Reconstruction of Dense Correspondences

Martin Eisemann, Jan-Michael Frahm,
Yannick Remion, Muhannad Ismael

8.1 Introduction

This chapter concentrates on dense image correspondence estimation with a special focus on stereo. Images are the basic input for a vast majority of algorithms dealing with the reconstruction of the real world. To analyze a scene from a collection of images it becomes inevitable to put these images into correspondence. These correspondences then form the basis for many subsequent analyses, including camera calibration, stereo and 3D reconstruction, motion information, scene flow and others. While some of these tasks like camera calibration require only sparse correspondences between the images, Chapter 7, others require per-pixel correspondence, also known as dense correspondence estimation.

Humans are extremely good at solving the correspondence problem which most of them do all the time during depth perception. Basically, the eyes serve as two cameras, slightly displaced, with respect to each other, that capture the surrounding from two different viewpoints. When focusing on an object at a certain distance one has already computed an estimate of the distance in the brain and therefore of the object's position in space. It turns out the same problem is quite difficult for a computer and has been researched for several decades now.

The difficulty in correspondence estimation is caused by several factors: images are often corrupted by sensor noise, e.g. when recorded in a poorly lit environment Section 1.1; the captured scene signal is discretized and represented by some finite image resolution; not every pixel actually has a correspondencing partner in the other views as it might be occluded; and ambiguities due to the absence of texture are difficult to solve.

If one can solve the dense correspondence problem a variety of different applications becomes possible especially in the field of computer vision. Robot navigation and autonomous cars require depth perception to avoid obstacles [Giachetti et al. 98, Kastrinaki et al. 03]. Quality assurance in industrial applications is often based on stereo algorithms to detect cracks

and ridges in manufactured products. Reconstruction of urban environments from images has recently gained a lot of interest in the research community [Gallup et al. 07, Frahm et al. 10]. The dense correspondences allow for video editing [Adobe Systems Inc. 13, The Foundry 13], super-resolution [Irani and Peleg 91], video stabilization [Matsushita et al. 06], to interpolate between images [Chen 95, Lipski et al. 10a], e.g. to create bullet time effects made famous in the blockbuster movie *The Matrix* and for specific tracking applications in graphics, e.g. the local pose optimization for texture correspondence matching in Chapter 11 is related. Disparity remapping based on the correspondences and reconstructed depth becomes important to avoid visual fatigue in stereoscopic cinema [Devernay and Beardsley 10].

The following will give a hands-on guide on how to compute dense correspondences between images. After a short overview of current state-of-the-art approaches, Section 8.2, a robust solution to the correspondence problem is described and extended, Section 8.3. It is described how to compute correspondences from multiple images, Section 8.4, and means to speed up the computations using graphics hardware are presented, Section 8.5.

8.2 Overview

This section gives a brief overview of different approaches dealing with the dense correspondence problem. The goal is to find the best corresponding (sub-)pixel position in neighboring views for every pixel of a reference image, if such corresponding positions exist.

The algorithms dealing with the correspondence problem can be broadly classified into two categories: stereo and optical flow. Intrinsically the problem is the same for both of them, finding good correspondences between the views, but they differ in the premises. Stereo can be seen as a special case of optical flow, where correspondences are searched along the same scanline (or epipolar line), reducing the solution space from 2D to 1D. The following will give a short overview of the most seminal papers in both categories and their contributions.

Stereo In analogy to the human eyes, the input to classic binocular stereo algorithms are two images $\mathbf{I}^l$ and $\mathbf{I}^r$, a left and a right one. The task is to find for every pixel $\mathbf{p}$ with pixel coordinates (x, y) in the left image a corresponding pixel $\mathbf{q}$ in the right view with pixel coordinates $(x - d_{\mathbf{p}}, y)$. $d_{\mathbf{p}}$ is called the disparity of pixel $\mathbf{p}$. The disparity information is typically saved in an intensity image, the so-called disparity map $\mathbf{D}$, where low/dark

Figure 8.1: Dense correspondence estimation in stereo for the Middlebury Tsukuba data set [Scharstein and Szeliski 02]. (a) The task is to find for each pixel at any position (x, y) in the left view (b) a corresponding position $(x - d, y)$ in the right view and encode the result in (c) a disparity map from which 3D coordinates can be reconstructed. In the stereo setting, the corresponding pixels lie on the same scanline, whereas in the more general problem of optical flow estimation the correspondence can be any position within the right view. Instead of comparing single pixel values, comparing neighborhoods of pixels (shown as the overlaid grid) results in higher robustness.

values encode low disparity and high/bright values encode high disparity, Fig. 8.1(c).

In stereo one generally distinguishes between local and global methods. In the first category local areas of one image are matched to local areas in the corresponding view, often called support regions. The difficulty lies in the choice of the support region as matching single pixels is highly ambiguous in most scenes. Simple rectangular windows around the pixel under consideration can be efficiently implemented [Hirschmüller et al. 02, Mühlmann et al. 02] but it can be difficult to choose the right size. By shifting the center position of the window and testing different sizes [Fusiello et al. 97] or by deactivating parts of the support region [Hirschmüller et al. 02, Veksler 02] one can hope that at least one constellation does not overlap with a depth discontinuity. This otherwise poses a matching problem as in many cases a depth discontinuity marks the separation line between two objects with different disparities and, therefore, a different amount of motion in image space from the left to the right view. The research community has thus investigated methods to find a good support region, with different criteria on how much influence each pixel inside this region should have on the final result [Hosni et al. 13].

One key component and a breakthrough for local methods in recent years have been the introduction of adaptive support weights [Yoon and Kweon 05]. The idea is to adjust the influence of neighboring pixels on the

final matching cost based on a similarity metric, most often color and spatial similarity. [Yoon and Kweon 05]'s bilateral weighting scheme is based on a Gaussian distribution depending on the spatial proximity and proximity of intensity values. To overcome the problem of spatially close but distinct objects influencing each other, the spatial proximity can be exchanged with a geodesic distance [Hosni et al. 09].

Unfortunately, the computation of adaptive support weights is costly if implemented in a naive way. To speed up the aggregation step it can be converted to an image filtering procedure. It turned out that the bilateral weighting scheme of [Yoon and Kweon 05] is equivalent to applying a cross-bilateral filter or derivations of it to the x, y-slices of a cost volume [Hosni et al. 11b, Richardt et al. 10, Zhang et al. 10a, Ju and Kang 09]. To further speed up the computation, the pixel-wise matching for fixed disparities can be elegantly formulated as a plane-sweeping algorithm on the GPU [Yang and Pollefeys 03, Gallup et al. 07, Zach et al. 08] allowing for real-time stereo implementations.

