-processing step.

Relaxed Stereo. The best point cloud shown in Chapter 7 is in Figure 7.8, where the 3D rigid body motion is fixed to the recovered dominant rigid body motion $(\mathtt{R}_\mathtt{E}, \mathbf{t}_\mathtt{E})$ of the static scene, whereby our algorithm in effect carries out a form of unrectified stereo matching. The reason

this is effective is because it reduces the search space from 9 DoF to only 3 DoF, rendering the matching problem considerably simpler, and in effect further restricting attention only to *plausible* labels. To recover better *points and normals* over non-rigid scenes, rather than choose randomly between (i) fixing the current 3D rigid body motion and perturbing only the depth and normal, and (ii) fixing the current depth and normal and perturbing only the rigid motion, let (i) dominate. Note, however, that this strategy may fail to capture motions that deviate substantially from the single recovered dominant 3D rigid body motion; accordingly, it is something to consider in conjunction with a motion segmentation (above).

3D Features. For depth SR and SphereFlow in Chapters 5 and 6, respectively, consider (additionally) using 3D features to obtain sparse correspondence seeds for initialization, in order to exploit what information is available in depth.

9 DoF Smoothness Term. In the aim of tailoring the approach presented in Chapter 7 to *obtaining attractive point clouds* rather than only focusing attention on 2D flow, it could be valuable to consider smoothness terms that take into account all 9 DoF of the parameterization. Note that as it was the case that attention in Chapter 7 was directed at 2D flow, we chose in that work to opt instead for a simpler 2 DoF smoothness term.

APPENDIX A

Arriving at Z/f

We state in Section 6.2.1 that, by similar triangles, the distance between two points $\mathbf{P}, \mathbf{P}' \in \mathbb{R}^3$ both situated at the same depth Z and projecting to neighboring pixels $\mathbf{x}, \mathbf{x}'$ is given by Z/f, where f is the camera's focal length in units of pixels (cf. Figure 6.1). An alternative derivation in the x- and y-directions of the image plane is obtained by computing the distance between the point $\mathbf{P}$ at depth Z projecting to the pixel $(i, j)^\top$, and the points at the same depth Z projecting to the pixel neighbors $(i+1, j)^\top, (i, j+1)^\top$, respectively. The point $\mathbf{P}$ is given by $Z \cdot \mathtt{K}^{-1}(i, j, 1)^\top$, where $\mathtt{K}$ is the 3×3 camera calibration matrix (cf. Section 2.4):

$$\begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix},$$

where $(x_0, y_0)^\top$ is the camera's principal point in pixel units. It follows that $\mathbf{P}$ is given in closed form by

$$Z \cdot \mathtt{K}^{-1} \begin{pmatrix} i \\ j \\ 1 \end{pmatrix} = Z \begin{bmatrix} 1/f & 0 & -x_0/f \\ 0 & 1/f & -y_0/f \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} i \\ j \\ 1 \end{pmatrix} = Z \begin{pmatrix} \frac{i-x_0}{f} \\ \frac{j-y_0}{f} \\ 1 \end{pmatrix}.$$

Horizontal Neighbors. We first consider the Euclidean distance δ_Z between two points both situated at depth Z and projecting to the horizontal pixel neighbors $(i, j)^\top, (i+1, j)^\top$:

$$\begin{aligned} \delta_Z &= \left\| Z \cdot \mathtt{K}^{-1}(i+1, j, 1)^\top - Z \cdot \mathtt{K}^{-1}(i, j, 1)^\top \right\|_2 \\ &= \left\| Z \left(\mathtt{K}^{-1}(i+1, j, 1)^\top - \mathtt{K}^{-1}(i, j, 1)^\top \right) \right\|_2 \\ &= \left\| Z \begin{pmatrix} \frac{i+1-x_0}{f} - \frac{i-x_0}{f} \\ \frac{j-y_0}{f} - \frac{j-y_0}{f} \\ 1 - 1 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} \frac{Z}{f} \\ 0 \\ 0 \end{pmatrix} \right\|_2 = \sqrt{\left(\frac{Z}{f}\right)^2} = Z/f. \end{aligned}$$

