Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Computer Science and Media

Cross-Scale Cost Aggregation and PatchMatch Filter: An Improved Dense Stereo Matching Method

Master's Thesis

Felicitas Höbelt
b. 05.06.1987 in Magdeburg

Matriculation Number 60167

First Referee: Prof. Volker Rodehorst
Second Referee: Prof. Charles Wüthrich

Submission date: November 28th 2016

Erklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe. Die Arbeit wurde in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Weimar, den 28. November 2016

Felicitas Höbelt

Acknowledgment

I would first like to thank Prof. Volker Rodehorst. From start to finish he allowed this thesis to be my own work, but offered valuable insights into the topic whenever I ran into a trouble spot.

I would like to thank Prof. Charles Wüthrich for being the second referee.

Dr. Jens Kersten receives special thanks for his constructive comments that helped rebuilding my self-confidence repeatedly throughout the process.

I would also like to acknowledge Jiangbo Lu and Yu Li, who were so kind as to answer my detailed questions about their work on the PatchMatch Filter framework and even provided me with some of their unpublished results.

Finally, I must express my profound gratitude to my family and friends for their unfailing support, encouragement and emotional care packages that kept me sane. Thank you.

Contents

Chapter 1

Introduction

The essential problem that concerns dense stereo matching is estimating the depth of a scene given two images. The human visual system is built to solve this problem successfully at very high rates (albeit not with the precision that would be needed for exact depth measurements). We are equipped with a highly optimized system, that can extract depth information even for random dot patterns. If you manage to relax your eyes and stare into infinity, the two images in figure 1.1 will merge to a 3D impression.

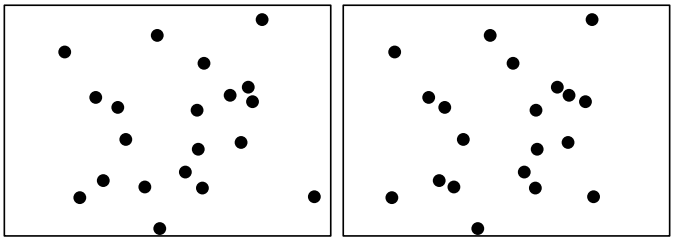


Figure 1.1: *Dot patterns like these are popular in stereo vision studies with humans, because no other depth cues are present such as converging lines or relative sizes of everyday objects.*

Automatic, high-precision depth sensing is an essential task for many applications such as surface reconstruction, virtual view rendering, and autonomous navigation of cars, robots, and UAVs.

Of course there exist methods to measure depth directly, for example with laser scanning, but the needed devices are often costly, heavy and power-greedy. Passive measurement beats active measurement in these categories, because photosensors make use of the natural light and offer high image resolutions at low cost.

However, automatic 3D-scene reconstruction from two images is not a trivial problem: A scene point, that is projected to two images, can be recovered by triangulation only, if the positions of the corresponding two projections in the two images are known. For a dense reconstruction, correspondences for each pixel need to be known.

Finding the correct pixel correspondences is the main goal of dense stereo matching algorithms. They have to deal with occlusions (not all parts of a scene are visible in both images), radiometric distortions and illumination changes, reflections, repeating patterns, and other sources of ambiguity (see fig. 1.2).

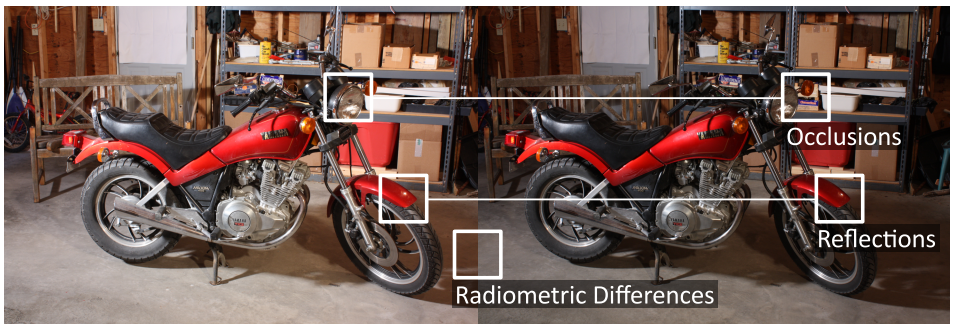


Figure 1.2: *Only some of the difficulties when searching for corresponding points are shown in the 'challenging' version of the Motorcycle image pair from the Middlebury Stereo Evaluation benchmark [7].*

These (for the best part unsolved) issues make dense stereo correspondence one of the most active research fields in computer vision. The development of stereo algorithms has also been driven by the need to understand and model the human visual system.

There have been huge improvements in the quality of stereo algorithms over time, yet the challenge is increased regularly by established benchmarks. Two fundamentally different classes of solutions are local stereo algorithms and global stereo algorithms. The first uses a (locally limited) correlation window to search for point correspondences in regions of maximum correlation between the two images. The second formulates an energy function, encapsulating more complex assumptions about the scene, and tries to minimize it with global optimization methods, resulting in high-quality depth images. Both classes have their pros and cons. Local methods are generally more flexible in their structure and manage to keep their complexity low, which is often a huge drawback in global methods.

After giving a short introduction into the topic and an overview over existing work in this field, a new local, correlation-based method for dense stereo correspondence will be proposed. It will make use of various components of established stereo matching methods, that have not been combined before. Some components remain in an experimental state, but promising future directions of development will be given. The conducted experiments show that even without finetuning the parameters, the proposed method achieves very good results compared to the method on which it is built (PatchMatch Filter) and also good results compared to the top-performing methods of the Middlebury stereo evaluation website.

Chapter 2

Basics of Stereo Correspondence Search

This chapter will introduce the most important concepts of stereo correspondence search. Also, an overview will be given of some of the recurring terms and general assumptions made throughout this work.

2.1 Geometric Foundation

Most of the following terms and equations are adopted from [28], a great resource for more detailed and generalized information concerning (multiple) view geometry.

When a non-planar 3D-scene is projected onto a 2D-image plane, the scene's depth information is lost in the process. The idea behind stereo matching is recovering that lost information given two 2D-images of the scene.

The sets of image points x_i and x'_i are produced by transforming the set of scene points X_i using the cameras' projection matrices P and P':

$$PX_i = x_i$$
$$P'X_i = x'_i$$

If the camera calibration is known, the 3D-points X_i can be recovered by triangulation to retain points in Euclidean space. To achieve this, corresponding point pairs have to be found, i.e. pairs of image points (x, x') that are projections of the same scene point.

Corresponding point pairs lie on the epipolar plane defined by the two centers of projection (the cameras) and the scene point they show. The infinite line that

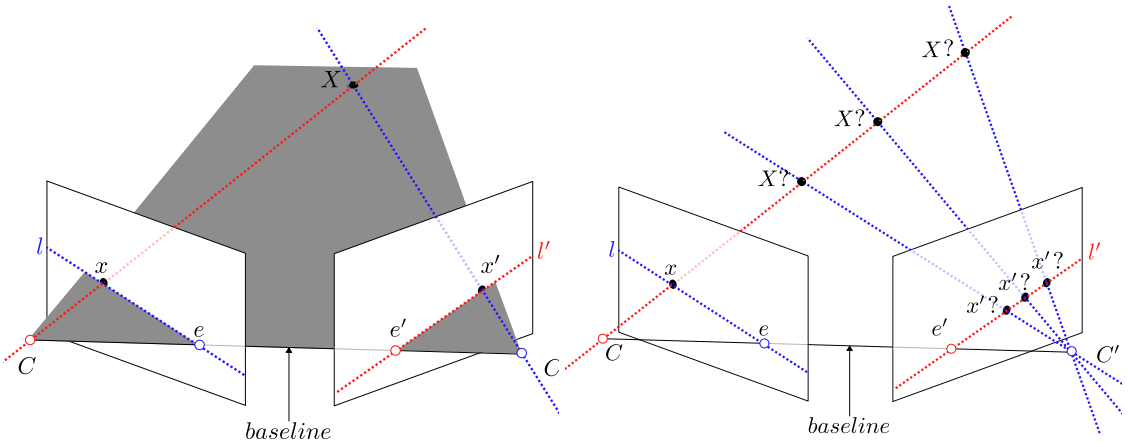


Figure 2.1: *With the scene point X the two projection centers C and C' form an epipolar plane. The line connecting C, x, and X is imaged in the other view as l' (figure 2.1a). If the scene point X is to be reconstructed from the images, the correct corresponding point x' for x has to be determined (figure 2.1b). The position of x' is restricted to the epipolar line l'. (The baseline is the line that connects C and C'. The epipoles are denoted by e and e'. Adapted from [28])*

connects the scene point X and the projected image point x in the first view is imaged in the second view also as an (epipolar) line, on which the corresponding image point x' (the second projection) must lie (fig. 2.1). Stereo correspondence search deals with automatically finding these corresponding points. The objective of dense (as opposed to sparse) stereo correspondence methods is to determine the corresponding point for each pixel of each input image in order to be able to reconstruct a dense 3D-scene.

The input views can be transformed so that they lie on one common image plane and their epipolar lines are parallel. This process is called rectification and the resulting images are called rectified. Rectification reduces the search scope for the correspondence search, as now corresponding points lie on the same horizontal line.

A corresponding pair is described in terms of relative pixel positions. The reference image contains the current reference pixel $p_R = (x_R, y_R)$, that is the target for which a match should be found. The other view is called search image, consequently named $p_S = (x_S, y_S)$. In stereo normal case (rectified images and search along horizontal lines) the difference in the x-coordinate of the two points is called disparity d as illustrated by figure 2.2:

$$d = x_R - x_S$$

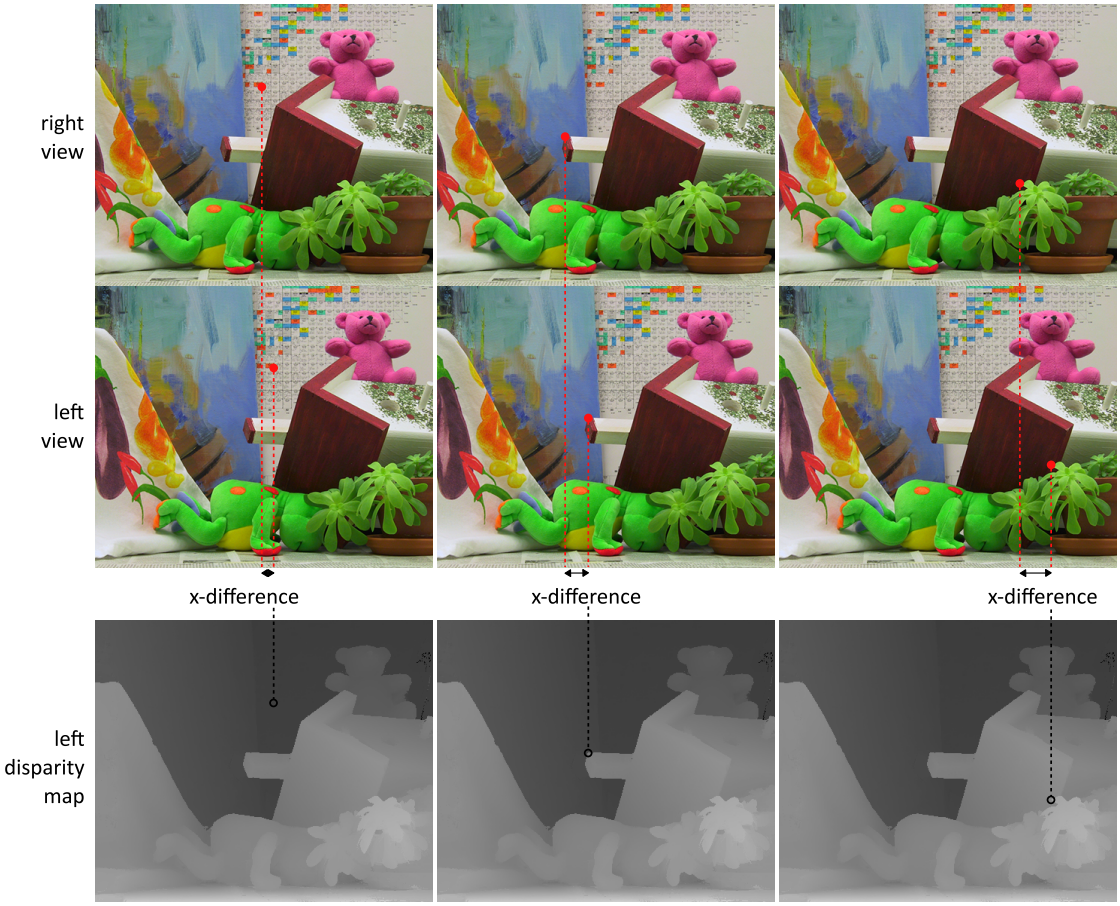
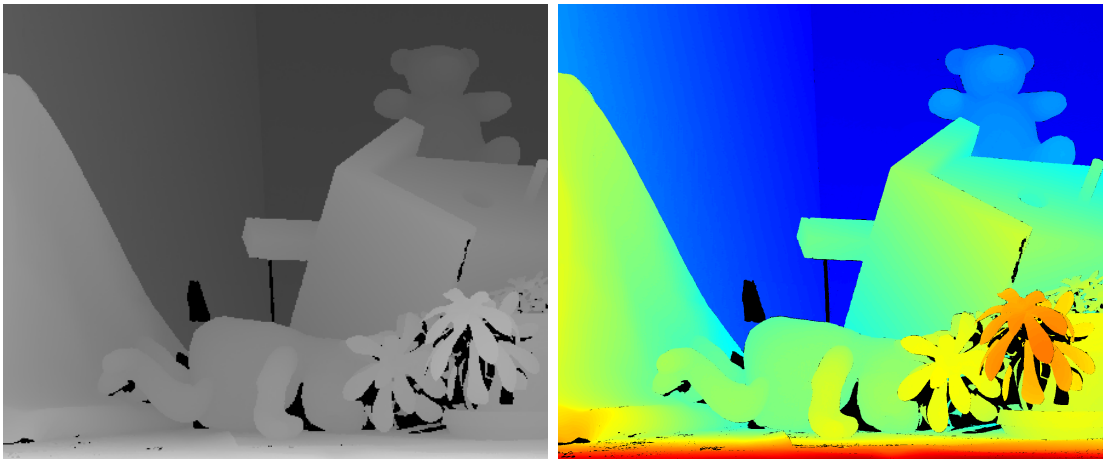


Figure 2.2: *The difference in the x-coordinate of corresponding points is the disparity. Objects in the background have a small absolute disparity (left image column), foreground objects have a large absolute disparity (right image column) The disparity map shown is the result of [45] for the quarter-resolution Teddy image pair from [7].*

The disparity will change its sign depending on which of the two views is considered the reference image. The found matches per pixel are stored as disparity value (at the respective pixel position), also called a disparity map. The disparity can be interpreted as inverse depth, hence, the disparity map can be interpreted as a depth image (see figure 2.3).

More general, the correspondence search can be seen as a labelling problem: assign each pixel a 1D-label (disparity) so a certain condition is fulfilled. It will be shown later how the assigned label can also be a 3D-label.

Rectified image pairs will be assumed throughout the thesis.



(a) (b)

Figure 2.3: *The ground truth depth values in figure 2.3a are of the left image of the Teddy image pair at quarter-resolution. The darker the pixel, the smaller the disparity. For displaying disparity ranges exceeding 255, the disparity values are color-coded. The ground truth depth values in figure 2.3b are from the full-resolution Teddy image pair. Black encodes unknown values in both.*

2.2 Classification Systems

The growing number of stereo correspondence or stereo matching methods cannot be compared and studied without classification systems. The taxonomy by Scharstein and Szeliski [60] is probably the most-applied. It identifies the core components of stereo correspondence algorithms. Other perspectives for categorizing methods are the analysis of the assumptions that are modelled by each method, or the context in which disparity decisions are made (also included in [60]). More properties for classification exist, such as complexity or density of the disparity output, but they will not be detailed here.

2.2.1 Core Components

By analyzing several stereo correspondence algorithms Scharstein and Szeliski [60] identified the essential tasks a stereo algorithm has to solve in order to compute a disparity map. Their paper lists

1. matching cost computation,
2. cost aggregation,
3. disparity computation/ optimization, and

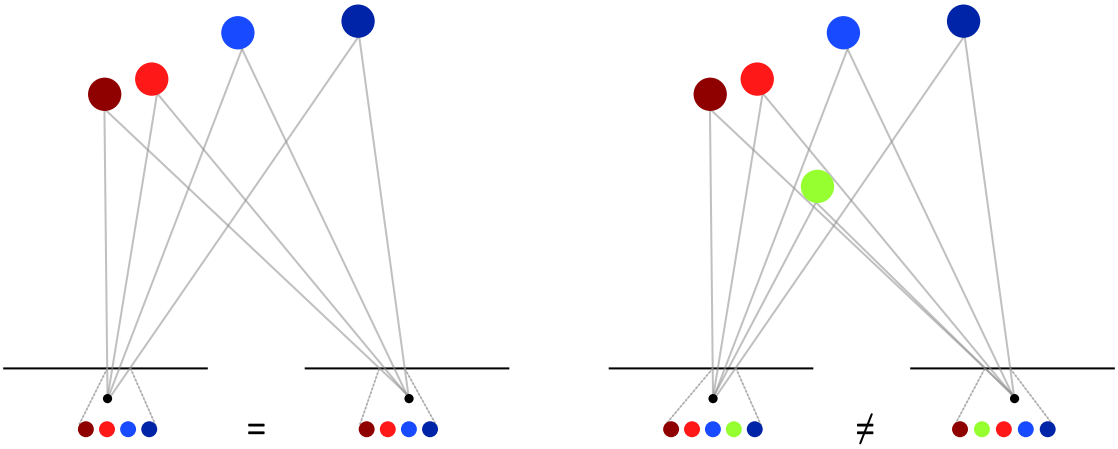


Figure 2.4: *The ordering of object projections only is preserved across images if the objects lie on a common depth plane (left). Objects not fitting into this plane will violate the ordering constraint (right).*

4. disparity refinement.