An implicit assumption made by the aforementioned techniques is that each local support region is basically a patch with fronto-parallel orientation to the image plane of the reference view. Treating the slices in the cost volume not as virtual planes representing a certain disparity but as real 3D planes in the scene one can easily use rotated versions of these slices to compute the matching cost for slanted surfaces [Gallup et al. 07]. The computation times, however, increase linearly with the number of orientations used. Therefore, [Zhang et al. 08] propose to iteratively refine the disparities and orientations in a feedback loop. Another alternative is to initialize each pixel with a random orientation and disparity and propagate good matches to neighboring pixels based on a PatchMatch update scheme [Bleyer et al. 11a].

The second category of stereo algorithms form the so-called global methods. Global stereo methods pose the matching problem as an energy minimization problem which is usually of the following form:

$$E(\mathbf{D}) = E_{\text{data}}(\mathbf{D}) + \alpha \cdot E_{\text{smooth}}(\mathbf{D}) \quad , \tag{8.1}$$

where $\mathbf{D}$ is the current estimate of the disparity map. The goal is to find $\mathbf{D}$ that produces the lowest energy. E_{data} in this context is a photo-consistency measure that can be equal to the matching function of the local methods but is traditionally simpler. Instead of implicitly stating a smoothness function in the form of a support region, as in the local approaches, here the smoothness is explicitly expressed within the error formulation as E_{smooth}. This regularization of the solution can be especially useful for textureless regions as it basically smoothes out the solution. Several optimization approaches have been proposed to minimize Eq.(8.1)

through dynamic programming [Veksler 05, Bleyer and Gelautz 08], graph-cuts [Boykov et al. 01, Hong and Chen 04, Bleyer and Gelautz 07] or belief propagation [Sun et al. 03, Yang et al. 06b, Taguchi et al. 08].

Interestingly, the usage of tree-reweighted message passing (TRW) and a comparison to ground truth results revealed that modern optimization algorithms yield energies that are actually lower than that of the ground truth solution [Szeliski et al. 08]. This indicates that the model in Eq.(8.1) is actually a limiting factor. Further advances, therefore, need to extend the model. Explicit occlusion handling or enforcing symmetrical matches between the input images was used, e.g., in [Kolmogorov and Zabih 01, Lin and Tomasi 04, Sun et al. 05, Woodford et al. 09]. Truncating the smoothness term to a user-defined maximum value favors large jumps in the disparity map instead of many small changes [Hirschmüller 05, Sun et al. 05, Yang et al. 06a]. Segmentation-based methods presegment the image into patches of coherent color and match whole segments at once [Deng et al. 05, Hong and Chen 04, Zitnick et al. 04]. The idea is that in many cases depth discontinuities coincide with segment borders. An extension of segmentation-based stereo is object-based stereo which matches semantic objects instead of single colored patches. In this way it becomes possible to handle even semi-occluded surfaces [Bleyer et al. 11b]. Extending the idea of object-based stereo one can estimate simple 3D approximations for the different objects [Bleyer et al. 12]. On the basis of these higher semantic concepts one can add sophisticated additional constraints to the optimization, for instance to prevent intersections between the objects or to add a gravity constraint.

Optical flow The problem of optical flow estimation is strongly related to the stereo problem and several of the aforementioned algorithms are applicable to both. Basically, optical flow estimation is a generalization of the stereo problem from a 1D solution space, the disparity map, to a 2D solution space, the flow or motion field. While stereo algorithms aim at reconstructing correspondences between images captured at the same instance in time, optical flow allows to track the motion of pixels also across the time dimension, e.g. in a video.

During the last 30 years, hundreds of research papers have been published in the field of optical flow and various surveys and benchmarks cover and compare the state-of-the-art [Barron et al. 94, Baker et al. 11]. The seminal work of [Horn and Schunck 81] and [Lucas and Kanade 81] laid the foundations for the algorithms to follow. Interestingly, similar to stereo, one can distinguish global and local approaches to the optical flow problem, explicitly enforcing smoothness in the solution [Horn and Schunck 81] and assuming local constancy within a window around each pixel [Lucas and

Kanade 81]. Neither assumption of smoothness holds at motion boundaries for which robust [Black and Anandan 96, Zach et al. 07] and anisotropic regularizers [Nagel and Enkelmann 86, Werlberger et al. 09, Sun et al. 10, Zimmer et al. 11] have therefore been proposed. To reduce the influence of outlier pixels caused by brightness changes and sensor noise the simple data terms based on color-constancy assumption are mostly replaced by robust penalizer functions [Black and Anandan 96, Brox et al. 04, Zach et al. 07] or pixel-descriptors [Mileva et al. 07, Liu et al. 08].

To cope with fast motion, scale-space approaches [Anandan 89] and iterative warping schemes [Alvarez et al. 00, Brox et al. 04] make use of image pyramids to find corresponding pixels. As downsampling works only well for sufficiently large objects several search schemes have been proposed in the literature that either perform a full search [Steinbrücker et al. 09, Linz et al. 10a, Lipski et al. 10b, Hosni et al. 11b] or use tracked features as reliable priors for the optimization [Brox and Malik 11]. In a more hardware-based approach [Lim et al. 05] make use of a high-speed camera to reduce the per pixel displacement to less than a pixel.

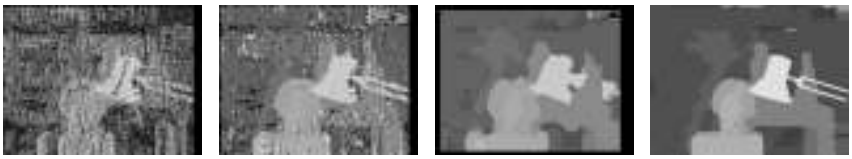
Probably due to its success in stereo, explicit occlusion handling has been introduced to optical flow estimation as well. The occlusion detection thereby is either based on the optimization residual and divergence of the flow [Xiao et al. 06, Sand and Teller 06], the symmetry of forward and backward flow [Alvarez et al. 07, Linz et al. 10a, Lipski et al. 10b], or is integrated in the image formation model making use of alternate exposure images [Sellent et al. 11] by alternate capturing of long- and short-exposed images in a video.

8.3 Dense Correspondence Estimation

In the following, an approach is described to compute dense correspondences between two images. The algorithm is mainly based on the fast cost-volume filtering by [Hosni et al. 11b] which is one of the top ranked local methods for stereo and which is also applicable to the more generalized optical flow problem[1]. For simplicity it is assumed that the images have already been rectified, i.e. corresponding points lie on the same scanline, Fig. 8.1. Otherwise, it is assumed that standard rectification algorithms are applied first [Hartley and Zisserman 03][2]. These are generally based on the camera registration procedures described in Chapter 7. These constraints will be loosened in the later part of this chapter (Section 8.5).