Vertical Neighbors. Finally, we consider the analogous Euclidean distance δ'_Z for the vertical pixel neighbors $(i, j)^\top, (i, j+1)^\top$:

$$\begin{aligned}
\delta'_Z &= \left\| Z \cdot \mathtt{K}^{-1}(i, j+1, 1)^\top - Z \cdot \mathtt{K}^{-1}(i, j, 1)^\top \right\|_2 \\
&= \left\| Z \left(\mathtt{K}^{-1}(i, j+1, 1)^\top - \mathtt{K}^{-1}(i, j, 1)^\top \right) \right\|_2 \\
&= \left\| Z \begin{pmatrix} \frac{i-x_0}{f} - \frac{i-x_0}{f} \\ \frac{j+1-y_0}{f} - \frac{j-y_0}{f} \\ 1-1 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} 0 \\ \frac{Z}{f} \\ 0 \end{pmatrix} \right\|_2 = \sqrt{\left(\frac{Z}{f}\right)^2} = Z/f.
\end{aligned}$$

Bibliography

[Ang06] Edward Angel. *Interactive Computer Graphics - A Top-Down Approach Using OpenGL*. Addison-Wesley, 4th edition, 2006.

[BA96] Michael 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.

[BI99] Aaron F. Bobick and Stephen S. Intille. Large Occlusion Stereo. *International Journal of Computer Vision*, 33(3):181–200, 1999.

[Bir98] Stan Birchfield. An Introduction to Projective Geometry (for Computer Vision), 1998.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[BK03] Yuri Boykov and Vladimir Kolmogorov. Computing Geodesics and Minimal Surfaces via Graph Cuts. In *International Conference on Computer Vision*, pages 26–33, 2003.

[BM04] Simon Baker and Iain Matthews. Lucas-Kanade 20 Years On: A Unifying Framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[BM11] Thomas Brox and Jitendra Malik. Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, 2011.

[BMK13] Tali Basha, Yael Moses, and Nahum Kiryati. Multi-view Scene Flow Estimation: A View Centered Variational Approach. *International Journal of Computer Vision*, 101(1):6–21, 2013.

[Bou] Jean-Yves Bouguet. Camera Calibration Toolbox for Matlab.

[Bou00] Jean-Yves Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm, 2000.

[BRFK12] Frederic Besse, Carsten Rother, Andrew W. Fitzgibbon, and Jan Kautz. PMBP: PatchMatch Belief Propagation for Correspondence Field Estimation. In *British Machine Vision Conference*, pages 1–11, 2012.

[BRR11] Michael Bleyer, Christoph Rhemann, and Carsten Rother. PatchMatch Stereo - Stereo Matching with Slanted Support Windows. In *British Machine Vision Conference*, pages 1–11, 2011.

[BSFG09] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B. Goldman. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM Transactions on Graphics*, 28(3):1–11, 2009.

[BSGF10] Connelly Barnes, Eli Shechtman, Dan B. Goldman, and Adam Finkelstein. The Generalized PatchMatch Correspondence Algorithm. In *European Conference on Computer Vision*, pages 29–43, 2010.

[BSL+11] Simon Baker, Daniel Scharstein, J.P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A Database and Evaluation Methodology for Optical Flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[BTG06] Herbert Bay, Tinne Tuytelaars, and Luc J. Van Gool. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision*, pages 404–417, 2006.

[BVZ01] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[BWS05] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/Kanade meets Horn/Schunck: Combining Local and Global Optic Flow Methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[CK02] Rodrigo L Carceroni and Kiriakos N Kutulakos. Multi-View Scene Capture by Surfel Sampling: From Video Streams to Non-Rigid 3D Motion, Shape and Reflectance. *International Journal of Computer Vision*, 49(2-3):175–214, 2002.

[DMG06] Frédéric Devernay, Diana Mateus, and Matthieu Guilbert. Multi-Camera Scene Flow by Tracking 3-D Points and Surfels. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2203–2212, 2006.

[DOIB12] Andrew Delong, Anton Osokin, Hossam N. Isack, and Yuri Boykov. Fast Approximate Energy Minimization with Label Costs. *International Journal of Computer Vision*, 96(1):1–27, 2012.

[DT05] James Diebel and Sebastian Thrun. An Application of Markov Random Fields to Range Sensing. In *Neural Information Processing Systems*, pages 1–8, 2005.