While these components are similar for almost all stereo algorithms, the concrete implementation of each component varies from method to method. It makes sense to examine the differences between methods to isolate meaningful classes of strategies. The strategies are derived from the respective context of use and the underlying assumptions that model the captured world.

2.2.2 Assumptions

The assumptions about the imaged world determine the limits of the respective method. The following list contains some conventional, reliable assumptions as well as unusual assumptions, which are employed for highly specialized tasks.

Photoconsistency is the assumption that the scene is made up of Lambertian surfaces, i.e. the color of two corresponding pixels is similar in both images. The assumption is weak in cases of lighting and exposure differences between images, and sensitive to noise. It does not hold for Non-Lambertian surfaces (e.g. reflecting surfaces). Related to photoconsistency is gradientconsistency, only here we expect the gradients to be consistens across views. Gradients are less prone to camera noise and varying exposure than intensities alone.

Smoothness assumes the scene consists of piecewise smooth surfaces, meaning the depth does not change apruptly. This constraint is regularly violated at object borders. Very thin objects (e.g. twigs, threads, cables) also pose a challenge.

The ordering assumption expects the relative position of scene points to not change between views. This constraint only works for scene points that lie approximately on the same depth plane (see fig. 2.4), therefore the ordering assumption is of little importance as of today.

Line constraints rely on edge detection and impose similar angles and segment lengths for corresponding lines. Because line segment lengths are unreliable and this constraint concerns only sparse features, it is seldomly used in dense stereo correspondence methods.

The uniqueness assumption requires a corresponding pixel pair to always be a one-to-one relationship. This constraint is violated in case of occlusion by another object closer to the camera, or self-occlusion, i.e. points representing the same (slanted) surface are projected to fewer pixels in one image. In these cases several pixels of one image can correspond to one pixel in the second image, as illustrated by figure 2.5.

Simple over complex is a niche constraint employed by [15], who aim to keep an overall low count of detected 3D surfaces in the scene. The approach was later extended to objects [16], where the constraint is used to favor compact 3D objects. This assumption allows for example the correct detection of large surfaces that appear segmented in both images.

Physical constraints are in summation all constraints that assume underlying concepts like weight, inertia and volume. In [27] a block model of the visible buildings is built by choosing a configuration so no building would 'tip over'.

Knowledge-based constraints encapsulate all constraints that are based on higher-level knowledge of the world, for example typical 3D-shapes for classes of objects (buildings, humans, cars).

An algorithm does not need to incorporate all of the listed assumptions. Some are more general (e.g. photoconsistency) and some may be of use only for highly specialized stereo correspondence (e.g. physical constraints). The implementation of the assumptions that make up the world model is in most cases distributed over the core components, for example the photoconsistency assumption is widely used in the matching cost computation while the uniqueness constraint is imposed as a disparity refinement or as confidence check in the post-processing step.

2.2.3 Decision Context

For each reference point we have a large number of potential corresponding points in the search image. We need to assess the quality of each potential corresponding

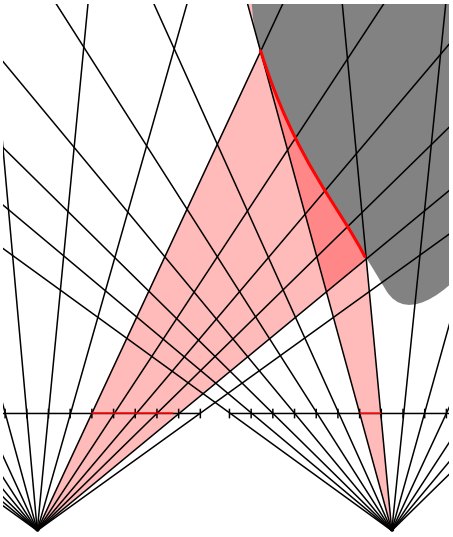


Figure 2.5: *Highly slanted surfaces violate the uniqueness constraint, because a pixel may be part of several corresponding pixel pairs.*

pair, that incorporates some of the assumptions we have about the world, typically photo- and gradientconsistency as well as smoothness. This quality measure is called matching cost and encodes the deviation from these assumptions: the smaller the cost for the pixel pair the more likely the respective disparity hypothesis. The goal of stereo algorithms is to find a disparity configuration that minimizes the overall costs.

The two most often applied assumptions, the photoconsistency and the smoothness assumption, have a certain scope of consequence for the disparities.

The photoconsistency matching cost only depends on the input images. Given intensities of two pixels, a disparity decision can be made locally, it does not depend on other disparities. Thus, the overall optimization of minimizing the photoconsistency deviations is done by minimizing the photoconsistency deviation per pixel.

Minimizing smoothness deviations is not a local, but a global optimization problem: the disparity decision for each pixel depends on the disparities of the pixels around it. Global methods employ this kind of explicit smoothness assumption in combination with the photoconsistency constraint (because only optimizing for smoothness would give a meaningless disparity map with the same value for each pixel). Because finding an exact solution of these optimization problems is often not possible in practice, optimization techniques giving a good approximate solution are employed.

Local methods still use smoothness in an implicit way: To stabilize the noisy photoconsistency results, the photoconsistency of the neighboring pixels are taken into account. The matching costs for a pixel are smoothed by averaging the matching costs within a local window (correlation window). This assumes close pixels belong to the same (smooth) surface, so the pixel under consideration and its immediate neighbors should all have similarly low matching costs (a high correlation) if the disparity hypothesis is correct. Naturally, the matching costs should be high (low correlation) for all pixels of a window for wrong disparity hypotheses. Because this disparity decision is still independent from other disparities, it can also be optimized locally.

It is important to note that the implicit smoothness assumption used by local methods does not necessarily generate a smooth disparity map (as with explicit smoothness used by global methods).

The distinction into local and global methods can be done in the same manner for other constraints, but conventionally local methods use photoconsistency and implicit smoothness, while global methods employ photoconsistency combined with explicit smoothness.

This thesis will mainly focus on local methods, although occasionally concepts used by global methods are mentioned to give a wider context.

Chapter 3

Related Work

This chapter will try to give an overview for the publications of methods that are relevant for this work.

Although the quality of the cost function results itself is of great importance for stereo algorithms, the corresponding section will be kept at a minimum. The second section will bring together different solutions for the large problem area of how to resolve matching ambiguities: from the diverse cost space smoothing approaches to hierarchical approaches to solutions utilizing distinctiveness and confidence. The final section covers work regarding the two tightly coupled concepts of continuous disparities and 3D-support windows.

3.1 Cost Functions

Cost functions determine the dissimilarity of two pixels p and q with respect to a certain world model assumption. High costs correspond to 'not similar', low costs correspond to 'similar'. The terms 'similarity' and 'dissimilarity (cost)' function are often used interchangeably, since they can easily be converted to one another. Most algorithms only keep track of the best (i.e. smallest) cost for each pixel ('winner takes all'). In the opposite approach ('loser gets nothing'), the set of solutions is narrowed down by excluding the disparities corresponding to high costs.

Although deviations from other assumptions, e.g. the smoothness assumption, can also be viewed as 'costs', the term 'cost function' usually indicates measuring the deviations from photo- or gradientconsistency.

An excellent review and thorough evaluation of different cost functions is done by Hirschmüller and Scharstein in [34]. Here, I will only focus on the cost functions relevant for the proposed method introduced in chapter 4 with comments on their

strengths and weaknesses.

The following cost functions are denoted C_x with x being the name of the respective method. The arguments are the pixel p and a disparity d from which a second pixel p' can be derived by substracting d from p's x-coordinate, assuming p is in the left image. I_L and I_R are the intensities of left respective right image.

Absolute/squared differences (AD/SD)

$$C_{AD}(p,d) = |I_L(p) - I_R(p-d)|$$
$$C_{SD}(p,d) = (I_L(p) - I_R(p-d))^2$$

If AD and SD summarized over all pixels of a neighborhood N of p, they become SAD (sum of absolute differences) and SSD (sum of squared differences)

$$C_{SAD}(p,d) = \sum_{q \in N_p} |I_L(p) - I_R(p-d)|$$
$$C_{SSD}(p,d) = \sum_{q \in N_p} (I_L(p) - I_R(p-d))^2$$

For color images (e.g. RGB), the results for each channel are either averaged [48] or summed up [42]. These color differences are sensitive to radiometric differences between images, so the color intensities are often replaced or complemented by the slightly more robust gradient.

Census transform

The Census transform generates a bit string for p, encoding the relative intensities of p's neighborhood N, i.e. if a pixel q in N has a lower (or as alternative: higher) intensity than p, the comparison result is stored in one bit. The position mapping between bit string and neighborhood pixels is arbitrary but fixed throughout the algorithm using it. The same is true for the condition when a bit is set to 1. An example Census transorm is illustrated in figure 3.1. The Census cost function $C_{Census}(p,d)$ is computed as the Hamming distance between the Census transforms of the two pixels p and $p-d$. The neighborhood is typically 3×3 (stored as 8 bits) or 5×5 (stored as 24 bits), or any size resulting in 32 or 64 bits, so it can be stored as built-in number types. This measure is very stable in presence of radiometric distortions but fails in regions without texture or considerable gradient variation.

Because for most applications and image data sets there exist no one-fits-all cost function, many methods resort to combining several complementary cost functions.

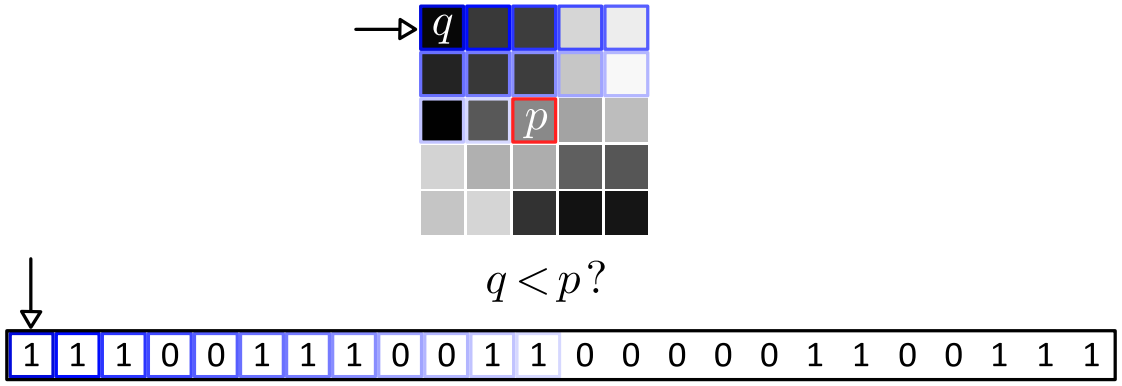


Figure 3.1: *The Census transform of a* 5×5 *window is stored in 24 bits. The comparison is done row-wise here as indicated by the color.*

One established combination (e.g. used in [14, 55], similarly found in [40, 42]) is the sum of truncated color and truncated gradient differences

$$C_{AD+Grad}(p,d) = (1-\alpha)\cdot \min(|I_L(p) - I_R(p-d)|, \tau_{col}) + \\ \alpha \cdot \min(|\nabla I_L(p) - \nabla I_R(p-d)|, \tau_{grad})$$

The ∇ operator denotes the gradient, τ_{col} and τ_{grad} are empirically determined truncation thresholds. The user-defined parameter $\alpha \in [0,1]$ controls the weighting of color and gradient components and is usually set to favor the latter.
A second robust combination is the sum of AD and census cost. In [48] a function ρ is introduced to handle outliers and weighting, parameters λ_{AD} and λ_{Census} are user-defined.

$$C_{AD+Census}(p,d) = \rho(C_{AD}(p,d), \lambda_{AD}) + \\ \rho(C_{Census}(p,d), \lambda_{Census})$$
$$\rho(c,\lambda) = 1 - exp(-\frac{c}{\lambda})$$

Besides a pre-determined weighting of the individual cost function components, cost functions may also be combined adaptively. Saygili et al. [56] examine the confidence af several cost function results, which varies according to local properties of the image regions, and in an experimental setup fuse up to 11 different cost measures adaptively. Klaus et al. [42] fuse two measures adaptively.

3.2 Resolving Ambiguities

The ideal cost profile for determining a correspondence has a distinct and unambiguous minimum for the correct match.

Working with the raw costs in realistic scenarios easily introduces matching er-

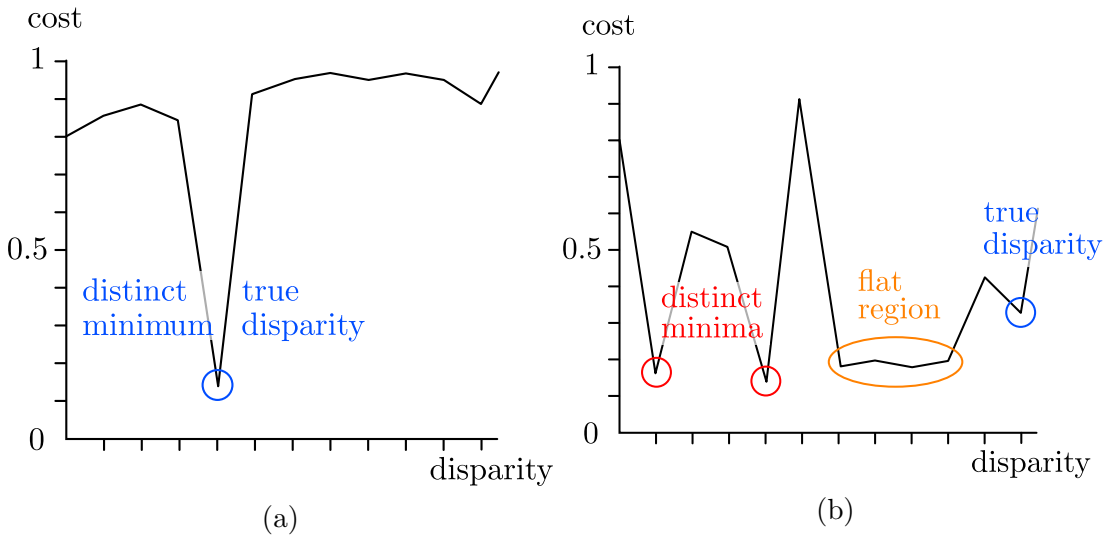


Figure 3.2: *In the ideal cost profile 3.2a the true disparity is a distinct minimum, but a realistic cost profile 3.2b includes ambiguities that obscure the true solution. (Adapted from [51])*

rors: cost curves can have several minima, flat minima or flat regions without a distinct minimum, or a correct match (according to groundtruth) does not lie in regions of low costs at all (fig. 3.2).

The ambiguities in the raw costs emerge from several systematic and structural weaknesses. Apart from the random noise component that is for the most part unavoidable in any real-world imaging, there are particular issues for stereo correspondence algorithms.

'Material' noise (systematic+structural) I would call the case where cost measurements fail because realistic non-lambertian material properties are not accounted for in the (photoconsistency) assumptions - making it a systematic problem. In part they are also structural, because they depend on the objects used in the scene.

Occlusions (systematic) , or to be exact half-occlusions, are regions that are visible in one image but not in the other. Although these parts of the scene do have a depth, their respective pixels cannot have a direct match. This is due to the violation of the expected 1-to-1 relationship of matching pixels: in case of an occlusion there are two scene points mapped to the same pixel (see figure 3.3). While most of the time two different objects contribute to this problem, a similar situation (self-occlusion) may arise within one strongly curved or slanted object.

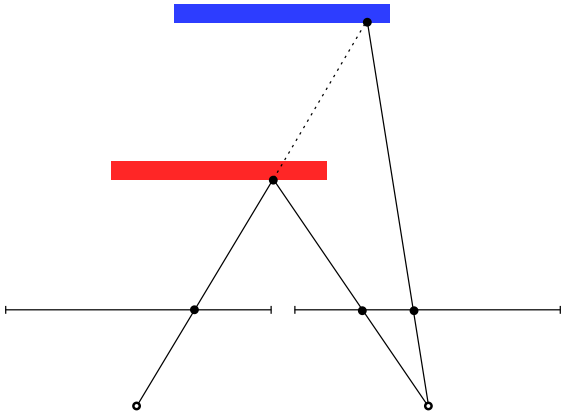


Figure 3.3: *The blue object in the background is occluded in the left view.*

Low depth precision (systematic) is present in systems set up to only compute the costs for discrete disparities. If the true depth does not fall exactly into one of the integer disparities, the respective cost will not be computed and the true minimum will be missed. This can be seen as 'staircase effect' in the depth image, as sketched in figure 3.4.

Repeating patterns and large textureless regions (structural) are visible in the cost profile of a pixel as several minima close in cost value or as flat low cost regions without a distinct minimum. A pixel might be very similar to more than one target pixel, because the target scanline contains a lot self-similar pixels. Random and 'material' noise amplify this effect.

While these ambiguities arise in all stereo correspondence algorithms working with passive sensor images, local and global approaches deal with them differently. There has been ongoing research to counter the effects of these problematic factors and to resolve ambiguities beyond 'just' choosing a stable dissimilarity measure.

The most elementary strategy for cost space refinement is 'cost aggregation' or smoothing the cost space. Although there exist sophisticated aggregation methods, cost smoothing is only able to solve a subset of ambiguities. The following subsections will cover different cost aggregation methods, as well as a summary of research dealing with complementary techniques, which include hierarchical constraints as well as exploiting distinctiveness and confidence.