The basic task of estimating a disparity map $\mathbf{D}$ can be formulated for

[1] Code is available at https://www.ims.tuwien.ac.at/publications/tuw-210567
[2] Code is available at http://www.robots.ox.ac.uk:5000/~vgg/hzbook/code/.



(a) (b) (c) (d)

Figure 8.2: Different dissimilarity functions. (a) Pixel-wise matching solely based on color/intensity is highly ambiguous. (b) A 3×3 correlation window is still noisy. (c) A 21×21 correlation window results in edge fattening. (d) The cost filter method of [Hosni et al. 11b].

each pixel $\mathbf{p}$ as

$$d_{\mathbf{p}} = \underset{0 \leq d \leq d_{\max}}{\operatorname{argmin}} \, c(\mathbf{p}, \mathbf{p} - d) \ . \tag{8.2}$$

The term $d_{\max}$ is a user-defined constant which must be larger than the expected maximum disparity. Note that due to rectification d is always a positive value or 0. In case the disparity map for the right image is to be computed $\mathbf{p} - d$ in Eq.8.2 is replaced by $\mathbf{p} + d$. For simplicity, only disparity computations for the left image are considered. The simplified notation $\mathbf{p} - d$ denotes the pixel 2D pixel position $(x_p - d_p, y_p)$ where (x_p, y_p) are the pixel coordinates of pixel $\mathbf{p}$. The symbol c denotes a cost / dissimilarity function.

Dissimilarity functions To find an appropriate disparity $d_{\mathbf{p}}$ for each pixel $\mathbf{p}$ one needs to find a suitable dissimilarity function c in Eq.(8.2). The probably most simple one would be a naive per-pixel matching, that is, $c(\mathbf{p}, \mathbf{q}) = |\mathbf{I}^l_{\mathbf{p}} - \mathbf{I}^r_{\mathbf{q}}|_2$ where $\mathbf{I}^l_{\mathbf{p}}$ denotes the pixel intensity of $\mathbf{I}^l$ at pixel position $\mathbf{p}$ and $|\cdot|_2$ is the Euclidean distance between the two vectors. But matching only simple intensity values is highly ambiguous and leads to very noisy results, Fig. 8.2(a).

Instead of matching single pixels one can match small image patches centered at $\mathbf{p}$. In this case the cost function becomes

$$d_{\mathbf{p}} = \underset{0 \leq d \leq d_{\max}}{\operatorname{argmin}} \sum_{\mathbf{q} \in \mathbf{W}_{\mathbf{p}}} c(\mathbf{q}, \mathbf{q} - d) \ \ , \tag{8.3}$$

where $\mathbf{W}_{\mathbf{p}}$ is a square window centered at $\mathbf{p}$, and c as defined above. Figure 8.2(b) and 8.2(c) shows the resulting disparity maps using correlation windows of the size 3×3 and 21×21 pixels, respectively. The choice of a right size has a notable influence on algorithmic performance, and no single

window size generally works for all cases. While smaller window sizes capture finer details, matching scores can be highly ambiguous. Larger window sizes, on the other hand, lead to edge fattening around discontinuities and oversmooth results. What is needed is an adaptive support weight that adjusts the shape of the window or, in other words, reduces the influence of pixels that do not belong to the same object as pixel $\mathbf{p}$.

Adaptive support weights Adaptive support weights adjust the influence of each individual pixel considered in the matching process. This can be formulated as a simple extension to Eq.(8.3)

$$d_{\mathbf{p}} = \underset{0 \leq d \leq d_{\max}}{\operatorname{argmin}} \sum_{\mathbf{q} \in \mathbf{W_p}} w(\mathbf{p}, \mathbf{q}) \cdot c(\mathbf{q}, \mathbf{q} - d) \quad , \tag{8.4}$$

where $w(\mathbf{p}, \mathbf{q})$ is a weighting function which should return a value of 1 if $\mathbf{q}$ has the same disparity as $\mathbf{p}$, and 0 otherwise. As this disparity is not known, the weight is usually based on some heuristic that represents the probability that pixel $\mathbf{q}$ exhibits the same disparity as $\mathbf{p}$. The most common assumption made is that pixels close to $\mathbf{p}$ are more likely to belong to the same object and have a more similar disparity than pixels farther away. Additionally, pixels with similar color are more likely to belong to the same object than those with dissimilar color values. The bilateral weighting scheme proposed in [Yoon and Kweon 05] expresses this correlation as

$$w_b(\mathbf{p}, \mathbf{q}) = exp\left(-\left(\frac{c_c(\mathbf{p}, \mathbf{q})}{\sigma_c} + \frac{c_s(\mathbf{p}, \mathbf{q})}{\sigma_s}\right)\right) \quad . \tag{8.5}$$

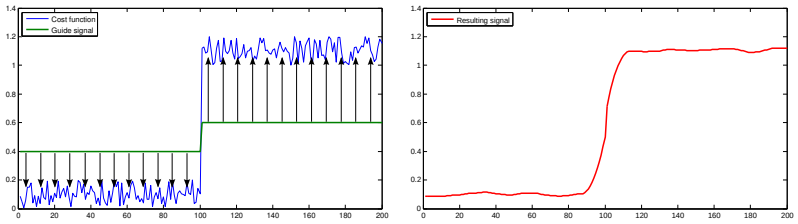
The function $c_c(\mathbf{p}, \mathbf{q})$ denotes the similarity in color defined as the Euclidean distance of pixels at position $\mathbf{p}$ and $\mathbf{q}$ in RGB space, whereas $c_s(\mathbf{p}, \mathbf{q})$ is the spatial component defined as the Euclidean distance of $\mathbf{p}$ and $\mathbf{q}$'s pixel coordinates. The terms σ_c and σ_s are user-defined constants that control the spread of each term similar to the window size before.

The computation of the bilateral weights for each pixel in the input image is time consuming. A fast and qualitatively even better alternative to the bilateral weighting scheme in Eq.(8.5) is the guided image filter [He et al. 10][3]. While the output is similar to the bilateral weighting, the computation is different

$$w_g(\mathbf{p}, \mathbf{q}) = \frac{1}{|\mathbf{W}|} \sum_{\mathbf{k}:(\mathbf{p},\mathbf{q}) \in \mathbf{W_k}} (1 + (\mathbf{I}^l_{\mathbf{p}} - \mu_{\mathbf{k}})^\top (\Sigma_{\mathbf{k}} + \epsilon \mathbf{U})^{-1} (\mathbf{I}^l_{\mathbf{q}} - \mu_{\mathbf{k}})) \quad , \tag{8.6}$$

with $\mu_{\mathbf{k}}$ and $\Sigma_{\mathbf{k}}$ being the mean vector and covariance of $\mathbf{I}^l$ in a squared window $\mathbf{W_k}$ of user-defined size, centered at and being constant for each

[3] Code is available at http://research.microsoft.com/en-us/um/people/kahe/eccv10/



(a) (b)

Figure 8.3: 1D example of the guided image filter [He et al. 10] for a 1D signal. (a) The filter takes a guide signal (green) and fits it locally to the given, potentially noisy, cost function (blue) resulting in (b) a smoothed but edge preserving signal (red).

pixel $\mathbf{k}$. $|\mathbf{W}|$ denotes the number of pixels in the window. $\mathbf{U}$ is the identity matrix and ϵ a smoothness parameter. While, at first glance, Eq.(8.6) appears highly complex in comparison to Eq.(8.5), it turns out that the computation requires only running a series of box filters which can be computed in constant time, independent of the window size. For details see [He et al. 10].