[FB81] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[FL11] Bill Freeman and Ce Liu. Markov Random Fields for Super-Resolution and Texture Synthesis. In *Advances in Markov Random Fields for Vision and Image Processing*. MIT Press, 2011.

[FRT97] Andrea Fusiello, Vito Roberto, and Emanuele Trucco. Efficient Stereo with Multiple Windowing. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 858–863, 1997.

[FW06] David J. Fleet and Yair Weiss. Optical Flow Estimation. In Nikos Paragios, Yunmei Chen, and Olivier Faugeras, editors, *Handbook of Mathematical Models in Computer Vision*. Springer Science+Business Media, Inc., New York, 2006.

[GBI09] Daniel Glasner, Shai Bagon, and Michal Irani. Super-Resolution from a Single Image. In *International Conference on Computer Vision*, pages 349–356, 2009.

[GYWG07] Minglun Gong, Ruigang Yang, Liang Wang, and Mingwei Gong. A Performance Study on Different Cost Aggregation Approaches used in Real-time Stereo Matching. *International Journal of Computer Vision*, 75(2):283–296, 2007.

[HB13] Simon Hadfield and Richard Bowden. Scene Particles: Unregularized Particle Based Scene Flow Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):564–576, 2013.

[HBK+14] Michael Hornáček, Frederic Besse, Jan Kautz, Andrew Fitzgibbon, and Carsten Rother. Highly Overparameterized Optical Flow Using PatchMatch Belief Propagation. In *European Conference on Computer Vision*, pages 220–234, 2014.

[HD07] Frédéric Huguet and Frédéric Devernay. A Variational Method for Scene Flow Estimation from Stereo Sequences. In *International Conference on Computer Vision*, pages 1–7, 2007.

[HFR14] Michael Hornáček, Andrew Fitzgibbon, and Carsten Rother. SphereFlow: 6 DoF Scene Flow from RGB-D Pairs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3526–3533, 2014.

[HKJK13] Philipp Heise, Sebastian Klose, Brian Jensen, and Alois Knoll. PM-Huber: PatchMatch with Huber Regularization for Stereo Matching. In *International Conference on Computer Vision*, pages 2360–2367, 2013.

[Hor86] Berthold K.P. Horn. *Robot Vision*. MIT electrical engineering and computer science series. MIT Press, 1986.

[HRF13] Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D Flow: Dense 3-D Motion Estimation Using Color and Depth. In *IEEE International Conference on Robotics and Automation*, pages 2276–2282, 2013.

[HRGR13] Michael Hornáček, Christoph Rhemann, Margrit Gelautz, and Carsten Rother. Depth Super Resolution by Rigid Body Self-Similarity in 3D. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1123–1130, 2013.

[HS81] Berthold K.P. Horn and Brian G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 17(1-3):185–203, 1981.

[HS88] Chris Harris and Mike Stephens. A Combined Corner and Edge Detector. In *Alvey Vision Conference*, pages 1–6, 1988.

[HZ06] Andrew Harltey and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, 2nd edition, 2006.

[IKH+11] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard A. Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew J. Davison, and Andrew W. Fitzgibbon. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *ACM Symposium on User Interface Software and Technology*, pages 559–568, 2011.

[JAHH92] James R. Bergen, P. Anandan, Keith J. Hanna, and Rajesh Hingorani. Hierarchical Model-Based Motion Estimation. In *European Conference on Computer Vision*, pages 237–252, 1992.

[KCLU07] Johannes Kopf, Michael F. Cohen, Dani Lischinski, and Matthew Uyttendaele. Joint Bilateral Upsampling. *ACM Transactions on Graphics*, 26(3):96, 2007.

[Kol06] Vladimir Kolmogorov. Convergent Tree-Reweighted Message Passing for Energy Minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

[KPS11] Rajkumar Kothapa, Jason Pacheco, and Erik B. Sudderth. Max-Product Particle Belief Propagation. Technical report, 2011.

[KS04] Yan Ke and Rahul Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 506–513, 2004.

[KSC01] Sing Bing Kang, Richard Szeliski, and Jinxiang Chai. Handling Occlusions in Dense Multi-view Stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 103–110, 2001.