3.2.1 Cost Space Smoothing

The obvious course of action to remove noise from a function is smoothing it e.g. by averaging or applying a Gaussian blur filter. Averaging values of a region around the respective value is justified by the smoothness assumption: surrounding

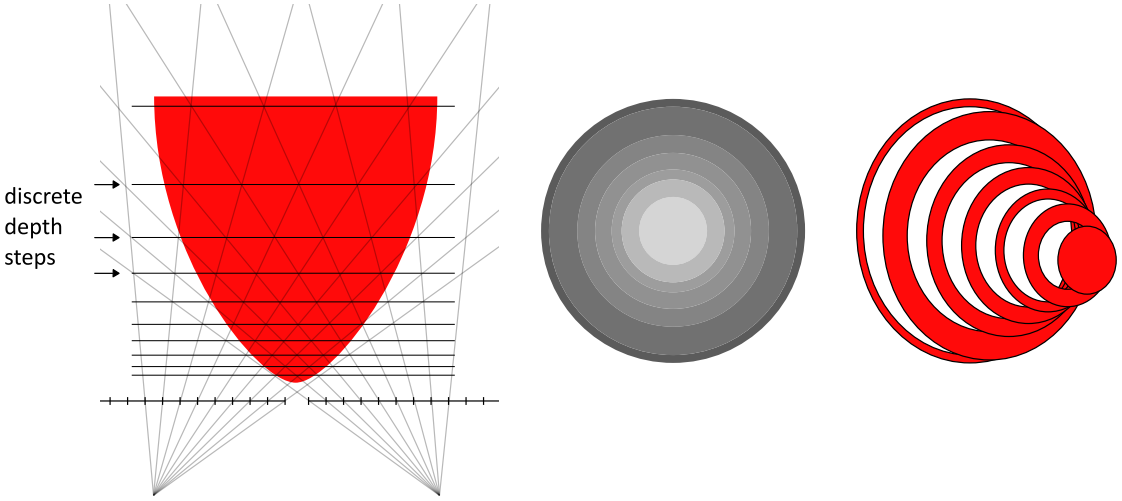


Figure 3.4: *If only discrete disparities are produced by the algorithm (matching pixels with pixels), slanted or rounded objects will display the 'staircase' artefact (center: depth mapped to gray-values, right: schematic 3D-view).*

pixels are likely to have similar costs for similar depths. The smoothing region used is commonly called support area, support region, or support window. Also 'aggregation' and 'support' are sometimes used interchangeably.

The efficiency and accuracy of support aggregation depends on the shape of the support area. The most simple and computationally cheap area is a square window of fixed size. The window is centered on the pixel of interest and the individual cost is updated with the average of all individual pixel costs in the support area. Determining the optimal window size is a delicate task: small windows are more prone to noise, whereas large windows are more likely to lose detail when the smoothness assumption is violated. This happens at depth discontinuities, i.e. the support region contains pixels from foreground and background and leads to an error known as 'foreground fattening' effect, where the boundaries of objects in the foreground become enlarged.

There exist several strategies on how to only select the pixels that should lie in the support area to avoid smoothing over object edges.

Binary Weights (Adaptive Window Shape)

Since a square or rectangular window is the most computationally efficient shape for a support region, some approaches try to improve accuracy by combining multiple such regions. In [17] and [25] aggregation is done for several shifted windows and the lowest aggregated cost is kept. Hirschmüller et al. [32] test different configurations to sum up only the best parts of a subdivided window. Arbitrary

shaped windows (though in practice still rectangular) based on region statistics are proposed by [20] and [41], Veksler [67] makes use of integral images to increase the number of tested candidate windows without losing performance. Inspired by the human visual system [53] use circular windows.

These mentioned shapes or combinations of shapes are mostly steered by the aggregated matching cost results, which is (inversely) used as a confidence measure: a candidate window with low costs is likely to be a good support window. As described towards the end of this section, confidence measures can be used to modulate the initial costs, sometimes even circumventing the aggregation step [57]. They are also often applied in any refinement steps taking place after the disparity computation.

Another more costly approach are irregular support regions that adapt to pixel properties such as intensities and position, essentially performing a local image segmentation. While (over)segmentation is used in global methods to decrease complexity as mentioned in section 3.3, it detracts from the superiority in speed of local methods. Additionally, segmentation algorithms tend to be very sensitive to their parameters, any errors will be propagated except the stereo algorithm has built-in mechanisms to recover from those.

Nevertheless, segmentation significantly improves the disparity map quality. Besides the straighforward use of an explicit image segmentation algorithm such as Mean Shift in a pre-processing step as done in [26, 63, 64], there exist a wide spectrum of stereo algorithms that define the support region ad hoc, resembling or borrowing from color segmentation algorithms.

Based on color similarity and connectivity Zhang et al. [76] construct cross-based irregular support regions by combining cross-shaped segments computed per pixel. The method by Unger and Ilic [65] relies on random walks for aggregating support. The walks do not cross color boundaries and thus are related to color segmentation, with the advantage of additionally gathering stochastic information about a region.

Continuous Weights

A powerful tool to combine the cost smoothing effect of large aggregation windows with discontinuity preservation are edge-aware image filters such as the Bilateral Filter [62]. These filters are applied to de-noise the cost volume (or cost slices) without blurring over intensity edges of the input image, which are presumably present at depth discontinuities. This means the filter ensures that a pixel's matching cost is stabilized only using pixels on the same side of the edge, i.e. of a similar depth. This alleviates the need for custom-built window shapes and explicit segmentation in pre-processing. Furthermore, implementations of edge-aware filters are included in most of the commonly used image processing libraries (e.g.

OpenCV [8], or Matlab's image processing toolbox [6]).

One of the most influential publications in the context of stereo matching is by Yoon and Kweon [73]. According to [55], their proposed adaptive support weights idea is essentially a naive implementation of the Bilateral Filter [62]. Yoon and Kweon refer to the gestalt principles of perception, in particular grouping by proximity and similarity. The weight of pixels in a support window is determined by their color similarity and their distance to the center pixel. Modified versions drop the proximity-term [14] or include additional gestalt principles ([53] add a 'discontinuity' term). However, the weights lack reusability because they fully depend on the center pixel. This makes these methods quite computationally expensive, especially for large window sizes as proposed in [73]. Min et al. [49] show how sparse samples of the support window can be used instead, in order to reduce the computational load. A similar weight distribution related to color segmentation, with emphasis on connectivity, can be computed by employing geodesic distances as proposed by Hosni et al. [37].

More recent edge-aware filters include the Guided Filter [29] and the Cross-based Local Multipoint Filter [23]. Both improve the quality of stereo results while being faster than the adaptive weights approach by [73] and the Bilateral Filter [62]: their runtime is linearly dependent on the image size, but independent of the filter kernel size. Rhemann et al. showed the Guided Filter's suitability for real-time stereo applications in [55]. Another factor in the popularity ([40, 55, 45]) of the Guided Filter is its straightforward implementation.

3.2.2 Hierarchical Constraints

To a degree structural ambiguities can be solved by cost space smoothing: if the majority of pixels falling in the support area is matched correctly, a wrong, ambiguous match will be smoothed out. However, this does not work if the distance between repetitions is larger than the size of the correlation window. To distinguish several similar image patches a larger window size covering more context (i.e. background) would be needed, which is a strategy leading to fattened or blurred boundaries, as already pointed out. Additionally, it is possible to miss the desired effect entirely for repetetive objects spaced far apart: surfaces of the respective background covered by the correlation window will be shifted in the search window to a point where they do not serve their purpose to 'anchor' the repeating patch anymore, as illustrated by figure 3.5.

The disparities computed by global approaches are ahead of local approaches considering repeating structures and large image regions with poor texture. They favor smooth disparity maps and solve the problem by global optimization, so for

Figure 3.5: *The background of the mirror shifts between left and right image. The reference window (top) captures a dark background. The search window for the correct match (bottom) captures a different, bright background. (Detail of Motorcycle image pair from [7])*

repeating (self-similar) patches in the reference image, the respective patches in the search image will be assigned in a way that minimizes the global costs. This means, the decision for a match is not only dependent on the individual matching cost, but also on the matching costs of all other parts of the pattern.

Hierarchical organization of image information, for example using multi-scale approaches or image segmentation, can improve the disparity map quality computed by local methods. Global methods greatly benefit from these techniques in terms of complexity and runtime.

Multi-scale approaches work on the original image plus (several) downscaled versions of it, which can be for example a Gaussian Image Pyramid. Early methods [66] perform the correspondence search on coarser scales first, and derive a hard disparity limit for the search range on finer scales. A huge disadvantage is the loss of details: if details with a large disparity (foreground) vanish in the downscaled image, the disparity bound is determined solely by the background objects (fig. 3.6). Either there are additional mechanisms to detect and recover from those errors [21] or the multiscale information is applied less restrictively. Min and Sohn [50] use the disparity computed for coarser scales as an initial value for their iterated refinement approach to speed up convergence. A similar approach can be found in the global methods proposed in [71] and [72]. Zhang et al. [75] stabilize the cost volume by incorporating inter-scale consistency without loss of detail. Their cross-scale aggregation method is detailed further in chapter 4.

Segmentation-based constraints are also popular with global methods as it is

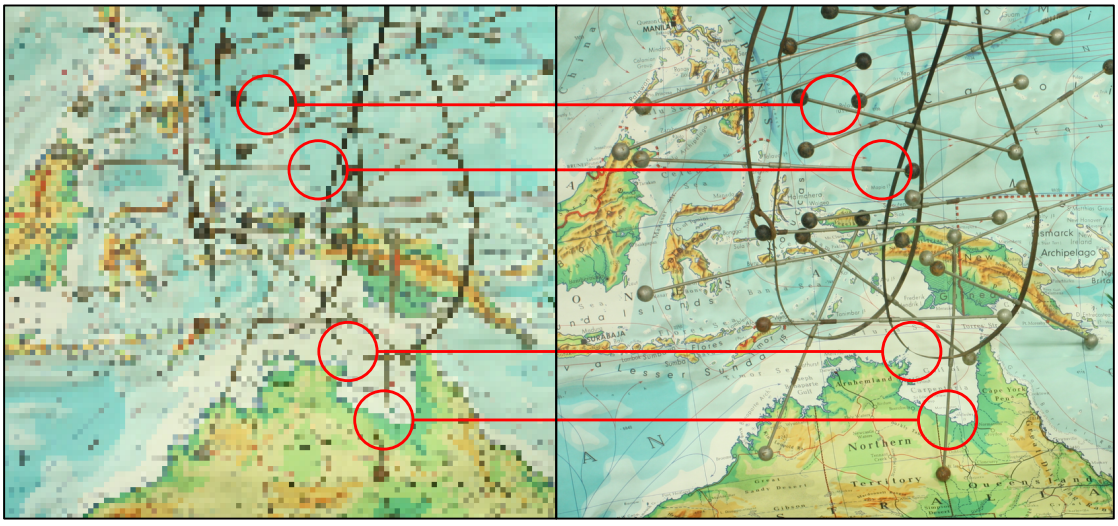


Figure 3.6: *If the disparity search range is determined by matching on coarser scales first, the disparity of the foreground details will not be considered on finer scales. (Detail of Australia image pair from [7])*

a nice way of reducing complexity while introducing a long-distance constraint. These methods tend to suffer from similar problems as multiscale approaches: detail is lost when the hierarchy-derived constraints are too rigid. This drawback become visible when for example all pixels in a segment are forced to lie on the same disparity plane [13, 35, 42, 68]. Countermeasures include extreme oversegmentation and fitting of a curved disparity surface to each segment [15, 74]. In [15] the employed segments are overlapping, to soften the constraint. Zhang et al. [74] regulate the overall effect of the segmentation constraint with a dedicated parameter. In [69] and [45] the segmentation is used for generating disparity proposals, that are refined further. In the method proposed by Huang et al. [39] segments reduce the search scope in a coarse-to-fine manner and, as in [45] and [47], accelerate the cost aggregation step.

3.2.3 Distinctiveness and Confidence

The mentioned hierarchical schemes try to prevent ambiguities that might arise in the matching process. Another strategy is to discover the actually present ambiguities first and only focus on resolving them.
Two terms are useful in the context of these strategies. The first is the notion of 'distinctivess' which is the opposite of 'ambiguousness'. A distinctive point has unique features with respect to points of the same group (e.g. the same scanline or the same image), a distinctive match has no competitors close in matching quality,

i.e. matching costs.
The second term is 'confidence' which expresses how reliable an assumption is. In stereo matching confidence for a point or match is used interchangeably for distinctiveness, although in general the two have to be distinguished. For example, if the distinctiveness is determined based on a small subsample of the available data, a point or match might be very distinct, but the confidence in this case would be low. Manduchi and Tomasi [46] compute cost-function related distinctiveness and confidence for all pixels before the stereo correspondence search. They match the pixels of the input images with themselves. Distinct pixels will have a minimum for a match with themselves, but high costs for all other points. Ambiguous points will have several minima close in cost values. The authors reason that distinct points are more likely to have distinct and thus high-confidence stereo matches, while expecting the opposite for ambiguous pixels. This leads to another type of hierarchy: matching the most reliable (distinct) points first. Furthermore, applying the ordering constraint, the reliable matches reduce the search space for matching the ambiguous points. To illustrate this, Manduchi and Tomasi sketch a stereo scenario, where a colorful bird in front of repetetive foliage is shown. The bird pixels will be matched first, and based on the initial ordering the disparity search for foliage pixels will be restricted to either the left or the right of the bird.

Saygili et al. [57] propose the adaptive combination of different cost measures based on their individual confidence for a region. By relying on the cost measure producing the least ambiguous cost profile per region, they reduce the overall ambiguity. A survey and evaluation of several confidence measures can be found in [52].

A widely used confidence test is the left-right cross-check. The disparity maps of both left and right input images are compared against each other pixelwise. If the disparity difference of pixel p and its matching point p' is larger than a threshold d_{max}, which is typically 1, p is labelled invalid. In short, the condition tested is:

$$|d_p - d_{p\prime}| \leq d_{max}$$

It is a simple and computationally cheap way of identifying invalid disparities, commonly done as part of the postprocessing step, but it can be conducted in-between iterations, too [24].

3.3 Increasing the Depth Precision

Digital images are a discretized projection of a scene. Many stereo matching algorithms only match pixels with pixels [57, 40]. The computed disparities in this case are discrete values, which makes it possible to cover all possible combinations

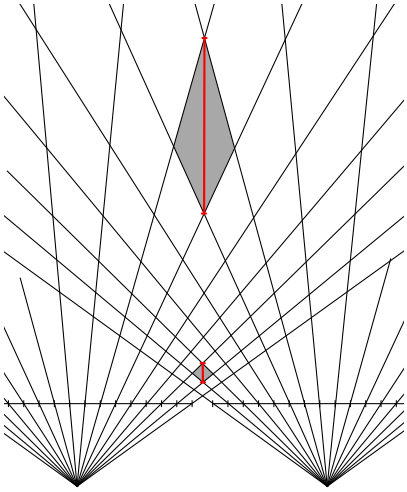


Figure 3.7: *Small disparity values cover a greater range of real depth than larger disparity values. This means, the depth of objects further away from the cameras will be less precise than the depth of closer objects.*

of pixel matches together with their costs - this is called the cost volume. However, this is not very close to real-world scenarios, where depth is a continuous quantity. The resulting 'staircase' effect was illustrated earlier in figure 3.4. On top of that, the depth resolution decreases the further the objects are located from the camera origin, as explained by figure 3.7.

The following sections will show why 3D-labels deserve to be called the workhorse of increasing depth precision, whereas the related 3D-support windows could be viewed as a useful byproduct.

3.3.1 Continuous Disparities

There exist different strategies to increase the depth resolution. One is to compute integer disparities first and then refine them to sub-pixel disparities in the disparity computation [31] or a postprocessing step [72] by interpolation. A slightly improved method increases the size of the discrete search space by considering a number of fixed sub-pixel steps, for example steps of 0.25 between pixel positions [21]. Although an exhaustive search is theoretically still possible here, it soon becomes practically infeasible with increasing image resolution and finer sub-pixel steps.

3D-label stereo methods go even further and try to estimate an optimal 3D-surface per pixel, from which the disparity can be computed. But switching to

continuous disparities comes with two problems. Firstly, we only have discrete input. To access pseudo-continuous (sub-pixel) data, intensity values are interpolated. Secondly, and this is by far the major problem, the search space becomes infinite. As a consequence, we cannot compute all possible disparity configurations (i.e. the cost volume) and choose the one associated with the lowest cost. How can a good minimum still be found in practice?

Zhang et al. [77] employ an integer-disparity-based method first and fit 3D-planes in this intermediate result. The planes are used to refine the integer disparities to sub-pixel disparities.

The publication of the (Generalized) PatchMatch method [11] made it possible to estimate 3D-planes from scratch in an efficient manner. This method fueled the creation of a wide range of sub-pixel precision approaches ([12, 38, 44, 45, 14, 30, 70]).
PatchMatch is an algorithm that smartly traverses the infinite search space, as Bleyer et al. [14] put it. In its essence, PatchMatch is a center-biased random search. PatchMatch converges towards a good label by testing randomly generated label hypotheses of exponentially decreasing distance to some initial label. The few large-distance label hypothesis tests prevent PatchMatch from being stuck in a local minimum, whereas allowing fine adjustments in case the current label is already close to the optimum. Another apsect of its efficiency is grounded in the crucial observation that, as mentioned in section 2.2, disparities vary smoothly in a scene (minus depth discontinuities between different objects). Neighboring pixels are very likely to share the same or at least a similar label, which is also exploited by PatchMatch when generating new label hypotheses. More details of the algorithm can be found in section 4.1.

3.3.2 3D-Support Windows

As pointed out in section 3.2, the support region around a pixel is used to smooth the noisy cost. A lot of research is concerned with finding the optimal shape or weights for the support region, but almost all these solutions assume constant depth for the support region. This assumption holds true only for scene surfaces with the same orientation as the image plane, i.e. fronto-parallel surfaces. For all other surfaces, especially highly slanted ones, the support pixels around a pixel of interest will be at a different depth. Thus, they cannot properly 'support' the pixel's current disparity hypothesis, because their matching cost will of course be higher for a wrong disparity.