Intuitively, the guided image filter takes a guide image, in this case the input image $\mathbf{I}^l$, and tries to fit it locally to the cost function $\mathbf{C}^d$ that encodes pixelwise costs for a certain disparity d.

$$\mathbf{C}^d_{\mathbf{p}} = c(\mathbf{p}, \mathbf{p} - d) = |\mathbf{I}^l_{\mathbf{p}} - \mathbf{I}^r_{\mathbf{p}-d}| \quad ,$$

This results in a smoothed version of $\mathbf{C}^d$ which is equal to aggregating the weighted costs in a given window around each pixel. For this, the best fitting linear transformation for local windows $\mathbf{W_k}$ around each pixel is computed, i.e., a scaling and an offset of the guide image, to get from $\mathbf{I}^l$ to $\mathbf{C}^d$. In a second step the linear transformation coefficients of all windows overlapping at a pixel are averaged. An example for a single-channel input is given in Fig. 8.3. Other commonly used matching techniques and pixel descriptors can also be found in Chapter 7.

Cost volume filtering Stacking the functions $\mathbf{C}^d$ for all disparities d onto each other into a 3D array $\mathbf{C}$ creates the so-called cost-volume. The filtered cost-volume can be extracted by filtering each x, y-slice that belongs to a fixed disparity d with the guided image filter as described above. The final disparity for each pixel $\mathbf{p}$ is then defined in Winner-Takes-All manner as

$$d_{\mathbf{p}} = \underset{0 \leq d \leq d_{\max}}{\operatorname{argmin}} \mathbf{C}^d(\mathbf{p}) \quad .$$

Occlusion Occluded pixels are detected using a left-right cross checking procedure. Once the disparities for image $\mathbf{I}^l$ to image $\mathbf{I}^r$ are computed, one can exchange both images and additionally compute the disparities from $\mathbf{I}^r$ to $\mathbf{I}^l$. A pixel $\mathbf{p}$ is marked as invalid, i.e. occluded, if $d^l_{\mathbf{p}} \neq d^r_{\mathbf{p}-d_{\mathbf{p}}}$ where $d^l_{\mathbf{p}}$ is the disparity at pixel $\mathbf{p}$ with reference image $\mathbf{I}^l$. Again note that the disparity is always positive and the sign in Eq.(8.2) is changed according to whether the disparity for the left or right image is computed.

One cannot assign disparities to pixels being occluded in one of the input images. If the application demands such an assignment, it has to be based on some kind of sensible heuristic. In [Hosni et al. 11b] a weighted median filter is used for filling invalidated pixels.

Extensions An advantage of the presented framework is that it naturally extends to higher dimensional and more fine-grained solution spaces at the cost of higher computation times. In the previous example each slice in the cost volume corresponds to a certain integer-valued disparity. One can easily increase the precision to fractional values by increasing the number of slices and assigning each slice to a certain fractional disparity. More generally speaking, each slice of the cost volume can be considered to be a distinct label l from a set $\mathcal{L} = \{1, \ldots, L\}$. The user only needs to specify how these labels are mapped to semantically meaningful parameters for the algorithm. That means one is not bound to interpret l only as integer-valued disparities but could extend the label space to fractional disparities as well, e.g. [Gehrig et al. 12]. Alternatively, a set of slanted windows could be included to better handle slanted surfaces that are not fronto-parallel, e.g. [Gallup et al. 07].

By defining a mapping from a 2D solution vector (u, v) to the label space $\mathcal{L}$ one can directly extend the presented stereo approach to optical flow problems by exchanging $d_{\mathbf{p}}$ and d in Eq.(8.2) by $(u_{\mathbf{p}}, v_{\mathbf{p}})$ and (u, v), respectively. In this context it should be noted that modern optical flow methods mostly use a more sophisticated cost function including not only color- but also gradient-similarity, details for the presented approach can be found in [Hosni et al. 11b].

Limitations A principal limitation of all local methods, such as the one presented, is their inability to cope with highly ambiguous data such as unicolored walls or objects. Depending on the application this may not be crucial, e.g., for image interpolation, as no visible artifacts will occur if objects of the same color are interpolated incorrectly. In other applications, such as autonomous driving vehicles or robot navigation, this may pose a high risk, because there, accurate disparity, which means accurate depth, is crucial. Imagine similar stone pillars standing next to each other. Matching

the right ones is highly ambiguous. In such cases more complicated global correspondence estimation algorithms are required, a good overview is given in [Bleyer and Breiteneder 13].

Another limitation of the presented problem formulation is it's discrete nature, which means it can only produce a solution that consists of combinations of preset labels. Even though labels may represent fractional values and the solution is therefore sub-pixel precise, it is always limited by the label space. The quality of any correspondence algorithm also depends highly on the scene content. While local methods are ranked high in the famous Middlebury benchmark [Scharstein and Szeliski 02], they are not always as successful in other benchmarks, e.g. [Geiger 12]. The reason could be a higher sensitivity to noise or ambiguities occurring more often in natural scenes. And finally occlusion handling can usually not be integrated into the matching process directly with local methods. Once the cost-volume has been created one could exchange the Winner-Takes-All strategy by a more sophisticated global label selection algorithm that could handle such cases by a better or more robust disparity assignment even for pixels occluded in one view. Therefore, the presented algorithm is a good starting point for further investigation of dense correspondence algorithms.

Section 8.4 extends the stereo correspondence estimation to multiple input cameras and deals with appropriate camera layouts and scene representations. Section 8.5 describes the plane-sweeping stereo algorithm that is easiy portable to the graphics card to allow even real-time correspondence estimation.

8.4 Multi-View Stereo

The following section gives an overview of multi-view stereovision. The term multi-view stereovision (MVS) refers to stereovision-based reconstruction from $n > 2$ views, $\mathbf{I}^0$ to $\mathbf{I}^{n-1}$, and is sometimes called *multiocular stereovision* in contrast to *binocular stereovision* from one pair of views, Section 8.3. MVS has been an active field of research for several decades and more than seventy algorithms are listed on the Middlebury Multi-View Benchmark website [Seitz et al. 06]. This benchmark provides a commonly accepted test suite to evaluate the quality of multi-view stereo algorithms.