[KZ04] Vladimir Kolmogorov and Ramin Zabih. What Energy Functions Can Be Minimized via Graph Cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

[LK81] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[Low04] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[LPB11] Antoine Letouzey, Benjamin Petit, and Edmond Boyer. Scene Flow from Depth and Color Images. In *British Machine Vision Conference*, pages 1–11, 2011.

[LRR08] Victor S. Lempitsky, Stefan Roth, and Carsten Rother. FusionFlow: Discrete-continuous optimization for optical flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[LRRB10] Victor S. Lempitsky, Carsten Rother, Stefan Roth, and Andrew Blake. Fusion Moves for Markov Random Field Optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1392–1405, 2010.

[LZ06] Gang Li and Steven W. Zucker. Surface Geometric Constraints for Stereo in Belief Propagation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2355–2362. IEEE Computer Society, 2006.

[MCNB12] Oisin Mac Aodha, Neill D.F. Campbell, Arun Nair, and Gabriel J. Brostow. Patch Based Synthesis for Single Depth Image Super-Resolution. In *European Conference on Computer Vision*, pages 71–84, 2012.

[MCUP04] Jiří Matas, Ondřej Chum, Martin Urban, and Tomáš Pajdla. Robust Wide-Baseline Stereo from Maximally Stable Extremal Regions. *Image and Vision Computing*, 22(10):761–767, 2004.

[MHPB13] Oisin Mac Aodha, Ahmad Humayun, Marc Pollefeys, and Gabriel J. Brostow. Learning a Confidence Measure for Optical Flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5):1107–1120, 2013.

[ML14] Marius Muja and David G. Lowe. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, 2014.

[MMP87] Jose L. Marroquin, Sanjoy K. Mitter, and Tomaso Poggio. Probabilistic Solution of Ill-Posed Problems in Computational Vision. Technical report, Cambridge, MA, USA, 1987.

[MOC02] Jiří Matas, Štěpán Obdržálek, and Ondřej Chum. Local Affine Frames for Wide-Baseline Stereo. In *International Conference on Pattern Recognition*, pages 363–366, 2002.

[MS01] Krystian Mikolajczyk and Cordelia Schmid. Indexing Based on Scale Invariant Interest Points. In *International Conference on Computer Vision*, pages 525–531, 2001.

[MS02] Kristian Mikolajczyk and Cordelia Schmid. An Affine Invariant Interest Point Detector. In *European Conference on Computer Vision*, pages 128–142. Springer-Verlag, 2002.

[MS04a] Krystian Mikolajczyk and Cordelia Schmid. Scale & Affine Invariant Interest Point Detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[MS04b] Lionel Moisan and Bérenger Stival. A Probabilistic Criterion to Detect Rigid Point Matches Between Two Images and Estimate the Fundamental Matrix. *International Journal of Computer Vision*, 57(3):201–218, 2004.

[MS06] Dongbo Min and Kwanghoon Sohn. Edge-Preserving Simultaneous Joint Motion-Disparity Estimation. In *International Conference on Pattern Recognition*, pages 74–77, 2006.

[MSKS03] Yi Ma, Stefano Soatto, Jana Košecká, and Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, New York, 2003.

[MTS+05] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiří Matas, Frederik Schaffalitzky, Timor Kadir, and Luc J. Van Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 65(1-2):43–72, 2005.

[MY09] Jean-Michel Morel and Guoshen Yu. ASIFT: A New Framework for Fully Affine Invariant Image Comparison. *SIAM Journal of Imaging Sciences*, 2(2):438–469, April 2009.

[NBK08] Tal Nir, Alfred M. Bruckstein, and Ron Kimmel. Over-Parameterized Variational Optical Flow. *International Journal of Computer Vision*, 76(2):205–216, 2008.

[Nis04] David Nistér. An Efficient Solution to the Five-Point Relative Pose Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–777, 2004.

[NL11] Sebastian Nowozin and Christoph H. Lampert. Structured Learning and Prediction in Computer Vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3-4):185–365, 2011.

[PBB+06] Nils Papenberg, Andrés Bruhn, Thomas Brox, Stephan Didas, and Joachim Weickert. Highly Accurate Optic Flow Computation with Theoretically Justified Warping. *International Journal of Computer Vision*, 67(2):141–158, April 2006.