Bleyer et al. [14] claim to be the first to explicitely employ slanted (3D) support windows. The more general term '3D-label stereo' was coined by [19]. 3D-label stereo can remove the fronto-parallel bias in cost aggregation (fig. 3.8), because it

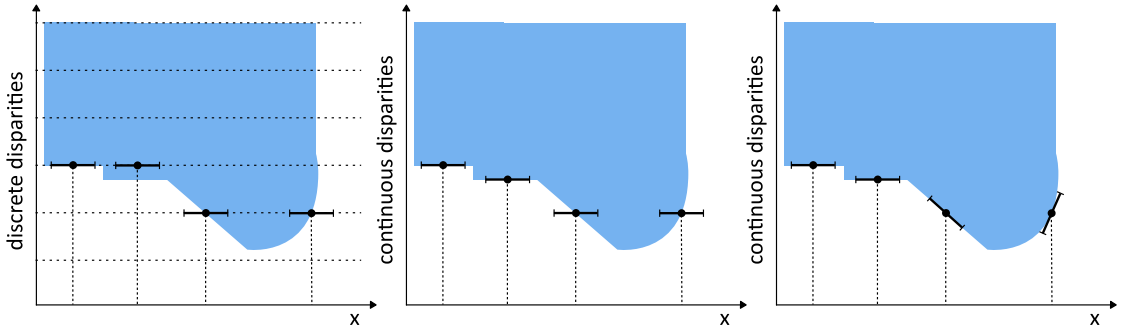


Figure 3.8: *Left: Discrete disparities and fronto-parallel support windows favour fronto-parallel surfaces that fall exactly into the depth steps. Center: The extension to continuous disparities can handle more fronto-parallel surfaces, but still produces artefacts for rounded and slanted surfaces. Right: Continuous disparities in combination with a 3D-support window can handle planar surfaces in all orientations and approximates rounded surfaces better. (Adapted from [14])*

can be naturally extended to generate 3D-support windows.

As an example consider the assumption that the scene can be approximated by 3D-planes, as proposed in [14]. The objective is to assign each pixel an optimal 3D-plane. When gathering support for an arbitrary plane (i.e. disparity) hypothesis, the support pixels are assumed to lie on this exact plane. The resulting disparities vary per support pixel according to the plane parameters.

3D-support window approaches range from choosing from a fixed number of slant hypotheses [10], over an infinite number of arbitrary slanted 3D-planes as in the example, to other arbitrary, more complex 3D-surfaces [74, 15]. The trade-off is between disparity map quality and computational effort, which depends on the number of dimensions of the searched parameter space.

Chapter 4

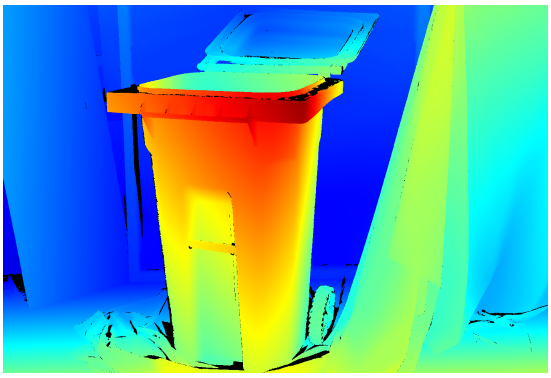
Proposed Method

From chapter 3 it can be concluded which properties in local stereo methods are most favorable for approaching the solution quality of global methods.

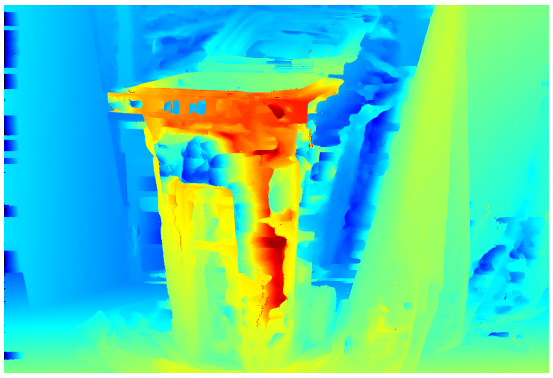
The method should produce high-precision, i.e. sub-pixel, disparities. Preferably, the matching itself should be precise and not just 'attached' by interpolation in a postprocessing step. One possibility is to estimate a 3D-surface per pixel. The method should make use of fast edge-aware image filters, as cost aggregation schemes using those perform best regarding speed and quality. Despite focussing on quality rather than speed in the context of this work, time complexity becomes important when working with high-resolution images and an infinite search space. The method should also be able to handle the pitfalls of most local methods: repetitive structures and large textureless regions. Other minor practical requirements include easy exchange of certain algorithm components and robustness to parameters, so users without comprehensive knowledge of the implementation details can still achieve high-quality results.

This chapter will present a new dense stereo correspondence method. The components that fit the design objectives above are explained in detail in the following sections. The PMF approach is discussed first, serving as main framework of the new method. The addition of the hierarchical cost aggregation scheme proposed by Zhang et al. [75] will provide stabilization for the proposed method. The propagation of promising labels is improved by employing a variation of Heap-PatchMatch as proposed by [11]. In an effort of further improvement other mechanisms, such as exploiting self-similarity in image patches, were introduced to the proposed method, but remain experimental artefacts for now. Although the application of these ideas is not fully fleshed out, working on them produced valuable insights and ideas, which are summarized in chapter 6.

(a) *The left image of the Recycle image pair from [7].*



(b) *The ground truth for the Recycle image pair from [7].*



(c) *The final disparity map of the Recycle image pair as computed by PMF.*

Figure 4.1: *The textureless blue plastic of the Recycle image pair as well as the uniform back wall pose a challenge for the PMF method (as defined in section 5.2.2).*

4.1 PMF-S Structure

PatchMatch Filter (PMF) by Lu et al. [45] is a framework for multi-labelling problems that fulfills most of the requirements on the feature list. It combines edge-aware filtering with efficient traversal of an infinite solution space using the popular PatchMatch algorithm.

In the context of stereo correspondence search, the authors propose two algorithms to label the input pixels: PMF-C, assigning a discrete ('constant') disparity per pixel, and PMF-S, assigning a slanted 3D-plane (determined by three parameters) per pixel. The case of PMF-C will not be considered any further, because PMF-S generates disparity maps with higher depth precision and thus superior quality.

The shortcomings of the original PMF based algorithms become visible in using more challenging image pairs containing repetitive structures and large textureless regions, which are not handled correctly (see fig.4.1). The method proposed here will take the components of PMF-S as base, the structure of the method will be

summarized in the following sections.

PMF-S is a local, iterated stereo algorithm with the goal of assigning a 3D label, i.e plane parameters, to each pixel of the stereo input images. The respective sub-pixel precision disparity map can be computed from the plane parameters using the following relations (after [14, 45]). Each pixel $p = (x_p, y_p)$ is assigned a plane, defined by the 3-element label vector $l_p = (a_p, b_p, c_p)$. The position of the corresponding pixel in the other view $p' = (x_{p'}, y_{p'})$ is given by

$$\begin{aligned} x_{p'} &= x_p \pm d_p = x_p \pm l_p \cdot [x_p, y_p, 1]^\top \\ y_{p'} &= y_p \end{aligned}$$

If p' resides in the right (left) view, the disparity value is substracted (added). Because rectified input is assumed, there is no vertical disparity.
To receive a support window for p aligned with the currently assigned label l_p, the support pixels q in a neighborhood N (a square window) are projected to the other view in the same manner:

$$\begin{aligned} x_{q'} &= x_q \pm d_q = x_q \pm l_p \cdot [x_q, y_q, 1]^\top \\ y_{q'} &= y_q \end{aligned} \tag{4.1}$$

PMF-S is closely following the PatchMatch Stereo [14] approach of Bleyer et al. but enhances the latter mainly in two ways:

1. The pixels are organized in meaningful groups (corresponding roughly to superpixels) as a means of enabling collaborative label search.

2. The cost aggregation in PMF-S is done by fast image filtering. The authors employ the Guided Filter (GF) [29] and as an alternative the zero-order cross-based local multipoint filter (CLMF-0) [23] instead of adaptive support weights (after [73]) as in PatchMatch Stereo, benefitting the algorithm in terms of both speed and quality.

The general structure is shown in figure 4.2. After a preprocessing step, the PMF-S algorithm runs a series of iterations encompassing cost computation, cost filtering, and label improvement. The final postprocessing step refines the disparity solutions.

4.1.1 Preprocessing

The input images are each segmented into k superpixels, which are organized as a graph (fig. 4.3). Each node represents a superpixel and each edge represents a 'neighbor' relation between two superpixels. Two superpixels are spatial neighbors

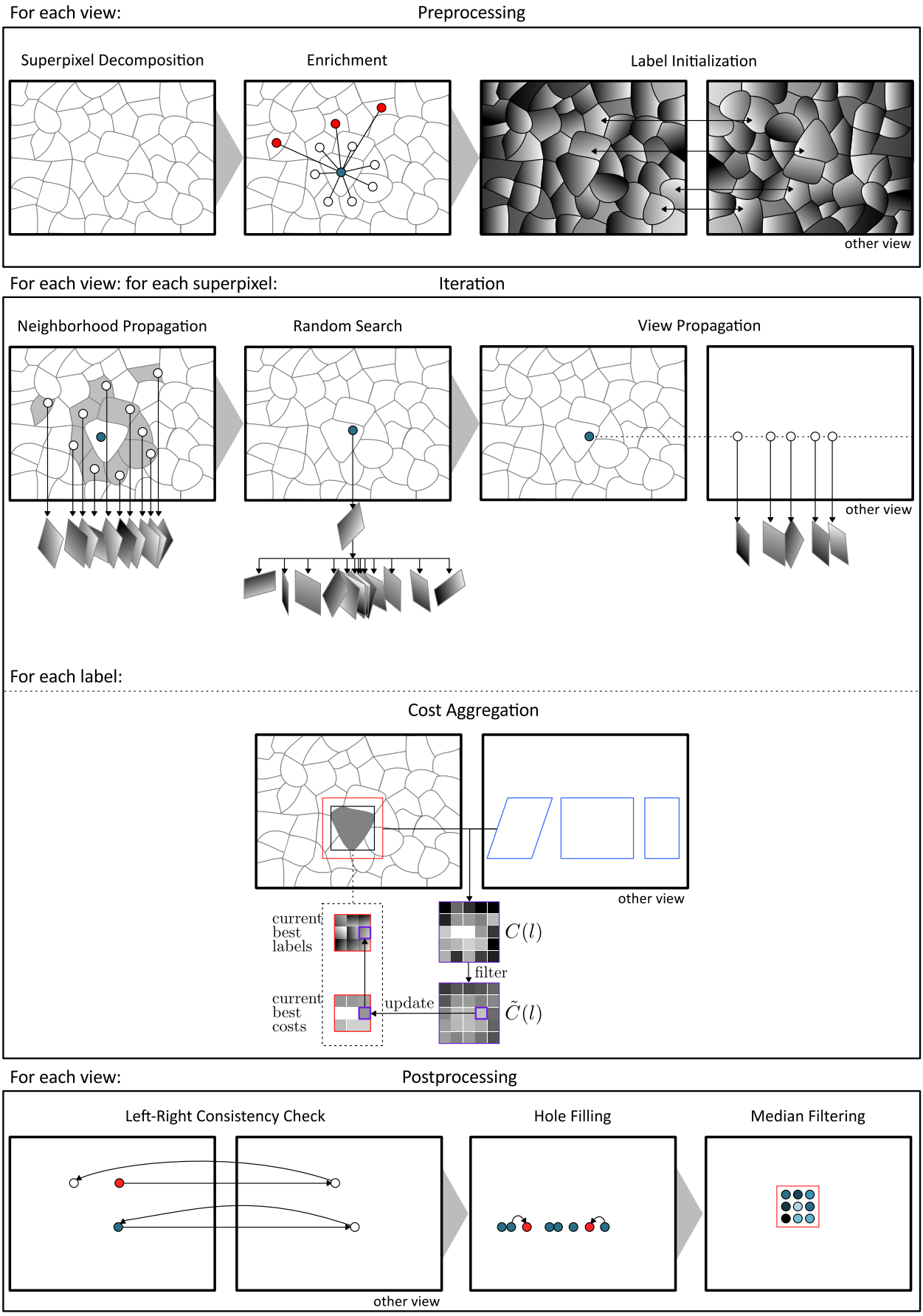


Figure 4.2: *The structure of PMF can be reduced to a preprocessing step, a number of iterations and a final postprocessing step.*

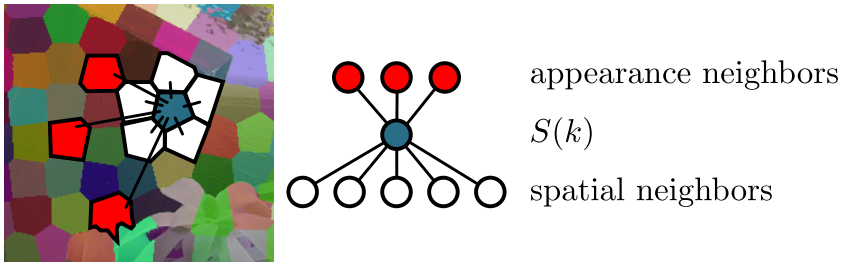


Figure 4.3: *Spatial neighbors share a boundary with $S(k)$. A maximum of κ appearance neighbors are determined by a loosely defined similarity to $S(k)$. They are also restricted to lie within a certain maximum path distance.*

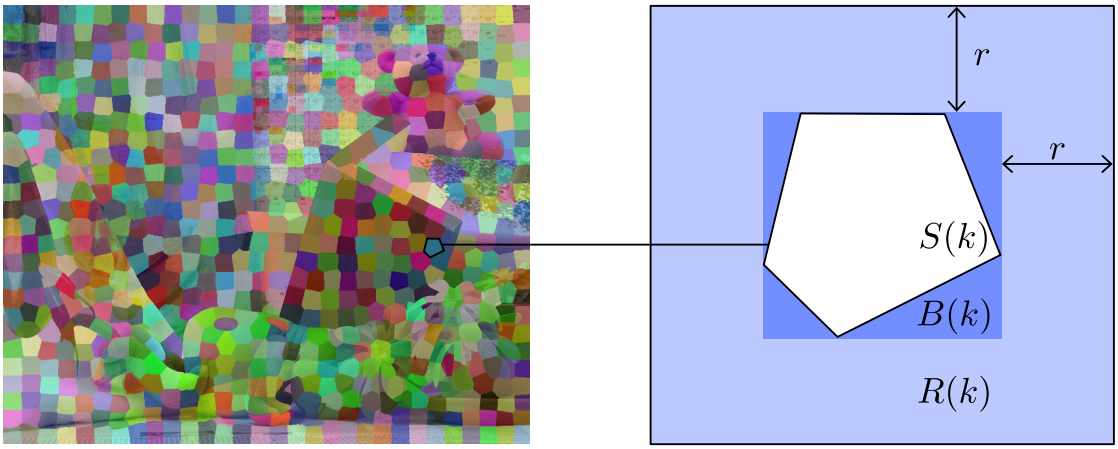


Figure 4.4: *Each superpixel $S(k)$ has an associated bounding box $B(k)$ and a subimage $R(k)$.*

if they share a boundary. Superpixels can also have a maximum of $\kappa = 3$ additional neighbors by appearance, as long as their spatial distance is not too large. The authors of [45] used a very large maximum distance for finding neighbors by similarity (2.5× maximum disparity value), I found it more practical to specify the distance in graph space, i.e. the maximum path length to reach a node (set to 3). The similarity is defined in a loose way, because the superpixels are generally not of the same size. Details can be found in the PMF paper. Despite a threshold for similarity being mentioned by the authors, no value is given. The corresponding value listed in appendix B.1 was determined empirically. In anticipation of the cost filtering step, for each superpixel $S(k)$ two auxilliary structures as seen in figure 4.4 are defined. One is the minimum bounding box $B(k)$, the other is a region $R(k)$ (also called 'subimage'), which is $B(k)$ padded by r pixels (limited to lie within the input image). The authors employ the superpixel segmentation algorithm SLIC [9]. SLIC is based on a randomized k-Means clustering and can be parameterized

to produce quite regular-shaped, compact superpixels - this property ensures tight bounding boxes $B(k)$.

Furthermore, each pixel is initialized with a random label or alternatively a label estimate by loosely matching superpixels across images. The same approach as for finding additional superpixel neighbors by appearance is used here.

4.1.2 Iteration

The iterated part of the method is responsible for computing and aggregating the costs for various labels as well as assigning each pixel the respective best scoring label. Since the label search space (3D-planes) is infinite, it is impossible to simply check each value. PMF-S makes use of some interlocking strategies to

1. smartly guess only 'promising' labels,
2. efficiently test these labels simultaneously for a number of pixels, and
3. propagate good labels across large distances.

The generation of promising labels is done by the PatchMatch method, which exploits the observation that spacially close pixels often share similar labels. The three PatchMatch search strategies are the following:

Neighborhood propagation Label guesses are generated by choosing a random current best label from all neighboring entities. In the case of PMF-S this implies one label per neighboring superpixel, in PatchMatch Stereo neighbors are single pixels.

Random search New label guesses are generated by a random exponential search around a randomly chosen current best label l_p of the respective superpixel. To allow the search space to be sampled evenly, [14] use a point P and a normal $\vec{n} = (n_x, n_y, n_z)$ to describe the planes. The random search at a pixel $p = (x_0, y_0)$ with a current best label l_p starts with defining the allowed maximum change for the disparity $[-\Delta_z^{max}, \Delta_z^{max}]$ and the normal vector components $[-\Delta_n^{max}, \Delta_n^{max}]$. The values drawn randomly from these intervals (one for the disparity distance and one for each normal vector component) are added to the start values given by l_p, giving a new label (plane). This random search it iterated, after every iteration the allowed maximum distances are reduced by a factor, e.g. halved. The stop criterium is some minimum value for the distances. Because the search distance is decreased exponentially, the set of new label guesses is biased towards labels similar to l_p. While the few large-distance jumps in the first random search iterations prevent the method from staying stuck in a local minimum, the greater number of only

slightly noise-disturbed labels can refine l_p in case l_p is already close to the optimal solution.

View propagation Also called 'inverse enrichment' in [11], in the context of stereo correspondence new label guesses can be found in the current best labels of the other image. Choosing a random pixel p (within the superpixel) in the reference image, new label guesses are all labels of the search image that result in a match of their respective pixel with p.

These three strategies (recall fig. 4.2) help to smartly guess good labels and also propagate them across large distances. A notable detail is that the random search is the only step that can introduce new label values. The neighborhood and the view propagation just compile a subset of already assigned labels.