An important assumption of any MVS method lies in its required, compatible or intended camera layout since various possibilities exist and may have an impact on the 3D reconstruction strategy, Section 2.2.

Most of MVS methods (notably among those on the Middlebury list) are designed for n cameras freely laid out in space. Some apply binocular stereovision (as previously discussed in Sect. 8.3) on different couples of views $(\mathbf{I}^i, \mathbf{I}^j)$ and then merge their separate binocular results. The main

difficulty in such approaches concerns regularizing the union of separate results, especially in scene areas where reconstructions overlap. Common problems to be solved in such areas are to reduce too high point density and to resolve ambiguities/inconsistencies. This task pertains to point cloud merging and is discussed in Chapter 10 in more detail. Another type of MVS approach for a free camera layout consists in fitting some form of geometric model of the scene in order to maximize its local photo-consistency in available views [Furukawa and Ponce 10].

Some other MVS methods, sometimes called *multi-baseline stereovision* methods, are designed for the "parallel" or "decentered parallel" camera layouts discussed in Section 2.2 and especially fitted for 3DTV content capture. Those layouts are characterized by aligned, evenly distributed and parallel cameras. As will be demonstrated below, such restrained settings induce a set of geometrical constraints on corresponding pixels from different views arising from the so-called *simplified multi-epipolar geometry*. Those constraints enable searching correspondences over every view at once as a multi-view matching process [Okutomi and Kanade 93, Szeliski and Golland 99, Niquin et al. 10, Ismael et al. 14]. This is also called *multi-scopic stereo matching* and consists in matching n-tuples of pixels instead of couples which yields a consistent and more robust reconstruction.

In the following, the main concepts behind multi-baseline stereovision are reviewed. The "parallel" layout of Section 2.2 implies the optical centers $\mathbf{o}_0 \dots \mathbf{o}_{n-1}$ to be aligned and evenly distributed on the base line, parallel optical axes orthogonal to this base line, cameras of same focal and dark-room depth, sensors of same size and resolution $nc \times nl$ centered on their optical axes with rows parallel to the base line. The "decentered parallel" layout, Fig. 8.4, generalizes this setting by translating the sensor centers off their optical axis in such a way that the *line of sight* of all the views, defined for each camera by its sensor center and optical center, now converge on a chosen 3D *convergence point* $\mathbf{c}$, possibly at finite distance [Prévost et al. 13]. The convergence point is of utmost importance in 3DTV content shooting as it will be displayed exactly on the center of the 3D display and thus controls how captured scene space is mapped in the perceived 3D space, Section 2.2. One should note that, in this layout, the convergent lines of sight no longer coincide with the parallel optical axes. The off-axis translation of the "sensor" may be achieved both at hardware design stage as sensor chip physical/mechanical decentering and/or, to a given extent, at software post-processing stage as region of interest (ROI) cropping. In the following, the term sensor ROI is used to denote all of the above mentioned possibilities.

Another benefit of such a layout is usually achieved thanks to rectification of the n views from aligned and evenly distributed cameras with

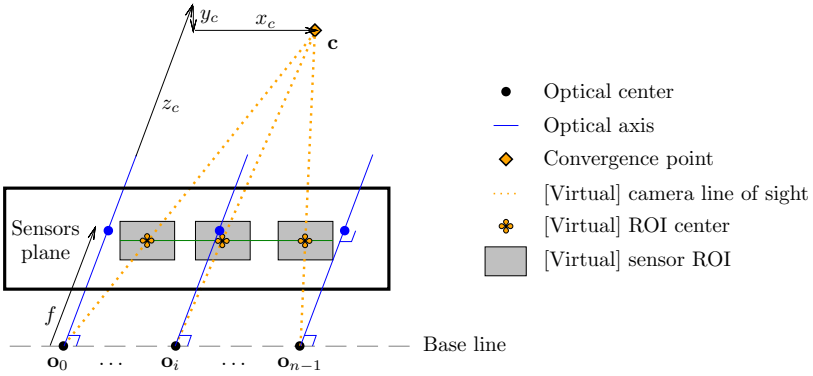


Figure 8.4: Decentered parallel camera layout.

(approximately) convergent optical axes. Similar to its binocular counterpart, the multiocular rectification consists in intersecting pixel rays by a plane at distance f (virtual sensors' plane) parallel to the common base line connecting the optical centers, Fig. 8.4. The rectified virtual sensor ROI in which the rectified views will be computed are then virtually laid in this sensors' plane with same size and orientation, so that the rows are parallel to the base line. Furthermore, their centers are chosen to make every line of sight converge at the chosen 3D convergence point $\mathbf{c} = (x_\mathbf{c}, y_\mathbf{c}, z_\mathbf{c})$ (coordinates expressed in reference frame of camera 0), Fig. 8.4. One should note that there is not as much freedom in the layout of the actual cameras as in the binocular case as the rectification process relies on actual optical centers being aligned and rather evenly distributed.

For n images $\mathbf{I}^i$ recorded or rectified in "(decentered) parallel" layout and numbered $i \in \{0, n-1\}$ from left to right, the *epipolar constraint*, previously discussed for the binocular case, Fig. 8.5, states that any pixel at $\mathbf{p}_i$ in any image $\mathbf{I}^i$ represents the actual 3D scene point projected onto $\mathbf{p}_i$. Pixel $\mathbf{p}_i$ and the optical center $\mathbf{o}_i$ of the camera are aligned on $\mathbf{p}_i$'s "pixel ray". Considering that pixel rays of corresponding pixels at $\mathbf{p}_i$ and $\mathbf{p}_j$ in two views $\mathbf{I}^i$, $\mathbf{I}^j$ have to intersect at their common 3D point $\mathbf{p}$, a straightforward derivation yields that optical centers $\mathbf{o}_i$, $\mathbf{o}_j$ and corresponding pixels at $\mathbf{p}_i$, $\mathbf{p}_j$ have to be coplanar (they belong to 2 intersecting and yet different lines). An *epipolar plane* is then defined for a couple of views (i, j) by both optical centers and any studied pixel in one of these views. The corresponding pixel in the other view has thus to be searched for within the *epipolar segment* defined as the intersection of this plane with the other

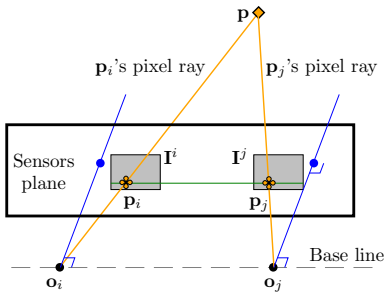


Figure 8.5: Simplified epipolar geometry.

image (black horizontal line in Fig. 8.5). When those two (rectified) cameras are set in "(decentered) parallel" layout, the epipolar segment in $\mathbf{I}^j$ defined by pixel $\mathbf{p}_i$ in $\mathbf{I}^i$ is part of the scanline of $\mathbf{I}^j$ of the same rank as the one holding $\mathbf{p}_i$ in $\mathbf{I}^i$.