[PD73] David H. Peucker and Thomas K. Douglas. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. *Cartographica*, 10(2):112–122, 1973.

[PKF07] Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. Multi-View Stereo Reconstruction and Scene Flow Estimation with a Global Image-based Matching Score. *International Journal of Computer Vision*, 72(2):179–193, 2007.

[PKG99] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. Self-calibration and Metric Reconstruction in Spite of Varying and Unknown Internal Camera Parameters. *International Conference on Computer Vision*, pages 90–95, 1999.

[PKT+11] Jaesik Park, Hyeongwoo Kim, Yu-Wing Tai, Michael S. Brown, and In-So Kweon. High Quality Depth Map Upsampling for 3D-TOF Cameras. In *International Conference on Computer Vision*, pages 1623–1630, 2011.

[QDC13] Julian Quiroga, Frédéric Devernay, and James L. Crowley. Local/Global Scene Flow Estimation. In *International Conference on Image Processing*, pages 3850–3854, 2013.

[RL01] Szymon Rusinkiewicz and Marc Levoy. Efficient Variants of the ICP Algorithm. In *International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001.

[SBM95] Malcolm Sambridge, Jean Braun, and Herbert McQueen. Geophysical Parametrization and Interpolation of Irregular Data using Natural Neighbours. *Geophysical Journal International*, 122(3):837–857, 1995.

[SRB10] Deqing Sun, Stefan Roth, and Michael J. Black. Secrets of Optical Flow Estimation and their Principles. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2432–2439, 2010.

[SS02] Daniel Scharstein and Richard Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

[SS03] Daniel Scharstein and Richard Szeliski. High-Accuracy Stereo Depth Maps Using Structured Light. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 195–202, 2003.

[SSZ03] Jian Sun, Heung-Yeung Shum, and Nan-Ning Zheng. Stereo Matching Using Belief Propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.

[STDT09] Sebastian Schuon, Christian Theobalt, James Davis, and Sebastian Thrun. LidarBoost: Depth Superresolution for ToF 3D Shape Scanning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 343–350, 2009.

[Sze11] Richard Szeliski. *Computer Vision - Algorithms and Applications*. Texts in Computer Science. Springer, 2011.

[TM11] Jing Tian and Kai-Kuang Ma. A Survey on Super-Resolution Imaging. *Signal, Image and Video Processing*, 5(3):329–342, 2011.

[TMdSA08] Federico Tombari, Stefano Mattoccia, Luigi di Stefano, and Elisa Addimanda. Classification and Evaluation of Cost Aggregation Methods for Stereo Correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[TPCB08] Werner Trobin, Thomas Pock, Daniel Cremers, and Horst Bischof. An Unbiased Second-Order Prior for High-Accuracy Motion Estimation. In *Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM)*, volume 5096, pages 396–405, 2008.

[VBR+05] Sundar Vedula, Simon Baker, Peter Rander, Robert T. Collins, and Takeo Kanade. Three-Dimensional Scene Flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):475–480, 2005.

[VBW08] Levi Valgaerts, Andrés Bruhn, and Joachim Weickert. A Variational Model for the Joint Recovery of the Fundamental Matrix and the Optical Flow. In *Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM)*, pages 314–324, 2008.

[Vek03] Olga Veksler. Fast Variable Window for Stereo Correspondence Using Integral Images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 556–564, 2003.

[vO06] J.D. van Ouwerkerk. Image Super-Resolution Survey. *Image and Vision Computing*, 24(10):1039–1052, 2006.

[VRS13] Christoph Vogel, Stefan Roth, and Konrad Schindler. An Evaluation of Data Costs for Optical Flow. In *German Conference on Pattern Recognition*, pages 343–353, 2013.

[VSR11] Christian Vogel, Konrad Schindler, and Stefan Roth. 3D Scene Flow Estimation with a Rigid Motion Prior. In *International Conference on Computer Vision*, pages 1291–1298, 2011.

[VSR13] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise Rigid Scene Flow. In *International Conference on Computer Vision*, pages 1377–1384, 2013.

[vSRH11] Jan Čech, Jordi Sanchez-Riera, and Radu Horaud. Scene Flow Estimation by Growing Correspondence Seeds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3129–3136, 2011.