The PMF-S specific organization of data into superpixels also enhances the fast label propagation across the image, but most of all enables efficient testing of label hypotheses, i.e. not pixel-wise as in PatchMatch Stereo, but superpixel-wise. For each label guess l the raw costs are computed for all pixels of the support region $R(k)$ of the respective superpixel $S(k)$ resulting in a raw cost slice $C(l)$. The used cost function is the sum of truncated color and truncated gradient differences as explained in section 3.1. The raw costs are computed pixel-wise, but because 3D labels result in a slanted support region and floating point disparities (see equ. 4.1, illustrated in fig. 4.5), the matching pixels' positions and intensities are linearly interpolated.
$C(l)$ is then filtered with the chosen edge-aware image filter. The filtered output denoted with $\tilde{C}(l)$ is used to update the current best pixels' labels: the current best label l_p for pixel p is updated with the new label l, if the new cost for the pixel $\tilde{C}(p, l)$ is lower than the current cost $\tilde{C}(p, l_p)$. To emphasize, this means the label is decided per pixel and not imposed by fitting the superpixel to one label. The superpixel graph structure is just used for organizing the pixels into groups with a (supposedly) similar enough labelling to serve cost aggregation and to speed up the label search.

Despite the whole region $R(k)$ being filtered, only pixels in $B(k)$ are updated with the filter result $\tilde{C}(l)$, because pixels outside $B(k)$ do not receive enough support to make a reliable label decision. The width increment r for constructing the region $R(k)$ is derived from the filter kernel radius r for this reason.

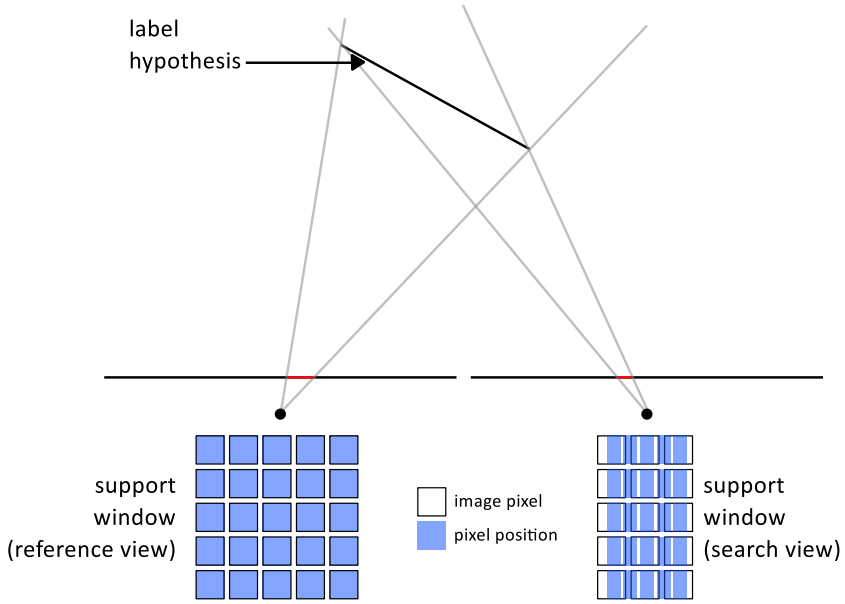


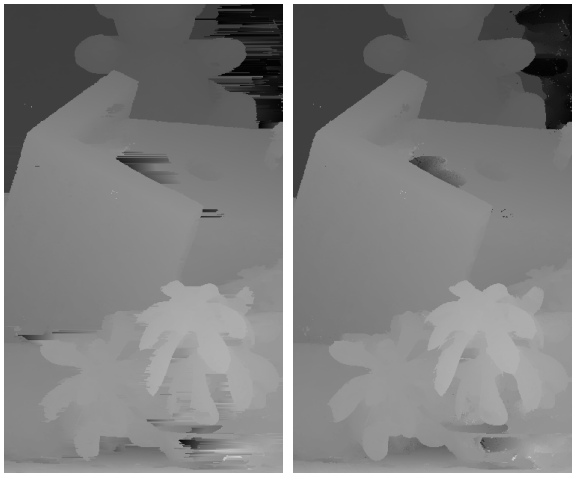
Figure 4.5: *The support window in the reference view is square, but because it assumes covering a slanted plane, the corresponding pixel positions are transformed accordingly (to sub-pixel positions), as viewed from the search view.*

4.1.3 Postprocessing

After a number of the above iterations, the disparity maps for both stereo images are postprocessed by first conducting a left-right cross-check to determine unreliable disparities. An appropriate correction is the assignment of a valid label in the neighborhood. Typically, most of the pixels failing the left-right disparity cross check lie in occluded regions. Occlusions originate in the horizontal camera parallax, so the candidate filling labels are searched in the same scanline as the invalid pixel (not vertically). Then, the label corresponding to the smaller disparity is assigned as the new valid label, since occluded regions are likely to be background regions. Finally, a small weighted median filter is applied to reduce the streaking artifacts as seen in figure 4.6 caused by the filling step.

4.2 Cross-Scale Regulation

Considering hierarchical image pyramid approaches, detail preservation has to be assigned a high priority. In restrictive hierarchical schemes, the disparity search starts with coarser scales, handing the information up to finer scales. The cross-scale cost aggregation method by Zhang et al. [75] works in the opposite direction. They use the matching cost across the image pyramid in a non-prohibitive way by



(a) *After filling* (b) *After median filtering*

Figure 4.6: *Detail of disparity map for Teddy pair: 4.6a shows streaking artefacts after the filling step that are reduced in the final disparity map 4.6b.*

starting at the finest scale and aggregating costs going down the image pyramid to the coarsest scale.

Cost Aggregation

The proposed method incorporates the cross-scale regulation approach when testing a label hypothesis. The employed image pyramid is the widely used Gaussian Pyramid, built by subsequent blurring and subsampling the input image (see fig. 4.7).

The raw matching costs are computed per scale s, resulting in several cost slices $C_s(l)$. The size and position of the support window is constant, only the disparity search range is affected by the scaling. This means, in cost slices of coarser scales a larger image area is covered than in those of finer scales. The filter kernel size and the size of the raw and filtered cost slices are fixed over all scales. The robust cost slice at the finest scale $\hat{C}_0(l)$ combines the filtered cost slices $\tilde{C}_s(l)$ of all scales by computing their (normalized) weighted sum. As illustrated by figure 4.8, the positions of pixels in the finest scale are traced across scales, so only a subset of filtered costs in coarser scales is used (center region), the rest (border region) is discarded.

The further processing of the robust cost slice $\hat{C}_0(l)$ is done as described in the PMF-S Iteration step, i.e. pixel labels are updated if lowered costs occur.

Figure 4.7: *The image pyramid used for the left image of the Teddy image pair.*

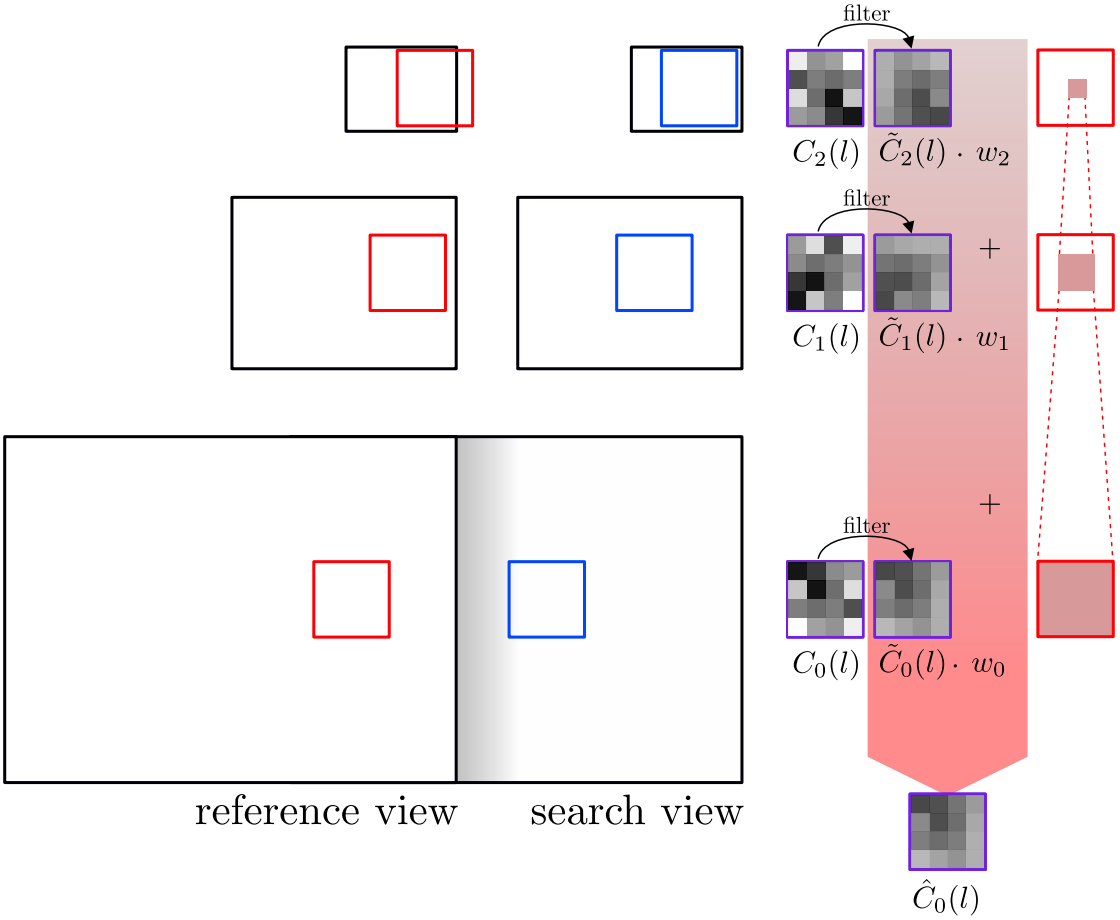


Figure 4.8: *The raw cost slices $C_s(l)$ are computed per scale and filtered. Their weighted sum is the cross-scale regulated cost slice $\hat{C}_0(l)$.*

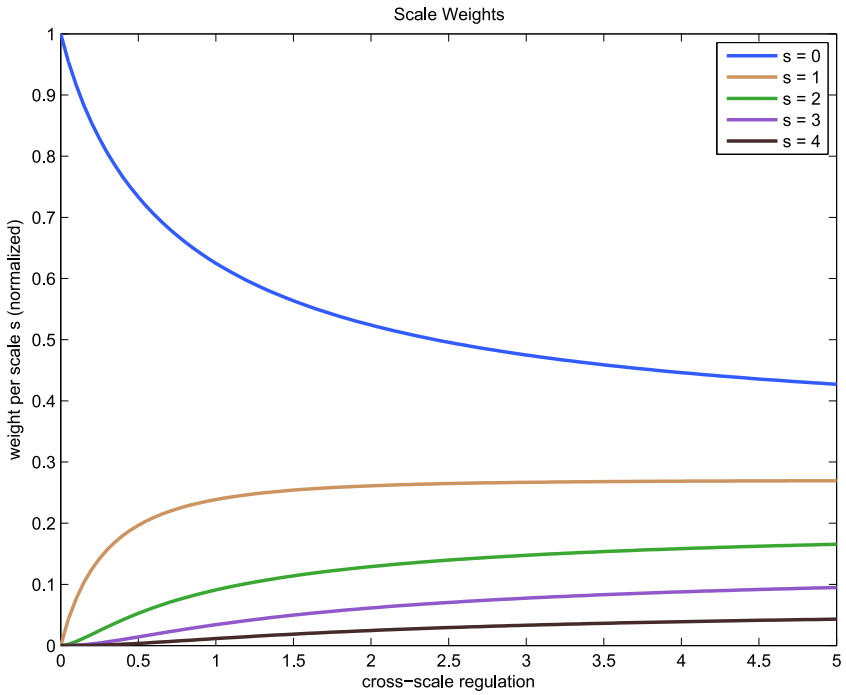


Figure 4.9: *The effect of λ on the (normalized) weights for each scale assuming five scales: the weight of a scale is always greater than the weight of its next coarser scale. Scale 0 corresponds to the finest scale (highest resolution), scale 4 is the coarsest scale (lowest resolution).*

Scale Weights

In [75] Zhang et al. reformulate the the cost aggregation step as a noise minimization problem (as described in section 3.2, the raw costs are noisy), but over multiple scales. This isolated minimization per scale is only useful for approaches that e.g. restrict the disparity search space. To enforce a regularization between different scales a supplementary condition is added to each sub-problem optimization using a Generalized Tikhonov regularizer. The strength of this cross-scale regulation is determined by the parameter λ. A detailed derivation of the scale weights computed from λ can be found in the corresponding paper, for the scope of this work it suffices to say that all regularized cost slices (denoted by a vector $\hat{v}$) are related to all filtered cost slices (denoted by a vector $\tilde{v}$) by a matrix A, that is derived from the regularization strength λ: $A\hat{v} = \tilde{v}$ solving for $\hat{v}$ gives $\hat{v} = A^{-1}\tilde{v}$ Due to its form, the matrix A is always invertible, symmetric, and constant since it only depends on λ. Because we only need the first element of $\hat{v}$ (corresponding to the cost slice of the finest scale) to be regularized, the respective scale weights used in the proposed method are found in the first row of A^{-1}.

For five scales as used in [75] the effect of λ on the scale weights is summarized in figure 4.9. Further examinations of the cross-scale regulation are carried out in chapter 5.

4.3 Heap-PatchMatch

In their paper [11] Barnes et al. proposed several generalizations of the PatchMatch method, one of which enables PatchMatch to find k nearest neighbors as opposed to just 1 nearest neighbor per image patch. In the simplest variant of organizing several nearest neighbors per image patch, the k nearest neighbors are stored ordered by their distance to the currently considered patch. This priority list (the authors use a max-heap) is updated as better i.e. closer nearest neighbors are visited by the algorithm. For each random search step promising new nearest neighbor positions are generated from all k stored neighbors instead of using only one.

This Heap-PatchMatch algorithm cannot be applied directly to the method proposed here for the following reason: the nearest neighbor positions in the original PatchMatch are discrete pixel positions while the PMF-S method determines sub-pixel positions. It is highly likely that the k best labels would be very close variations of the current best label of the list, which would hardly benefit the random label search at all. The candidate labels stored for one position should have a minimal distance from each other to contribute to the search.

To realize this condition, the labels of the heap are categorized into k 'buckets'. Each bucket is responsible for holding the label with the lowest cost within a certain disparity range. For each new label the corresponding bucket is determined. Within the same bucket the new label replaces the label with higher cost. If the number of used buckets, i.e. the heap, is limited, the heap may be pruned after a new bucket was added. In all other cases the new label is discarded - even though it might have lower costs associated than the label in the last ranked bucket. This strategy maintains a certain variety in labels for starting the random search.
The defining parameters are the number of used buckets k and the disparity range covered per bucket. One minor drawback can be noticed if the optimal label falls into the border region of its bucket: the bucket next to this region will probably hold a label very close to the optimal label, meaning the list is essentially shortened to $k - 1$ useful buckets.

4.4 Exploiting Self-Similarity

Evaluating self-similarity of image regions in stereo matching is not a new idea. Manduchi and Tomasi [46] analyze and exploit the distinctiveness of pixels: the most distinct pixels are matched first and these matches limit the disparity range for the more ambiguous pixels, assuming a constant ordering in both images.

The approach presented here differs from [46] in the way that not one (distinctiveness) value is assigned per pixel, but a 'pattern' encoding the similar patches around the patch in question.

This idea developed from one of the seasoned gestalt principles of perception that objects are grouped by similarity. Treating repeating patterns as an entity is a strength of the human visual system: pickets are clustered to a fence, a lot of similar stones form a wall, the face of a skyscraper is made of a grid of single windows.

In stereo matching a repetitive pattern in one image is likely to be present in the other image, too. For example, if the pixel is part of a picket on the lefthand part of the fence, it will also be on the lefthand part of the fence in the second image. If we could detect the pattern beforehand and formulate pixel positions relative to their pattern, maybe we could restrict the pixel position in the other image upon detection of a similar pattern. To repeat the thoughts of [46]: the relative position within a pattern is useful only in case the ordering constraint is not violated, i.e. the pattern elements lie approximately on the same depth plane.

In practice, this plan can be subdivided into several subtasks, namely detection, description, and matching implications for each pattern.

Pattern Detection

The goal is to detect patterns in the cost function profile of an image patch that will emerge also when performing the stereo correspondence search. For a straightforward self-similarity test the input image is matched with itself using the same cost computation and cost aggregation method as in the stereo matching procedure. The self-similarity is the inverse of the matching cost: a high cost between patches corresponds to high dissimilarity while a low cost corresponds to high similarity. The self-similarity is analyzed per scanline, because predominantly horizontal ambiguities interfere with a correct disparity estimation. For each pixel, called the reference pixel, a characteristic ambiguity pattern is determined by looking at peaks in the self-similarity results. The respective reference peak is found at the position of the reference pixel. The most significant of the remaining peaks determine the position of similar image patches. The easiest way to distinguish the insignificant peaks from those peaks that constitute a meaningful pattern is to use a fixed threshold. Explorative test runs showed noisy patterns after this

thresholding. So in a primitive effort to further stabilize the pattern, clusters of similarity peaks close in horizontal distance were merged to one pattern element. This is done because we can rely on the edge-aware cost filtering step to resolve ambiguities within a certain range (depending on the filter kernel size).

Because the stereo matching is constrained to k 3D-labels per pixel (if we employ the Heap-PatchMatch extension with k buckets), the pattern should consist also of at most k elements. Extracting a larger pattern in the self-similarity matching of the reference image would be possible, if for example discrete pixel positions for a match are assumed. However, for finding a pattern of similar structure when stereo matching, the method also needs to track the k best 3D-labels for the search image, which contain three floating point values each and thus cost more memory to store. A large k would quickly become infeasible, although there might be other approaches to tackle this problem. For tests with pattern extraction a small k was used in addition to low resolution images, cropped to include repetitive details.

Pattern Description

Because we are only interested in the relative position of a pixel or image patch within a horizontal pattern and assume a constant ordering with respect to the search image, the pattern for a pixel is shortened to the number of lefthand pattern elements and the number of righthand pattern elements. Further considerations could include additional information, such as the confidence of the respective element based on its self-similarity value or distances between pattern element positions.