In the multi-baseline context, because the optical centers are aligned, epipolar planes defined for a given pixel $\mathbf{p}_i$ in $\mathbf{I}^i$ and any other view $\mathbf{I}^j$ coincide. Successive pairwise binocular epipolar constraints thus ensure that corresponding pixels have the same y-value in every view $\mathbf{I}^i$. Hence, any 3D point $\mathbf{p} = (x_\mathbf{p}, y_\mathbf{p}, z_\mathbf{p})$ is projected into the [rectified] views $\mathbf{I}^i$, $\mathbf{I}^j$ onto corresponding pixels whose coordinates are respectively $\mathbf{p}_i$ and $\mathbf{p}_j = \mathbf{p}_i - (d^{i,j}_{\mathbf{p}_i}, 0)$, where $d^{i,j}_{\mathbf{p}_i} = u_i - u_j$ is called *horizontal disparity*.

A relationship between the horizontal disparity $d^{i,j}_{\mathbf{p}_i}$ and $\mathbf{p}$'s depth $z_\mathbf{p}$ can be established based on scale ratios between similar triangles. Let us consider the scale ratios in two pairs of such triangles, with apices on $\mathbf{c}$ and $\mathbf{p}$, respectively, and the camera centers $\mathbf{o}_i$ and $\mathbf{o}_j$, Fig. 8.6. Let $e_{i,j}$ be the distance between the center of the sensor ROI of camera i and j, defined by the triangle with apex on $\mathbf{c}$, and $e_{i,j} - d^{i,j}_{\mathbf{p}_i}$ be the distance between the corresponding pixels $\mathbf{p}_i$ and $\mathbf{p}_j$, in the sensors' plane, defined by the triangle with apex at $\mathbf{p}$. The relation between these two triangles yields the disparity-to-depth relation:

$$\left.\begin{array}{rl} e_{i,j} & = b_{i,j} \cdot (z_\mathbf{c} - f) \cdot z_\mathbf{c}^{-1} \\ e_{i,j} - d^{i,j}_{\mathbf{p}_i} & = b_{i,j} \cdot (z_\mathbf{p} - f) \cdot z_\mathbf{p}^{-1} \end{array}\right\} \Rightarrow d^{i,j}_{\mathbf{p}_i} = f \cdot b_{i,j} \cdot (z_\mathbf{p}^{-1} - z_\mathbf{c}^{-1}) \quad .$$

When the optical centers are evenly distributed (i.e., $\forall i, j \in \{0, \ldots, n-1\}, b_{i,j} = (j-i) \cdot b$), disparity values are scaled by $(j-i)$:

$$\forall i, j \in \{0, \ldots, n-1\} \qquad d^{i,j}_{\mathbf{p}_i} = (j-i) \cdot f \cdot b \cdot (z_\mathbf{p}^{-1} - z_\mathbf{c}^{-1}) = (j-i) \cdot d_\mathbf{p} \quad .$$

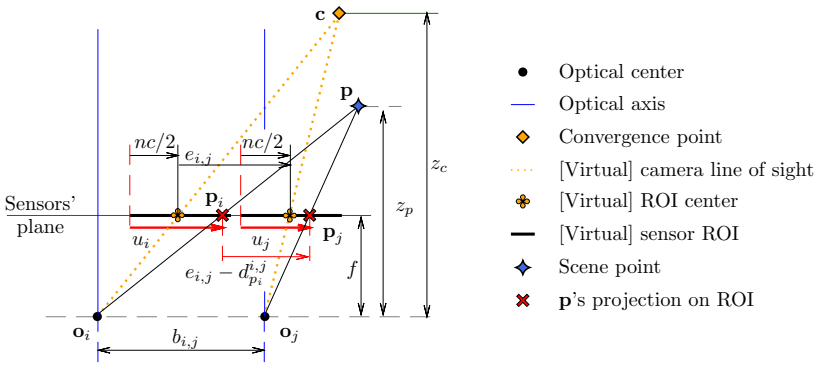


Figure 8.6: Projective geometry in off-axis simplified epipolar geometry (top view).

and disparity values for successive views are identical:

$$\forall i \in \{0, \ldots, n-2\} \quad d^{i,i+1}_{\mathbf{p}_i} = d_{\mathbf{p}}.$$

This common disparity $d_{\mathbf{p}}$ among each pair of successive views is conveniently used for each disparity assumption d for a pixel at $\mathbf{p}_i$ in $\mathbf{I}^i$. Instead of testing only two corresponding pixels for photo-consistency, one builds an associated geometrically consistent n-tuple in the multiscopic stereo matching process:

$$\forall j \in \{1, \ldots, n-1\} \quad \mathbf{p}_j = \mathbf{p}_i + (i-j) \cdot (d, 0) \quad . \tag{8.7}$$

These n-tuples contain one pixel per image, ordered according to their image number. Furthermore, thanks to Eq.(8.7), they all lie in the same epipolar plane and a common horizontal disparity assumption d is assigned to them. As such, pixels of a single n-tuple are corresponding projections on every view of a single 3D point.

To summarize, the presented multi-baseline stereovision paradigm reformulates the dense correspondence reconstruction problem as the answer to the question: "which of the geometrically consistent n-tuples correspond to actual 3D points in the scene according to their photo-consistency in the n views?".

Aside from the differences in the camera layout they are intended to handle, MVS methods may also be categorized according to what data or representation of the world they operate on [Seitz et al. 06]:

Scene-based methods employ a 3D scene model whose projections on views are checked for photo-consistency. As they are designed for a general freely arranged camera layout, they often use a 3D volume and rely on photometric similarity measures of the projections of the voxel cells, and remove others from the volume. Voxel coloring [Seitz and Dyer 99] preserves voxels whose cost is below a threshold. Space carving [Kutulakos and Seitz 00] progressively removes the photo-inconsistent voxels from an initial volume. More recently, a different category of methods has been proposed that use a scene model composed of a collection of planar patches or surfels whose depth and orientation are separately optimized to maximize their photo-consistency. For representing patches, such methods rely on planar polygons [Habbecke and Kobbelt 06], circular disks [Habbecke and Kobbelt 07] or pre-segmented superpixels [Micusik and Kosecka 10]. The seminal work of [Furukawa and Ponce 10] fits patches on pixels around detected sparse features, then expands them in order to fill gaps between their projections, and afterwards reconstructs and refines a mesh.

Some multi-baseline methods make use of the disparity space introduced by [Yang et al. 93] for reconstruction instead of working on the standard 3D scene. Making use of photo-consistency and visibility reasoning, [Ismael et al. 14] optimize a so-called *materiality map* in this space for improved multi-view reconstruction.