[WPB+08] Andreas Wedel, Thomas Pock, Jürgen Braun, Uwe Franke, and Daniel Cremers. Duality TV-L1 Flow with Fundamental Matrix Prior. In *Image and Vision Computing*, pages 1–6, November 2008.

[WRV+08] Andreas Wedel, Clemens Rabe, Tobi Vaudrey, Thomas Brox, Uwe Franke, and Daniel Cremers. Efficient Dense Scene Flow from Sparse or Dense Stereo Data. In *European Conference on Computer Vision*, pages 739–751, 2008.

[WTRF09] Oliver Woodford, Philip Torr, Ian Reid, and Andrew W. Fitzgibbon. Global Stereo Reconstruction Under Second-Order Smoothness Priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2115–2128, 2009.

[XJM12] Li Xu, Jiaya Jia, and Yasuyuki Matsushita. Motion Detail Preserving Optical Flow Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1744–1757, 2012.

[YFW00] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Generalized Belief Propagation. In *Neural Information Processing Systems*, pages 689–695. MIT Press, 2000.

[YK05] Kuk Jin Yoon and In So Kweon. Locally Adaptive Support-Weight Approach for Visual Correspondence Search. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 924–931, 2005.

[YWHM10] Jianchao Yang, John Wright, Thomas S. Huang, and Yi Ma. Image Super-Resolution Via Sparse Representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, 2010.

[YYDN07] Qingxiong Yang, Ruigang Yang, James Davis, and David Nistér. Spatial-Depth Super Resolution for Range Images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[ZK01] Ye Zhang and Chandra Kambhamettu. On 3D Scene Flow and Structure Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 778–785, 2001.

[ZPB07] Christopher Zach, Thomas Pock, and Horst Bischof. A Duality Based Approach for Realtime TV-L1 Optical Flow. In *Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM)*, pages 214–223, 2007.

Michael Hornáček

Schönbrunnerstr. 81/16
1050 Vienna, Austria

birth date

25/7/1984 (Montréal, Québec, Canada)

citizenship

Canadian and Czech

education

2008-2010 **TU Vienna**, *Vienna, Austria*

Dipl.-Ing. (M.Sc.), computer science

Thesis: *Extracting Vanishing Points across Multiple Views*

2004-2008 **McGill University**, *Montréal, Québec, Canada*

BA, computer science (major) and linguistics (minor)

2001-2004 **John Abbott College**, *Ste-Anne-de-Bellevue, Québec, Canada*

DEC (Diploma of Collegiate Studies), Liberal Arts

employment

1/10/2011-31/10/2014 **Institute of Software Technology and Interactive Systems, TU Vienna**, *Vienna, Austria*

Research Assistant (Projektassistent)

Work towards a Ph.D. with focus in computer vision, funded by Microsoft Research through its European Ph.D. scholarship programme and co-supervised by Carsten Rother of Microsoft Research Cambridge (later TU Dresden, Germany)

- 3 month research stay at the Computer Vision Lab (CVLD) at TU Dresden (March-May 2014)
- Teacher's assistant (TA) duties

1/11/2010-30/9/2011 **Institute of Photogrammetry and Remote Sensing, TU Vienna**, *Vienna, Austria*

Research Assistant (Universitätsassistent)

Responsible for coordination and software development for the institute's involvement in a European Space Agency (ESA)-funded project in remote sensing for land surface hydrology using SAR data from the two-satellite Sentinel-1 constellation

- Published main results as a JSTARS journal paper (acceptance rate: 30-40%)
- Software development in IDL and Java for other ESA-funded projects

Teacher's assistant (TA) duties

1/10/2009-29/10/2010

VRVis Research Center, *Vienna, Austria*

Researcher

Research and software development work in the field of computer vision

- Work on master's thesis, the main results of which were published as an identically titled CVPR paper
- Work on computer vision-related extensions to the company's C# computer vision/graphics framework, with focus on multiple-view 3D scene recovery

28/4/2008-5/9/2008

McGill University, *Montréal, Québec, Canada*

Network Monitoring Assistant

12/3/2007-19/12/2007

Maintenance and development of the SNIPS and MRTG-based network monitoring systems for McGill University's Network Infrastructure group