Pattern Matching

The self-similarity pattern shall serve as an additional constraint for determining the correct match for the reference pixel. The Heap-PatchMatch component finds the k current best labels with respect to the search image. Let us assume this set of disparities represents the reference pattern as seen in the search image (including the correct match for the reference pixels) and denote it search pattern. The search pattern is naively stabilized in the same manner as the reference pattern. The elements of the reference model and the search model are shifted towards each other to find the minimal costs configuration (see fig. 4.10). These costs result from the label-associated costs where the patterns overlap plus maximum costs for unnecessarily ‘overhanging’ pattern elements.

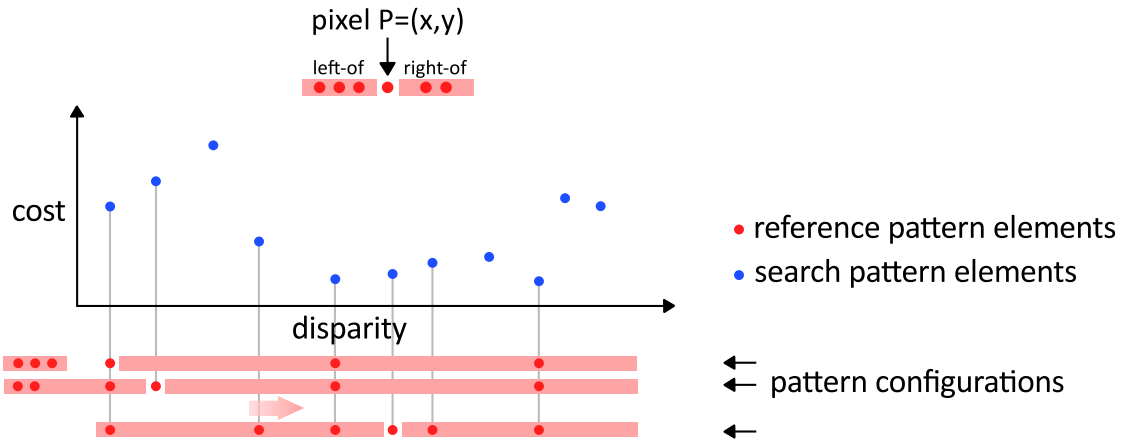


Figure 4.10: *The elements of the reference pattern are distributed so the relative ordering is preserved and the costs of the respective search elements is minimal over all reference pattern elements.*

Conclusion

There certainly exist smarter approaches to derive a general pattern from the data points and to integrate it into the matching process. The explorative experiments concerning the current rather premature approach revealed both flaws and potential in each of the three steps.

Investigating the ambiguity pattern idea and the accompanying open issues thoroughly unfortunately was not possible within the limitations of this work, so a lot of unsolved problems and questions are left for future explorations and listed in chapter 6.

Chapter 5

Experiments and Results

This chapter will cover the evaluation of the method proposed in chapter 4. The focus of the experiments will be the quality of the final disparity map, as opposed to runtime or memory requirements, which is more important for real-time algorithms. The used benchmark datasets and quality measures will be introduced in the first section. The following section will cover details to the conducted experiments. A final summary will complete the chapter.

5.1 Data Sets and Quality Metrics

The evaluation of stereo algorithms generally has different aspects, depending on the area of application. The most defining quality trait is the correctness of the computed disparity map with respect to the ground truth. Other properties of the algorithm can be evaluated, such as runtime performance (for real-time applications), or memory and computational complexity (for standalone devices with restricted storage and energy capacities). There may be special prerequisites such as the ability to cope with non-rectified input images or multi-modal input.

For the proposed method the disparity map quality will be evaluated. Properties such as runtime or memory load are of minor importance for now.

Obtaining appropriate, realistic input images with accurate ground truth is a complex task. Artificial 3D-scenes are a possible stand-in, but often do not provide enough realistic detail and noise (e.g. radiometric distortions), which will be present in target data acquired in practice. For this reason, many researchers rely on public benchmarking data sets which present realistic challenges and a carefully measured ground truth. Another advantage is the comparability of the different algorithms, if they are evaluated using the same data.

Some well-known benchmarks for stereo are the Middlebury stereo evaluation website [7], the KITTI vision benchmark [5], and the HCI robust vision challenge

[2]. The KITTI and HCI data sets contain stereo videos with a focus on outdoor scenes. They are clearly directed at real-time algorithms meant to be used for assisted and automatic driving. The Middlebury image pair data sets consist of close range indoor scenes with challenging scene details.

The proposed method will be evaluated using a subset of the Middlebury data sets, details will be provided by the following sections.

5.1.1 Middelbury Stereo Evaluation

Scharstein and Szeliski created the Middlebury stereo evaluation benchmark as part of their publication [60]. The data sets evolved over time and got more challenging:

The first data set (2001,[60]) consists of piecewise planar scenes. The 2003 data set, the Teddy and Venus image pairs (published in [61]), contain rounded surfaces and are more complex than the older data set. The 2005 and 2006 data sets [59, 33] are collections of more image pairs with ground truth, captured using the same technique as in [61]. The technique for obtaining the most recent 2014 data set is described in [58].

Researchers can upload their correspondence algorithm results, which are evaluated and ranked among the other stereo algorithms on the Middlebury stereo website. There are two evaluations publicly available, from which the older one (evaluation version 2) is no longer open for submissions. The evaluation version 3 is designed to overcome some flaws of the former version. The most important improvement is a separation of the test data set into test and training data sets. The ground truth of the training data set is available to researchers, for the test data set the ground truth is hidden - this prevents overfitting to the test data. Furthermore, for almost all image pairs of the evaluation data set (2014) there exist a 'perfect' and an 'imperfect' version. The imperfections include for example different scene lighting (a lamp in the scene is switched on in one image, but switched off in the other), different exposure between images of one pair, or an imperfect rectification. The version used in the ranking is indicated by the name of the respective image pair, e.g. PianoL is the same scene as in Piano, but with a difference in lighting of the scene.

All image pairs come in three resolutions: full (F), half (H) and quarter (Q) resolution. Full resolution for most of the image pairs ranges from 5 to 6 Megapixels. Two training image pairs adopted from older data sets (Art and Teddy) are smaller. For the online ranking it is not important which resolution is used to compute the disparity map, but the uploaded result is always compared to the ground truth at full resolution (and upsampled before, if necessary).

5.1.2 Error Categories

The quality metrics are specified in [60]. Evaluation version 3 offers more quality metrics than will be presented in the following short overview.

The error values that are given throughout this chapter are the percentage of bad pixels B for a disparity map. A pixel $p = [x, y]$ is labelled as bad pixel, if its disparity value d_p differs from the ground truth D_p. The allowed difference is determined by the error threshold $\delta_d \in \{0.5, 1.0, 2.0, 4.0\}$ with

$$B = \frac{1}{|N|} \sum_{p \in N} (|d_p - D_p| > \delta_d), \tag{5.1}$$

where N is the region containing all tested pixels. The region N can be the complete image ('all') or a reduced region, for example only containing nonoccluded regions that are visible in both views ('nonoccl'). The specific combination of the error threshold and the evaluated region will be called error category in the experiments. If not explicitely indicated otherwise, the analyses refer to the 'all' error categories.

5.2 Experiments

The original PMF as implemented by the authors of [45] was submitted to evaluation version 2. The method proposed in this work will be evaluated for its different additional components using more challenging image pairs. To save time and resources, the evaluation is carried out on a subset of the 2014 training data set provided by the Middlebury stereo evaluation benchmark. A full list of the reduced data set can be found in table 5.1, along with the biggest challenges for each pair. The evaluation results presented here are obtained computing the disparity maps at half (H) resolution and comparing it with the half resolution ground truth. There are no further details mentioned about the upsampling method used in the online evaluation, and no guesswork should be involved in the evaluation. Either way, it is unlikely that the results are severely warped by working at half resolution from start to finish.

The questions that shall be answered by experiments concern the importance of the segmentation in the preprocessing step and the effect of the cross-scale aggregation scheme on the quality. To a minor degree, the effect of adapting the cost function of the method should also be explored.

The values for fixed parameters that are not mentioned in the individual experiments' sections are listed in appendix B.1.

Image Pair	Left View	Challenges
Adirondack		• horizontally repetitive structure (chairback) • imperfect rectification
Jadeplant		• many occluded regions (twigs, lattice) • small, rounded objects (leaves) • imperfect rectification
Motorcycle		• thin structures (cables, handlebar) • highlights • imperfect rectification
Recycle		• large regions with weak texture (background, blue plastic) • imperfect rectification
Shelves		• large regions with weak texture (back wall) • horizontally repetitive structure (books, shelves) • imperfect rectification
Teddy		• highly slanted groundplane • foreground details (plant)
Vintage		• large regions with weak texture (background, white plastic) • reflections (screens) • imperfect rectification

Table 5.1: The data set for the evaluation contains a subset of the challenging, realistic images used in the Middlebury stereo benchmark.

5.2.1 Segmentation

The questions that shall be answered by the segmentation experiments are:

- How does the regularity of the superpixels affect the disparity map quality?
- How does the size of the superpixels affect the disparity map quality?

To answer them, 10 disparity map results of the correspondence algorithm per parameter setting will be captured and analyzed. The varied parameters are:

- The initial grid size of SLIC, denoted by $SLICsize$, which determines the approximate size of the resulting superpixels. $SLICsize \in \{20, 45, 70\}$ will be denoted $size_{20}$, $size_{45}$ and $size_{70}$. Results with identical $SLICsize$ parameter will be called $SLICsize$ groups.
- The regularity of the resulting superpixels $SLICreg \in \{0, 1, \infty\}$ where ∞ stands for replacing the SLIC superpixel algorithm with a regular grid, generated without considering any kind of pixel intensity information. The regular grid 'superpixels' at the borders are simply cropped to fit the image dimensions. These settings will be denoted by reg_0, reg_1 and reg_∞. Results with identical $SLICreg$ parameter will be called $SLICreg$ groups.

Effects of both parameters are shown exemplary in table 5.2. Other than these varied parameters the algorithm conforms to the basic PMF. The cross-scale aggregation is switched off, as well as the Census costs. The Guided Filter settings correspond to those mentioned in the PMF paper [45]: $\epsilon = 0.01^2$ and kernel radius $r = 9$. The data gathered with these experiments will also serve as baseline quality against which the later experiments considering the added functionality will be ranked.

The complete results can be found in form of a number table as well as graphical plots in appendix A.1.

Observations

- The mean errors considering 'all' pixels and the mean errors considering only 'nonoccluded' pixels behave similarly within each image pair. The 'nonoccluded' errors are slightly higher in every single case, which will be discussed later.
- The errors of one parameter setting behave overall similarly within each image pair across the different error thresholds, for example the relative ordering of errors of the $size_{70}$ group in the Adirondack data set is constant for all 4 error thresholds.

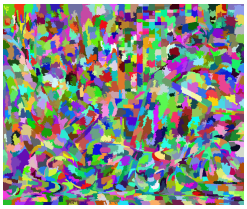
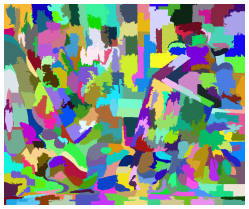
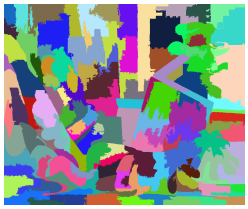
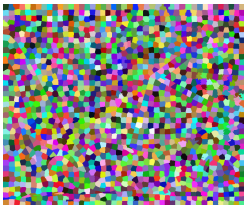
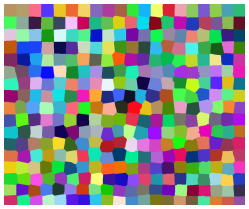
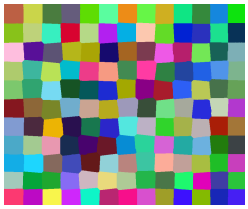
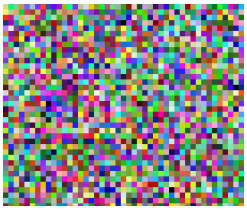
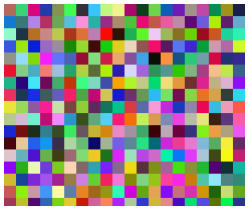
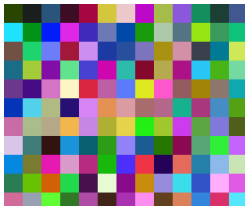
	$size_{20}$	$size_{45}$	$size_{70}$
reg_0			
reg_1			
reg_∞			

Table 5.2: The left image of the Teddy image pair is segmented using all tested parameter combinations.

- The standard deviation is small (≤ 2.18) for 5 out of 7 image pairs, suggesting a high reliability of the collected data. The other image pairs (Adirondack and Jadeplant) have standard deviations up to 5.10.
- Across all image pairs and error threshold categories, the results of reg_0 are the best within their $SLICsize$ group or at least on par compared to the other $SLICreg$ settings. For the reg_1 and reg_∞ settings the results are inconclusive with respect to the disparity map quality, but in one third of the $SLICsize$ groups both $SLICreg$ settings are on par.
- Across all image pairs and error threshold categories, in 11 of the 21 $SLICreg$ groups the errors get smaller with the superpixel size ($err_{20} \leq err_{45} \leq err_{70}$). 9 of the remaining $SLICreg$ groups show errors very close in value.

Conclusions

The reg_0 setting performs best in all error threshold categories. A reason for that could be the amount of 'overhead' filtering: The bounding boxes for irregular superpixels are on average much larger than those of more regular superpixels, leading to repeated filtering of the surrounding pixels that are covered by a lot of different bounding boxes. Despite no explicit runtime measurements, in informal tests a higher runtime could be observed with more irregular superpixels. The best $SLICsize$ in the experiments is $size_{20}$, which means more superpixels per image. Because the label generation and filtering happens per superpixel, there are more 3D-labels tested per iteration. A smaller size also increases the runtime. However, it has to be noted that the setting with the smallest, infinitely regular superpixels approximately has the same quality as the setting with large, most irregular superpixels.

The errors are slightly higher if only the 'nonoccluded' areas are considered - which is unusual. A possible explanation might be that most of the errors are not caused by occlusion but by wrong matching in the foreground. The hole filling step after the left-right cross-check is able to fill in the background correctly for the most part, but there might be regions in the foreground that are estimated wrongly in both disparity maps and, thus, are not labelled as invalid. Such a region can be seen in figure 5.1.

Although the parameter set includes extreme values for the size and regularity, the base-method (PMF) is able to produce surprisingly reliable results for most of the input data set. Given that the method is not intended to be used with a regular grid, the choice of the SLIC regularity parameter $SLICreg \in [0, 1]$ will not be of great importance for practical problems, as long as the $SLICsize$ parameter is chosen small. Exchanging the SLIC segmentation with a regular grid might be of interest for a future real-time application - although at this stage, the method

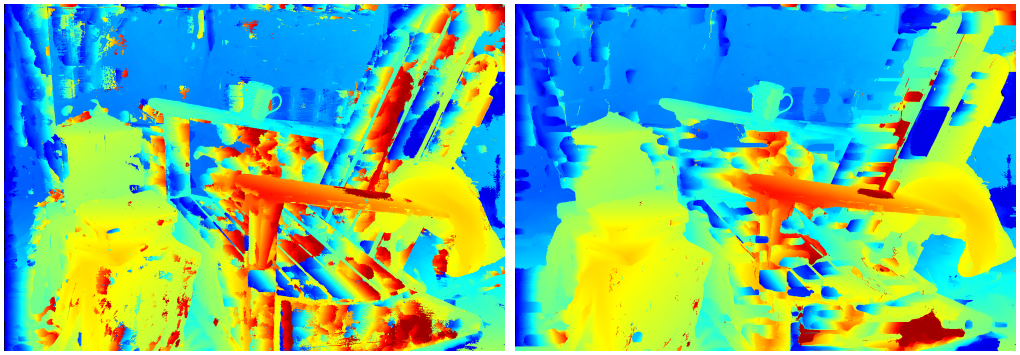


Figure 5.1: *Large regions of wrong matches (bright red and dark blue) can be seen on the back of the chair in the Adirondack disparity map before (left) and after post-processing (right): The regions cannot be corrected by the filling step.*

is nowhere near suitable for that purpose. A $SLICsize$ value of 45, used in the experiment as good 'intermediate' size, would probably count as a rather large value in practice.

5.2.2 Cost Function

The question that shall be answered by the cost function experiments is:

- Does the use of Census and a stronger cost smoothing improve the quality?

To answer them, 5 disparity map results of the correspondence algorithm per parameter setting will be captured and analyzed. The method employed here will be denoted PMF+C. The method for comparison will be a method from the segmentation experiment, with $SLICreg = 1.0$ and $SLICsize = 20$, and denoted PMF. The varied parameter set compared to PMF is:

- A stronger smoothing with the Guided Filter: $\epsilon = 0.05^2$ and a 5×5 Census cost function as described in section 3.1.

The SLIC settings are fixed at $SLICsize = 30$ and $SLICreg = 0.8$ as a practical tradeoff between quality and runtime. As shown earlier in section 5.2.1 the effects of the regularity can be neglected for small superpixel sizes, making any superpixel size between 20 and 45 acceptable. Important parameters are the cost function outlier regulation parameter $\lambda_{Census} = 10$ and the weighting parameter $\beta = 0.8$, i.e. 80 percent of the costs are Census-related.

An exemplary disparity map result is given in figure 5.2. All results (detailed number table and error bar plots) can be found in appendix A.2.