Image-based methods compute a set of depth or disparity maps which are merged later [Narayanan et al. 98, Goesele et al. 06] or to which they apply constraints [Gargallo and Sturm 05, Szeliski 99] to ensure a consistent 3D scene reconstruction. Some methods that expect a more restrictive camera layout, typically multi-baseline, directly match n-tuples as multiscopic pixel sets [Niquin et al. 10,Kang and Szeliski 04], as described above. Amongst methods intended for a free camera layout, some computationally more intensive techniques are dedicated to MVS from community photo collections (CPC) and have gained an increasing interest. They have to handle a large number of uncalibrated views of a scene [Goesele et al. 07]. New difficulties then arise as such views are typically shot at different times, with differing acquisition geometries (viewpoints, angles, focal lengths, resolutions), and usually differing environmental conditions (weather, exposure, occlusions). This makes it necessary to restrict the matching to subsets of views sharing similar exposure, and empower the methods to deal with significant baselines (distances) between the cameras.

Feature-based methods compute dense correspondences by first matching feature points which can be more robustly estimated than a complete disparity map. In a second step a surface model is fitted to

the reconstructed features [Taylor 03].

Image-based methods that rely on multiscopic matching of n-tuples share an important advantage with scene-based methods: implicit consistency of the reconstruction. Furthermore, both take full advantage of pixel redundancy to avoid as many false matches as possible while enabling smart occlusion handling schemes. The photo-consistency cost implied in those methods is often defined, for each 3D point of interest, as the aggregation of dissimilarity costs of its corresponding pixels over a set $\mathcal{R}$ of several pairs of views

$$c(\mathbf{p}) = \sum_{(i,j)\in\mathcal{R}} c(\mathbf{p}_i, \mathbf{p}_j) \quad . \tag{8.8}$$

Here $c(\mathbf{p}_i, \mathbf{p}_j)$ is the same cost function as used before in the binocular case and $\mathbf{p}_i$ is the pixel position of the backprojected 3D point $\mathbf{p}$ into the i-th view $\mathbf{I}^i$. Commonly employed pair sets $\mathcal{R}$ consist of:

- successive views $\mathcal{R} = \{\ (i, i+1) \mid \forall i \in \{0, \ldots, n-2\}\ \}$,
- every available pair $\mathcal{R} = \{\ (i, j) \mid \forall i, j \in \{0, \ldots, n-1\}, i < j\ \}$,
- pairs specifically selected according to geometrical considerations and/or similar recording conditions.

The first option is often preferred in a rectified layout as it makes the stereo method less sensitive to colorimetric shifts among the image set. Contrarily, the third is used when a very large number of views is available with widely spread viewpoints.

Using all views to compute the dissimilarity cost in Eq.(8.8) rarely leads to high-quality reconstructions, as a scene point $\mathbf{p}$ may be occluded in some of the cameras. However, as multiple views are available visibility may be reconstructed as well during the correspondence estimation [Kolmogorov and Zabih 02, Kutulakos and Seitz 00, Seitz and Dyer 99]. This visibility information can be used to improve the correspondence reconstruction by:

- restricting to a useful set of image pairs $\mathcal{R}_\mathbf{p} = \{(i, j) \in \mathcal{R} \mid$ with $\mathbf{p}$ visible in both i and $j\}$,
- weighting the dissimilarity costs in Eq.(8.8) according to $\mathbf{p}$'s visibility in the images, or
- replacing the dissimilarity cost by a predefined, heavy, penality cost for pairs for which $\mathbf{p}$ would occlude some already reconstructed 3D point.

Multi-view stereovision methods vary strongly with respect to methodology and tend to be computationally more complex than their binocular counterparts, as they have to deal with more data. Nevertheless, they tend to

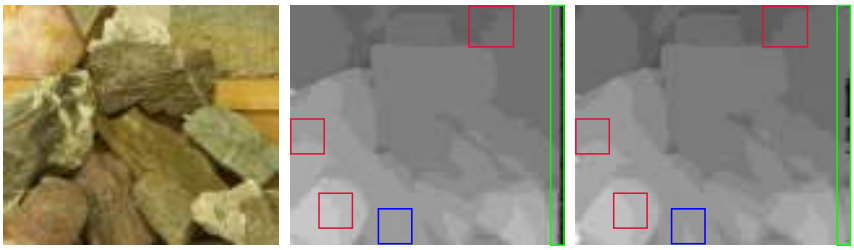


Figure 8.7: Results from a multi-baseline, scene-based method [Ismael et al. 14] on the Middleburry data set "Rocks2": left, one source view; center, disparity map computed from two views only; right, disparity map computed from a set of 4 views. Green rectangles highlight an area more completely reconstructed from 4 views; red rectangles focus on some areas more regularly reconstructed from four views; the blue rectangle points to a region with higher accuracy in the 4-view case.

exploit the increased redundancy to achieve more robust reconstructions: Fig. 8.7 shows that, with similar context, data, method and parameters, results computed from four views are more regular and complete and contain less outliers than those using two views.

8.5 Stereo on the GPU

Stereo estimation is typically a significant computational expense and improving the computation time of stereo has long been in the focus of research. One characteristic that has been leveraged is that the depth/disparity estimation in local stereo for pixel (x, y) has no dependencies to any other pixel, Section 8.2 and 8.3. Hence, parallel computation has long been explored for improving the computation time. Commodity graphics hardware (GPUs) nowadays provides a massively parallel processing platform with thousands of parallel compute cores and a significantly higher memory bandwidth than CPUs have at their disposal. Yang and Pollefeys [Yang and Pollefeys 03] proposed to leverage these highly parallel architectures to improve the computational performance utilizing the plane-sweeping stereo algorithm [Collins 96]. Besides leveraging the parallelism of GPUs, their method further leverages the high efficiency of texture mapping in GPUs.

Plane-sweeping stereo [Collins 96] is a multi-view stereo method with $n > 2$ views, $\mathbf{I}^0$ to $\mathbf{I}^{n-1}$ which does not rely on multiocular rectification, Section 8.4. It can use any set of multiple overlapping views to perform

stereo estimation. Plane-sweeping stereo estimation only requires the camera calibration of the views $\mathbf{I}^0$ to $\mathbf{I}^{n-1}$, as for example computed by structure from motion, Chapter 7. The core idea of plane-sweeping stereo is to perform the dense correspondence estimation by testing a series of plane hypotheses Π^i with $i = 1, \ldots, K$ for the scene, i.e. it assumes the scene is on a plane and then tests this hypothesis. Once all K plane hypotheses have been tested, the best plane is chosen for each pixel. This, however, does not mean that the scene has to be planar, as the plane only represents a local planar approximation of the scene for the function used to compute the matching cost. Please note that each plane is basically a slice in the cost-volume introduced in Section 8.3.