- Rebuilt and streamlined the systems, reducing costs and increasing service robustness
- Developed several modules for McGill's custom MRTG system, including a billing mechanism in Perl and a time series snapshot (archiving) facility in bash

Ericsson Research Canada, *Town of Mount Royal, Québec, Canada*

2/1/2008-25/4/2008

Intern (Software Development)

- Wrote plugin to Maven in Java for Ericsson's Software Configuration Management (SCM) group

Winter 2005

McGill University, *Montréal, Québec, Canada*

Translator

Translations from Czech to English of communist-era Czechoslovak government documents for history professor Lorenz Lüthi's book, *The Sino-Soviet Split* (Princeton University Press, 2008)

publications

M. Hornáček, F. Besse, J. Kautz, A. Fitzgibbon, C. Rother, "Highly Overparameterized Optical Flow Using PatchMatch Belief Propagation," European Conference on Computer Vision (ECCV), 2014

M. Hornáček, A. Fitzgibbon, C. Rother, "SphereFlow: 6 DoF Scene Flow from RGB-D Pairs," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014

M. Hornáček, C. Rhemann, M. Gelautz, C. Rother, "Depth Super Resolution by Rigid Body Self-Similarity in 3D," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013

W. Wagner, D. Sabel, M. Doubková, **M. Hornáček**, S. Schlaffer, A. Bartsch, "Prospects of Sentinel-1 for Land Applications," IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2012

M. Hornáček, W. Wagner, D. Sabel, H.-L. Truong, P. Snoeij, T. Hahmann, E. Diedrich, M. Doubková, "Potential for High Resolution Systematic Global Surface Soil Moisture Retrieval via Change Detection Using Sentinel-1," IEEE Journal of Special Topics in Earth Observations and Remote Sensing (JSTARS), 2012

M. Hornáček and S. Maierhofer, "Extracting Vanishing Points across Multiple Views," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011

selected talks

"SphereFlow: 6 DoF Scene Flow from RGB-D Pairs," EU-Korea Conference on Science and Technology (EKC), Vienna, Austria, July 24, 2014 (invited talk); 38th Annual Workshop of the Austrian Association for Pattern Recognition (ÖAGM), Institute of Science and Technology (IST), Klosterneuburg, Austria, May 22, 2014 (invited talk); 34th Pattern Recognition and Computer Vision Colloquium, Czech Technical University, Prague, Czech Republic, April 3, 2014 (invited talk)

"Depth Super Resolution by Rigid Body Self-Similarity in 3D," VRVis Research Center, Vienna, Austria, July 25, 2013; Centre for Intelligent Machines (CIM), McGill University, Montréal, Québec, Canada, June 20, 2013; University College London, London, United Kingdom, June 5, 2013

"GNSS Positioning in Support of Surface Soil Moisture Retrieval and Flood Delineation in Near Real Time," United Nations/Latvia Workshop on the Applications of Global Navigation Satellite Systems, Riga, Latvia, May 17, 2012

teaching

TA for "Stereo Vision," spring semester of 2013

TA for seminars "Seminar on Computer Vision and Pattern Recognition," "Seminar on Image and Video Analysis and Synthesis", and "Seminar on Media Informatics," multiple semesters 2011-2014

TA for "Microwave Remote Sensing," spring semester of 2011

reviewing

JSTARS, IJCV (2014); ISPA, JSTARS, CVWW (2013); DAGM-ÖAGM (2012); CJRS (2011)

awards

Travel grants from Microsoft Research Cambridge to attend CVPR 2013 in Portland, Oregon, USA; CVPR 2014 in Columbus, Ohio, USA; ECCV 2014 in Zürich, Switzerland

Travel grant from the United Nations Office for Outer Space Affairs (UNOOSA) to attend the 2012 United Nations/Latvia Workshop on the Applications of Global Navigation Satellite Systems in Riga, Latvia

Ph.D. scholarship from Microsoft Research Cambridge (2011-2014)

Dipl.-Ing. (M.Sc.) with distinction (2010)

IT skills

C/C++, C#, Java, Perl, bash, MATLAB, IDL; OpenCV, OpenGL, PCL

languages

Fluent English, German, Slovak; intermediate Czech, Polish, French