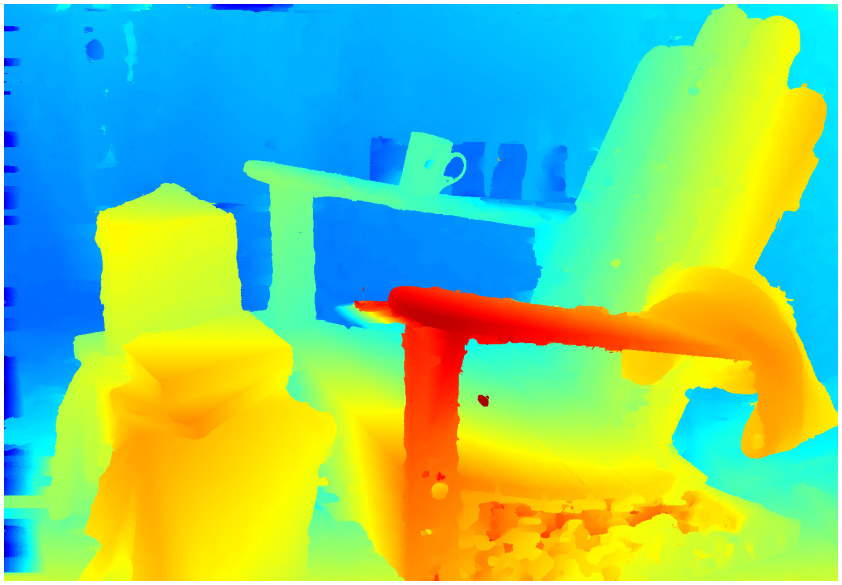


Figure 5.2: *The disparity result of PMF+C for the Adirondack image pair.*

Observations

- An overall low standard deviation (≤ 1.13, including 'nonoccl' error data) shows the reliability of the data.
- All error measurements (including 'nonoccl' error data) are below the corresponding error of PMF. The most impressive improvement can be seen for the Adirondack data set, with the 'all' errors in the 2.0 and 4.0 threshold categories being reduced to one third of the PMF error. Also the data sets Motorcycle and Recycle can achieve errors of about two thirds compared to the PMF error in all but the 0.5 threshold category, for which they reduce the error to 82% (Motorcycle) and 74% (Recycle). For all other image pairs, the relative improvements range from 6.5% to 26.5%.

Conclusions

The PMF+C method is clearly superior to the basic PMF. The positive effect of the cost function substitution and the stronger filtering is all the more delightful because it comes at virtually no costs. Replacing the cost function has little to no repercussions with respect to the structure of the implementation. Additionally, the increase in computational effort caused by the Census window is negligible, at least for small window sizes (5×5 in this case). Other cost functions might even further decrease the overall error.

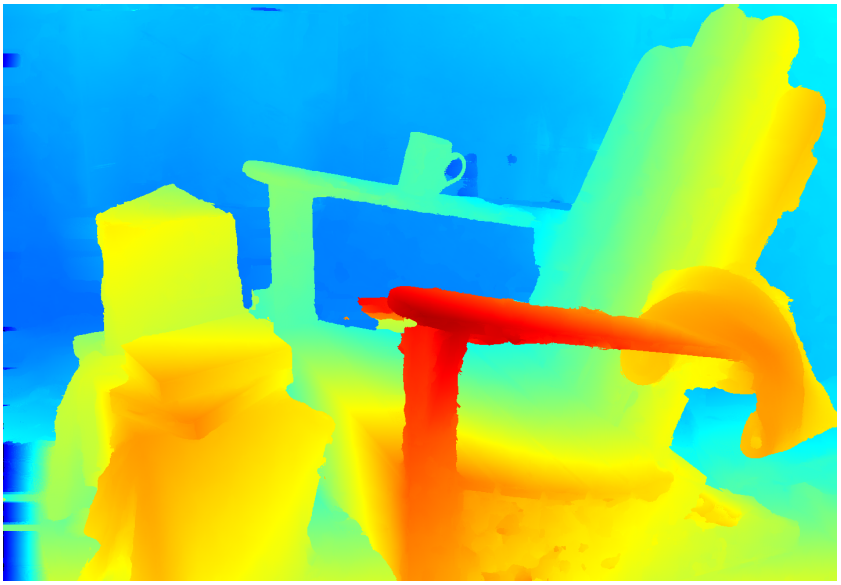


Figure 5.3: *The disparity result of PMF+C+CS for the Adirondack image pair,* $\lambda = 1.175$.

5.2.3 Cross-Scale Regulation

The question that shall be answered by the cross-scale regulation experiments is:

- How does the cross-scale regulation in combination with Census affect the quality?

To answer them, 5 disparity map results of the correspondence algorithm per parameter setting will be captured and analyzed. The method employed here will be denoted PMF+C+CS. For a cross-scale regulation parameter $\lambda = 0$, PMF+C+CS is PMF+C. The corresponding results from the cost function experiment in section 5.2.2 will be reused here. The varied parameters are:

- The cross-scale regulation parameter λ is set in a way that will result in equally spaced corresponding weights for the finest scale w_0.
  $\lambda \in \{0.123, 0.313, 0.62, 1.175, 2.41, 7.4, 500\}$ will give w_0 in steps of 0.1, starting with $w_0 = 0.9$ for $\lambda = 0.123$ up to $w_0 = 0.3$ for $\lambda = 500$.

The segmentation parameters, the cost function parameters, and the Guided Filter settings are the same as for the cost function experiment in section 5.2.2: $SLICsize = 30$ and $SLICreg = 0.8$, $\lambda_{Census} = 10$, $\beta = 0.8$, 5×5 Census window, and $\epsilon = 0.05^2$. The only fixed cross-scale related parameter is the number of used scales $n_s = 5$ as proposed by the authors of [75].

Image Pair	Error Difference Over Varied λ, for Threshold (all) 0.5	1.0	2.0	4.0
Adirondack	4.16	4.65	4.96	4.95
Jadeplant	2.62	2.67	1.78	1.79
Motorcycle	2.06	1.69	1.45	1.18
Recycle	2.37	2.50	1.95	1.97
Shelves	1.58	3.23	5.06	6.81
Teddy	1.07	1.90	2.07	1.77
Vintage	5.49	7.73	7.82	8.12

Table 5.3

An exemplary disparity map result is given in figure 5.3. A complete table with the numerical results as well as corresponding graphical overviews can be found in appendix A.3.

Observations

- The values for the standard deviations are overall small (< 1.00, including 'nonoccl' error data, only for the Jadeplant and Vintage image pairs standard deviations are ≤ 1.39 including 'nonoccl' error data). This is especially important here, because the mean error values for different parameter settings are closer than for the other two experiments (see table 5.3).

- For all image pairs there is a decrease in mean error compared to PMF+C (corresponding to $\lambda = 0$). The image pair with the smallest improvements is Teddy, to the point where the results are almost identical for smaller λ.

- A recurring pattern when examining an increasing λ is a smooth decrease in mean error down to a certain minimum, followed by a smooth increase in mean error.

- Letting the best result for each image pair / error threshold combination vote for its corresponding λ respective w_0, allows to observe further details: The minimum mean error over all error threshold categories is found for a (here: virtual) weight of 0.56. Errors for the stricter thresholds (0.5, 1.0) benefit from a slightly lower cross-scale regulation with $w_0 = 0.62$ compared to $w_0 = 0.51$ for the more tolerant thresholds (2.0, 4.0). Image pairs with large textureless regions (Adirondack, Shelves, Vintage) achieve the best results for a large λ, on average $w_0 = 0.42$. An exception from this is the Recycle image pair ($w_0 = 0.63$). The other image pairs contain mostly regions of rich detail, such as the leaves in Jadeplant, the thin metal structures in

Motorcycle, or the plant in the Teddy foreground. They have an average weight of $w_0 = 0.67$.

Conclusions

The cross-scale regulation enhances PMF+C for most image pairs, at worst the error results are level (as seen in the Teddy image pair). The amount of cross-scale regulation seems to depend on the image content. Images with more ambiguities need the enlarged context of the coarse scales to circumvent the drawbacks of the local approach. However, if fine details have to be preserved, the finest scale needs not to be superseded by coarser scales in the process.

5.3 Summary

The experiments show that the family of PMF methods provide stable, predictable results, even between the most extreme parameter variations. This is an advantage in practice, where users need to find good input parameters without knowing too many implementation details. Furthermore, most parameter concepts, such as superpixel size or cross-scale regulation, can easily form a proper mental image making the effects of changing the corresponding values predictable.

Of all method variants, PMF+C+CS is clearly the best-performing with respect to disparity map error. The current top performers in the Middlebury ranking are global methods. PMF+C+CS is not submitted to the online evaluation, but the hypothetical ranking of all three PMF-based methods using the mean error can be seen in figure 5.4. The base PMF method is the same as denoted under section 5.2.2 ($SLICreg = 1.0$ and $SLICsize = 20$), the parameters for PMF+C are also distinctive. For the ranking of PMF+C+CS, the error measured for a cross-scale regulation of $\lambda = 1.175$ (corresponding to the scale weight $w_0 = 0.6$) is used. The results of the submitted IDR method [43] are included here, because IDR is the subject of another most recent master's thesis [36] supervised by the computer vision in engineering group. Except for one image pair/error threshold combination (Motorcycle, 4.0 error threshold) PMF+C+CS is achieves a higher rank than IDR, although it has to be noted that IDR is a real-time stereo correspondence algorithm while PMF+C+CS does not need to balance runtime performance and disparity map quality.

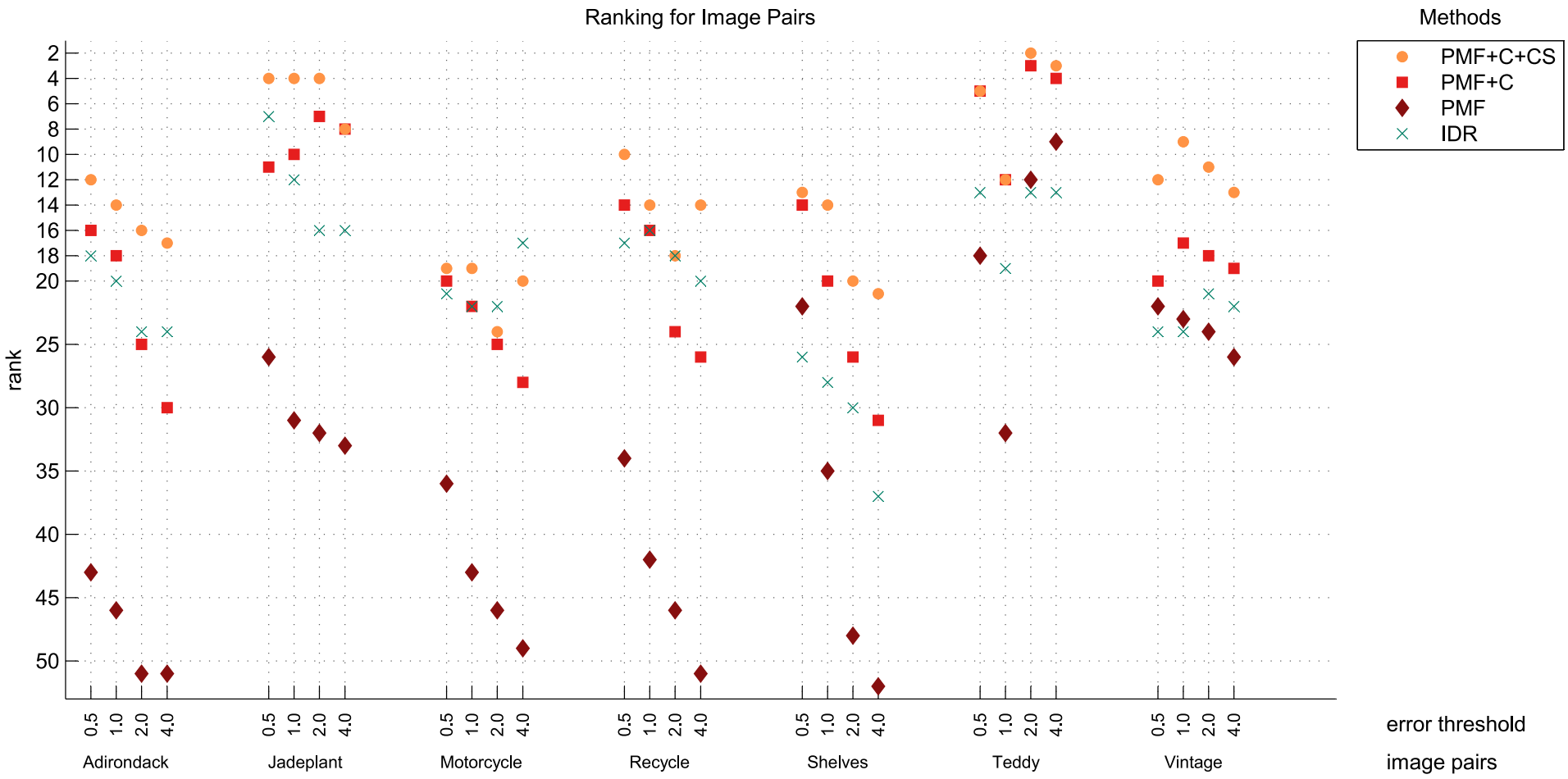


Figure 5.4: *This ranking is determined based on the public online ranking on the Middelbury website from November 18th, 2016. As mentioned before, the online evaluation is always carried out using the full resolution ground truth, while here the half resolution ground truth is used.*

Chapter 6

Conclusion and Outlook

The first section of this final chapter will sum up the most important insights and the main contributions of this work. There are plenty of open questions, half-formed ideas and other loose ends that deserve further investigation. These are layed out in the second section.

6.1 Contributions

This work merges the ideas of two publications: the superpixel-based PatchMatch Filter framework [45] and the cross-scale regulated cost aggregation [75]. Additionally, by replacing a component of the cost function the proposed method is further enhanced. On the whole, the framework-character of the PatchMatch Filter method is preserved, maintaining extensibility for future changes. The effects of the single, stable components are analyzed, if possible, over the whole range of their defining parameters.

Heap-PatchMatch is build in as a minor improvement, but has not been tested for effects on the performance of the method. Considering the possibilities of exploiting self-similarity - despite the fact that no stable progress is included in the final version of the proposed method - several possible directions of further exploration exist.

The ranking estimations show that local methods still have a chance at competing with the qualitative results of global approaches: for two of the test image pairs the proposed method achieves a rank of 5 or better for individual error thresholds.

6.2 Future Work

There are numerous questions that should be answered by supplementary research.

6.2.1 Cross-Scale-Related

The experiments suggested that the optimal value for cross-scale regulation depends on the structure of the input images. The cross-scale regulation does not have to be fixed for the whole image, it could be adaptively changed depending on the actual need for cross-scale regulation in a certain image region. This is somewhat related to self-similarity: the more ambiguous a region is, the more it could benefit from cross-scale information. This could help to retain sharp detail while stabilizing large uniform regions. To take the cross-scale regulation even further: the weights do not have to be controlled by one parameter, but can be set independently. For example, each region could have only one dedicated scale for cost stabilization - the cross-scale cost aggregation could be executed selectively instead of adaptively. Finding a good number of scales or testing with different scaling methods, for example with better detail preservation, are also tasks that promise a higher disparity map quality while being very non-invasive to the current method.

6.2.2 PatchMatch-Related

A pending experiment would be examining the effect of Heap-PatchMatch. A second PatchMatch extension made in [11], that is yet to be implemented in the proposed method, is 'forward enrichment': Label hypotheses for a (super)pixel $S(k)$ can also be derived from the current best label of the pixels that the current best label of $S(k)$ points to ('the nearest neighbors of the nearest neighbors').

Heap-PatchMatch could also be of use in the post-processing step, when pixels are declared invalid because of their best labels. The correct, valid label could be in the list of best labels per pixel. Also, the Heap-PatchMatch lists could be used to establish confidences for single labels, for example based on cost differences between them.

The PatchMatch parameter for the search scope step is set once, although the search could be accelerated with an adaptive search scope, which is for example depending on the respective cost of the start label. When the costs are low, the focus should be on the search close to the label (which one could say would be the current, untested setting). When costs are high, the bias of PatchMatch is shifted towards more distant labels, or at least is less center-focussed. Maybe there are other, even better ways of steering PatchMatch than by using matching costs.

6.2.3 Self-Similarity-Related

This topic is probably the most open one. Some of the most interesting questions are: How can the quality of the pattern detection be improved? Which additional

information should be included in the pattern description (such as the already mentioned distances or confidences)? What is the best way to use patterns as matching constraints (e.g. the costs of assigning a pattern position to the respective image patch should also be considered)? Can the patterns be uncovered, described, and applied symmetrically, for left and right image?

Most importantly, the benefit needs to be large to justify the greater computational and memory load. Even if all three tasks of a pattern matching approach (detection, description, and matching) are done by most sophisticated and smart algorithms, the effect on the overall quality might be not significant at all because of the fragile ordering constraint and occlusions preventing recovery of pattern elements.

6.2.4 General Questions

It would be interesting to adapt the proposed method for optical flow computation - which is done in the original PatchMatch Filter publication [45]. Including vertical disparities in the search can avert errors caused by an imperfect rectification.

An improvement that would be necessary for practical application is the refinement of the current algorithm implementation. Becauso of the explorative nature of this work, the runtime performance is not a priority, there is for example no kind of concurrency.

Furthermore, the interpolations due to the arbitrary 3D-position of the support window is costly. To accelerate the disparity search it might be of advantage to 'downgrade' the support windows to fronto-parallel and the disparities to integers in the first few iterations before allowing arbitrary labels again in later iterations.

One of the generally accepted constraints for local approaches is their inability to handle explicit smoothness. Using explicit smoothness in a global context is certainly not possible, since that is what sets the two approaches apart. However, explicit smoothness constraints in a locally defined context might be applicable to local approaches.