In plane-sweeping stereo the depth map relative to one of the images $\mathbf{I}^0, \ldots, \mathbf{I}^{n-1}$ is computed. This image is referred to as the reference view $\mathbf{I}^{ref}$ and its camera projection matrix is transformed to be $\mathbf{P}^{ref} = [\mathbf{U}_{3\times3} \quad 0_{3\times1}]$ where $\mathbf{U}$ is again the identity matrix. All other images' camera projection matrices are transformed as well to be in the same coordinate system as the reference image $\mathbf{I}^{ref}$. Given this unified coordinate system, the plane hypotheses Π^i are chosen with respect to $\mathbf{I}^{ref}$ to sample the depth interval $[d_{near}, d_{far}]$ with K steps [4] from a closest distance d_{near} to a farthest distance d_{far}. The plane hypotheses Π^i in Yang and Pollefeys [Yang and Pollefeys 03] were chosen to be fronto-parallel to the reference image $\mathbf{I}^{ref}$, i.e. their normals n_i are equal to $[0\ 0\ 1]^T$ and their distance d_i corresponds to the depth (distance from the reference camera to the plane). Conceptually, to test any specific plane hypothesis Π^i all views can be warped onto the plane and their photoconsistency [5] can be evaluated using any of the cost functions from Section 8.3. This would require the definition of a raster on the plane hypothesis Π^i, which is a challenge. Equivalently, a hypothesis Π^i can be tested with respect to the reference view $\mathbf{I}^{ref}$ by warping all other images $\{\mathbf{I}^0, \ldots, \mathbf{I}^{n-1}\} \backslash \mathbf{I}^{ref}$ to the reference view $\mathbf{I}^{ref}$. Photoconsistency at each pixel (x, y) in the reference image $\mathbf{I}^{ref}$ can then be tested using the warped images. The warp of pixel (x, y) from the reference view $\mathbf{I}^{ref}$ to image $\mathbf{I}^j$ over plane Π^i is a planar mapping and can be described by a planar homography $\mathbf{H}_{\Pi^i,\mathbf{P}^j}$. Here, $\mathbf{P}^j = \mathbf{K}_j \begin{bmatrix} \mathbf{R}_j^T & -\mathbf{R}_j^T \mathbf{C}_j \end{bmatrix}$ is the camera projection matrix of the camera corresponding to image $\mathbf{I}^j$ with $\mathbf{K}_j$ being the camera calibration matrix and $\mathbf{R}_j$, $\mathbf{C}_j$ representing the rotation of the camera and the camera center, respectively [Hartley and Zisserman 03]. The homography $\mathbf{H}_{\Pi^i,\mathbf{P}^j}$ is given

[4] Please note that the steps are typically not equidistant steps. They are chosen to have equal disparity sampling in the reference view. For more details on the hypotheses generation see Gallup *et al.* [Gallup et al. 07].

[5] Photoconsistency is the color similarity, i.e. the highest photo consistency is achieved when all views have the same color.

Figure 8.8: Left: Plane-sweeping stereo's reference image from an outdoor video sequence. Right: Depth map computed by plane-sweeping stereo using a total of eleven views.

by:

$$\mathbf{H}_{\Pi^i,\mathbf{P}^j} = \mathbf{K}_j \left(\mathbf{R}_j^T + \frac{\mathbf{R}_j^T \mathbf{C}_j n_i^T}{d_i} \right) \mathbf{K}_r^{-1} \quad . \tag{8.9}$$

Then, the location (x_j, y_j) in image $\mathbf{I}^j$ of the warped pixel (x, y) in the reference image $\mathbf{I}^{ref}$ can be computed by

$$(x'\ y'\ w')^T = \mathbf{H}_{\Pi^i,\mathbf{P}^j} (x\ y\ 1)^T \text{ and } x_j = \frac{x'}{w'}, \quad y_j = \frac{y'}{w'} \quad . \tag{8.10}$$

If the scene point that projects into pixel (x, y) is on the plane Π^i then the colors of pixel (x, y) in the reference image $\mathbf{I}^{ref}$ and the color of pixel (x_j, y_j) in image $\mathbf{I}^j$ should be very similar. Similar to the binocular case, their similarity can be measured by a variety of measures, as explained in Section 8.3.

Using the GPU, the warping between the reference image $\mathbf{I}^{ref}$ and image $\mathbf{I}^j$ with the homography $\mathbf{H}_{\Pi^i,\mathbf{P}^j}$ can be performed using projective texture mapping [Segal et al. 92], i.e., a projection of an input image onto a geometric primitive. This projection is highly efficient on GPUs. The similarity evaluation and the selection of the best plane hypothesis for each pixel in the reference view are independent for each pixel and hence can be performed in parallel as well. This high degree of parallelism provides speedup factors of one hundred and more for stereo estimation on GPUs [Gallup et al. 07]. In [Gallup et al. 07] it is further proposed to improve the local approximation of the scene geometry with the plane hypotheses Π^i by using multiple plane orientations. This indeed improves the accuracy of the stereo estimation by better modeling planes that are seen under oblique viewing directions, for example the ground plane. Figure 8.8 shows an image and its example depth map computed by multi-way plane-sweeping [Gallup et al. 07].

8.6 Summary

This chapter only touched the tip of the iceberg that represents the field of dense correspondence estimation. Nevertheless, the knowledge provided here poses a useful basis for understanding any of the other current state-of-the-art correspondence techniques and provides a flexible basic framework from which to build upon. The simplicity of local methods makes them attractive from a beginner's perspective as well as a computational view. Choosing the right dissimilarity function proves crucial for the quality of the algorithm, but with well-chosen adaptive weights, state-of-the-art results are achievable. Posing the aggregation step as a filtering process can dramatically improve the speed of the correspondence algorithm. To handle occlusions, a symmetry check between the input images can be used and several extensions including higher dimensional solution spaces and slanted surfaces can be easily incorporated at the cost of higher computation times.

The finding that current state-of-the-art dense correspondence algorithms achieve energies in the cost function that are below that of ground truth scenes [Szeliski et al. 08] may appear dissatisfying for researchers starting in this area. It should be mentioned, though, that this opens up the door for more creative approaches that do not follow the standard paths but try to come up with novel ideas and complete new algorithms that differ in more than the choice of a new data or regularization term. Maybe the way to go are also specialized algorithms specifically targeting certain scenes or applications. While from a vision perspective the goal is to find the best automatic algorithm for dense correspondence matching, it may make sense to have an algorithm that one can improve through additional user input [Klose et al. 11, Ruhl et al. 13], e.g., for multimedia applications like image interpolation. In such cases, these interactive correction tools are of utmost importance. Further on, if massive amounts of images are to be matched, speed may be of highest interest [Frahm et al. 10].