Appendices

Appendix A

Experiment Results

A.1 Segmentation

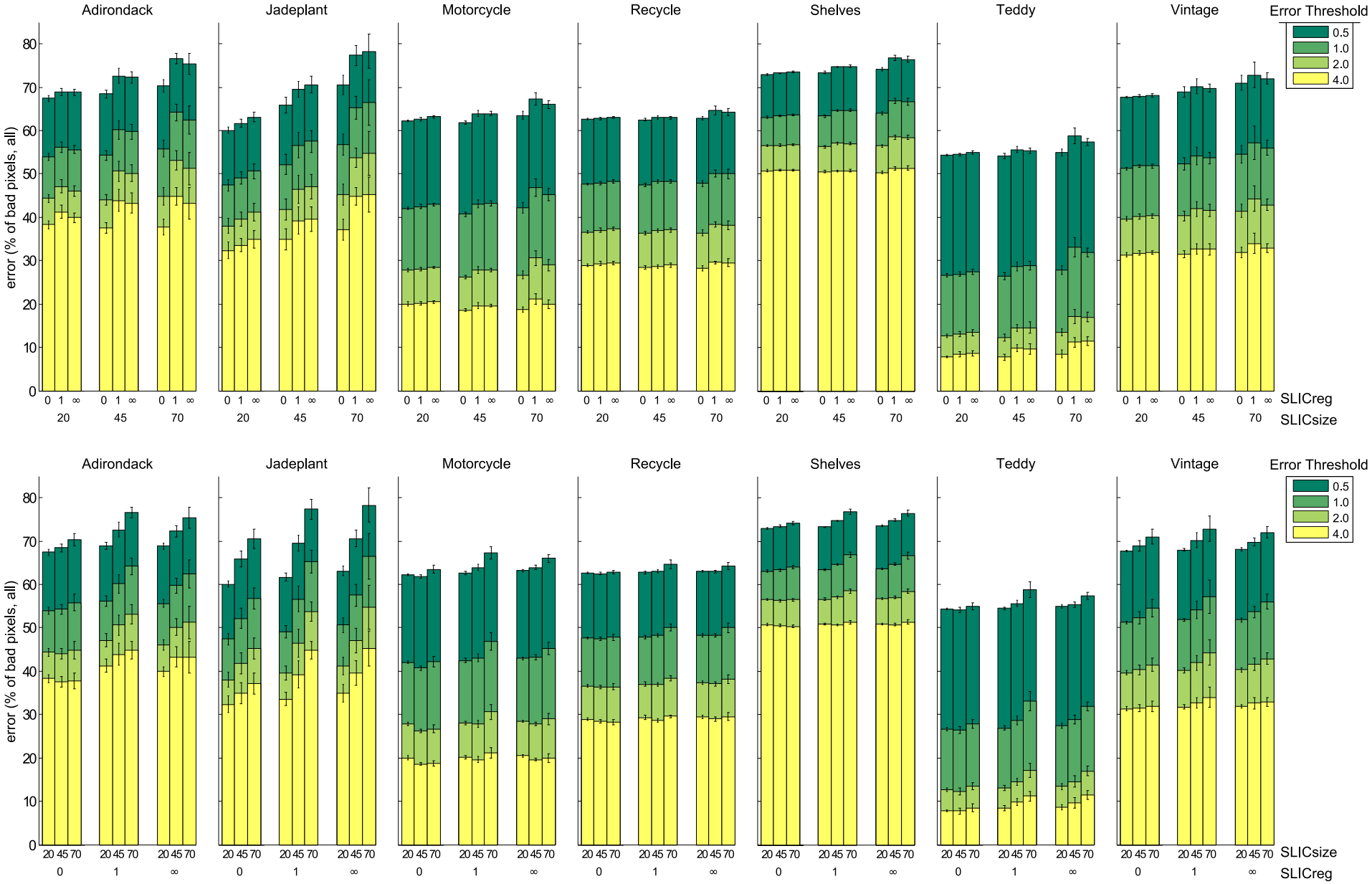
A.1.1 Errors (% of Bad Pixels) of PMF (Table)

Image Pair	SLIC size	SLIC reg	all ± std								nonoccl ± std							
			0.5		1.0		2.0		4.0		0.5		1.0		2.0		4.0	
Adirondack	20	0.0	67.55	±0.72	54.10	±0.87	44.49	±0.90	38.40	±0.88	67.81	±0.72	54.30	±0.88	44.66	±0.91	38.55	±0.89
	20	1.0	68.90	±0.75	56.09	±1.12	47.00	±1.48	41.06	±1.56	69.16	±0.75	56.31	±1.13	47.18	±1.49	41.22	±1.57
	20	∞	68.99	±0.90	55.81	±1.11	46.28	±1.34	40.13	±1.22	69.26	±0.90	56.02	±1.12	46.46	±1.34	40.28	±1.23
	45	0.0	68.29	±1.01	54.09	±1.24	43.86	±1.35	37.36	±1.31	68.55	±1.02	54.30	±1.24	44.03	±1.36	37.50	±1.32
	45	1.0	72.77	±1.66	60.22	±1.96	50.71	±2.24	44.00	±2.43	73.05	±1.67	60.45	±1.97	50.91	±2.25	44.17	±2.44
	45	∞	72.02	±1.52	59.37	±1.96	49.66	±2.28	42.88	±2.50	72.29	±1.53	59.60	±1.97	49.85	±2.29	43.04	±2.51
	70	0.0	70.28	±1.36	55.71	±1.94	44.64	±1.99	37.53	±1.80	70.55	±1.37	55.93	±1.94	44.81	±2.00	37.68	±1.81
	70	1.0	76.68	±1.20	64.22	±1.81	53.08	±2.07	44.77	±1.88	76.97	±1.21	64.47	±1.82	53.28	±2.08	44.94	±1.88
	70	∞	75.26	±2.39	62.17	±3.10	51.05	±3.55	43.04	±3.55	75.55	±2.40	62.41	±3.11	51.25	±3.56	43.20	±3.56
Jadeplant	20	0.0	60.15	±0.66	47.52	±1.39	38.24	±1.77	32.57	±1.83	61.11	±0.67	48.28	±1.41	38.84	±1.80	33.09	±1.86
	20	1.0	61.74	±0.92	49.08	±1.35	39.56	±1.54	33.55	±1.46	62.72	±0.93	49.87	±1.37	40.19	±1.56	34.09	±1.48
	20	∞	63.00	±1.11	50.37	±1.79	40.86	±2.07	34.71	±2.07	64.01	±1.13	51.17	±1.81	41.51	±2.11	35.26	±2.10
	45	0.0	65.61	±1.90	51.91	±2.32	41.46	±2.53	34.59	±2.50	66.66	±1.93	52.74	±2.36	42.12	±2.57	35.14	±2.54
	45	1.0	69.60	±1.71	56.56	±2.77	46.31	±3.12	39.02	±3.07	70.71	±1.74	57.46	±2.81	47.05	±3.17	39.64	±3.11
	45	∞	70.60	±1.78	57.40	±2.44	46.82	±2.81	39.43	±2.73	71.72	±1.81	58.32	±2.48	47.56	±2.85	40.06	±2.77
	70	0.0	70.98	±2.41	57.28	±2.83	45.81	±2.82	37.71	±2.82	72.11	±2.45	58.19	±2.87	46.53	±2.87	38.31	±2.87
	70	1.0	77.11	±2.34	64.83	±2.75	53.27	±2.56	44.36	±2.48	78.34	±2.38	65.87	±2.79	54.12	±2.60	45.07	±2.52
	70	∞	77.99	±3.86	66.01	±5.10	54.36	±4.92	45.00	±3.80	79.23	±3.92	67.06	±5.18	55.22	±5.00	45.72	±3.86
Motorcycle	20	0.0	62.21	±0.29	42.01	±0.36	27.74	±0.41	19.99	±0.42	63.59	±0.29	42.94	±0.37	28.35	±0.42	20.43	±0.43
	20	1.0	62.63	±0.35	42.46	±0.44	28.02	±0.40	20.10	±0.37	64.01	±0.35	43.40	±0.45	28.64	±0.41	20.54	±0.38
	20	∞	63.13	±0.28	42.94	±0.30	28.35	±0.24	20.40	±0.25	64.53	±0.29	43.90	±0.31	28.98	±0.25	20.85	±0.26
	45	0.0	61.95	±0.49	40.74	±0.56	26.23	±0.40	18.58	±0.22	63.32	±0.50	41.65	±0.57	26.80	±0.41	18.99	±0.23
	45	1.0	64.02	±0.75	43.20	±0.96	27.93	±0.74	19.62	±0.61	65.44	±0.77	44.15	±0.99	28.55	±0.76	20.05	±0.62
	45	∞	64.02	±0.57	43.21	±0.59	27.87	±0.44	19.57	±0.32	65.44	±0.59	44.17	±0.60	28.49	±0.44	20.00	±0.32
	70	0.0	63.45	±0.96	42.05	±1.17	26.62	±0.87	18.64	±0.67	64.86	±0.98	42.98	±1.20	27.21	±0.89	19.05	±0.68
	70	1.0	67.49	±1.56	47.12	±2.11	30.81	±1.84	21.32	±1.22	68.98	±1.59	48.16	±2.16	31.49	±1.88	21.79	±1.25
	70	∞	65.88	±0.90	45.06	±1.42	28.90	±1.32	19.85	±1.05	67.34	±0.92	46.05	±1.46	29.54	±1.35	20.29	±1.08

Recycle	20	0.0	62.62	±0.14	47.71	±0.23	36.62	±0.30	28.93	±0.26	63.70	±0.15	48.52	±0.23	37.25	±0.30	29.43	±0.27
	20	1.0	62.78	±0.31	47.88	±0.43	37.00	±0.50	29.31	±0.51	63.85	±0.31	48.70	±0.43	37.63	±0.51	29.81	±0.52
	20	∞	63.04	±0.24	48.29	±0.32	37.31	±0.41	29.48	±0.40	64.12	±0.25	49.12	±0.33	37.95	±0.42	29.98	±0.41
	45	0.0	62.58	±0.35	47.55	±0.40	36.34	±0.39	28.45	±0.31	63.65	±0.36	48.36	±0.41	36.96	±0.39	28.93	±0.32
	45	1.0	63.03	±0.35	48.18	±0.48	36.78	±0.46	28.56	±0.42	64.11	±0.36	49.00	±0.49	37.41	±0.46	29.05	±0.43
	45	∞	62.99	±0.26	48.27	±0.44	37.07	±0.48	28.96	±0.43	64.07	±0.27	49.10	±0.44	37.70	±0.48	29.46	±0.44
	70	0.0	62.78	±0.45	47.76	±0.63	36.35	±0.64	28.18	±0.58	63.86	±0.46	48.58	±0.65	36.97	±0.65	28.67	±0.59
	70	1.0	64.62	±0.90	50.09	±0.83	38.32	±0.55	29.50	±0.43	65.72	±0.91	50.95	±0.84	38.98	±0.56	30.01	±0.43
	70	∞	64.23	±0.78	50.05	±1.01	38.24	±0.94	29.51	±0.87	65.33	±0.79	50.91	±1.03	38.89	±0.96	30.02	±0.88
Shelves	20	0.0	73.05	±0.18	63.15	±0.21	56.52	±0.23	50.77	±0.27	81.36	±0.20	70.34	±0.24	62.95	±0.26	56.55	±0.30
	20	1.0	73.32	±0.13	63.45	±0.18	56.62	±0.27	50.83	±0.27	81.66	±0.15	70.66	±0.20	63.06	±0.30	56.61	±0.30
	20	∞	73.55	±0.16	63.66	±0.19	56.81	±0.19	50.94	±0.18	81.91	±0.18	70.90	±0.21	63.27	±0.21	56.73	±0.21
	45	0.0	73.44	±0.28	63.37	±0.30	56.31	±0.29	50.56	±0.26	81.79	±0.31	70.57	±0.33	62.71	±0.32	56.30	±0.29
	45	1.0	74.76	±0.10	64.70	±0.17	57.10	±0.24	50.73	±0.27	83.26	±0.12	72.05	±0.19	63.60	±0.26	56.50	±0.30
	45	∞	74.83	±0.31	64.78	±0.38	57.11	±0.33	50.82	±0.33	83.34	±0.35	72.14	±0.42	63.61	±0.36	56.60	±0.37
	70	0.0	74.12	±0.38	63.99	±0.40	56.50	±0.26	50.35	±0.28	82.55	±0.43	71.27	±0.45	62.93	±0.29	56.08	±0.31
	70	1.0	76.69	±0.59	66.87	±0.63	58.49	±0.52	51.31	±0.38	85.41	±0.65	74.48	±0.71	65.14	±0.58	57.14	±0.42
	70	∞	76.45	±0.68	66.71	±0.77	58.49	±0.55	51.40	±0.46	85.15	±0.76	74.30	±0.86	65.14	±0.61	57.25	±0.51
Teddy	20	0.0	54.35	±0.22	26.60	±0.35	12.64	±0.31	7.85	±0.30	55.64	±0.23	27.23	±0.35	12.94	±0.32	8.03	±0.31
	20	1.0	54.44	±0.25	26.85	±0.48	13.07	±0.65	8.46	±0.65	55.74	±0.26	27.49	±0.49	13.39	±0.66	8.66	±0.67
	20	∞	54.85	±0.39	27.28	±0.65	13.34	±0.72	8.61	±0.55	56.16	±0.40	27.93	±0.66	13.66	±0.73	8.82	±0.56
	45	0.0	54.10	±0.52	26.40	±0.82	12.32	±0.78	7.75	±0.63	55.39	±0.53	27.02	±0.84	12.61	±0.79	7.94	±0.65
	45	1.0	55.55	±0.67	28.69	±0.91	14.60	±0.81	9.87	±0.83	56.88	±0.69	29.37	±0.93	14.94	±0.82	10.11	±0.85
	45	∞	55.40	±0.61	28.81	±1.00	14.55	±1.23	9.60	±1.17	56.72	±0.62	29.50	±1.02	14.90	±1.26	9.83	±1.20
	70	0.0	54.98	±0.77	27.86	±0.95	13.50	±0.76	8.62	±0.90	56.28	±0.80	28.52	±0.97	13.82	±0.78	8.82	±0.93
	70	1.0	58.62	±1.83	32.97	±2.18	17.24	±1.53	11.24	±1.08	60.01	±1.87	33.76	±2.23	17.65	±1.57	11.51	±1.11
	70	∞	57.49	±0.97	32.12	±1.20	17.09	±1.19	11.52	±1.08	58.86	±0.99	32.89	±1.23	17.50	±1.21	11.79	±1.10

Vintage	20	0.0	67.69	±0.21	51.25	±0.25	39.69	±0.44	31.36	±0.45	69.39	±0.22	52.54	±0.25	40.69	±0.45	32.15	±0.46
	20	1.0	68.03	±0.23	51.85	±0.35	40.20	±0.55	31.74	±0.51	69.74	±0.24	53.15	±0.36	41.21	±0.56	32.54	±0.53
	20	∞	68.12	±0.34	51.85	±0.45	40.30	±0.48	31.82	±0.43	69.83	±0.34	53.16	±0.47	41.31	±0.49	32.62	±0.44
	45	0.0	69.11	±1.23	52.58	±1.49	40.58	±1.29	31.65	±1.00	70.85	±1.26	53.91	±1.53	41.60	±1.32	32.44	±1.03
	45	1.0	70.21	±1.58	54.16	±1.91	41.97	±1.57	32.60	±1.12	71.98	±1.61	55.52	±1.95	43.02	±1.61	33.42	±1.15
	45	∞	69.72	±0.93	53.56	±1.30	41.53	±1.41	32.49	±1.19	71.47	±0.96	54.91	±1.33	42.57	±1.45	33.31	±1.23
	70	0.0	71.06	±1.65	54.66	±1.95	41.57	±1.49	32.00	±1.12	72.85	±1.69	56.03	±2.00	42.61	±1.53	32.80	±1.14
	70	1.0	73.48	±3.49	58.23	±4.74	45.22	±4.02	34.67	±3.04	75.32	±3.57	59.69	±4.86	46.35	±4.12	35.53	±3.12
	70	∞	71.85	±1.28	55.78	±1.74	42.70	±1.35	32.74	±0.97	73.65	±1.31	57.18	±1.79	43.77	±1.39	33.57	±1.00

A.1.2 Errors (% of Bad Pixels) of PMF (Plots)

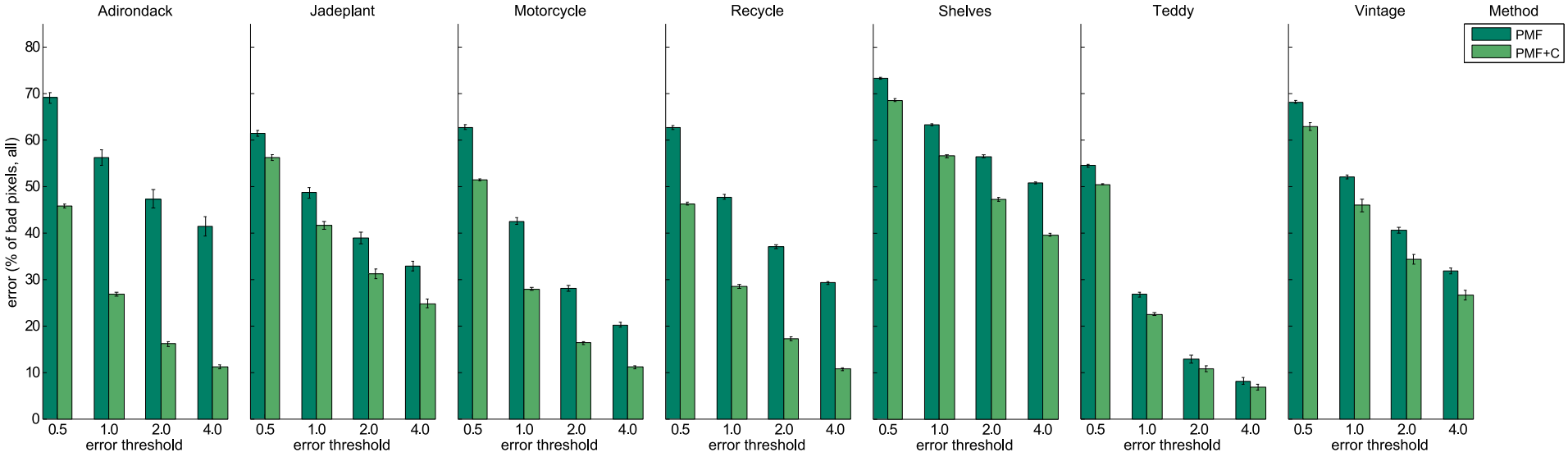


A.2 Cost Function

A.2.1 Errors (% of Bad Pixels) of PMF+C (Table)

Image	all ± std								nonoccl ± std							
Pair	0.5		1.0		2.0		4.0		0.5		1.0		2.0		4.0	
Adirondack	45.77	±0.32	26.77	±0.44	16.09	±0.47	11.19	±0.38	45.95	±0.32	26.87	±0.44	16.15	±0.47	11.23	±0.38
Error wrt. PMF(%)	66.43		47.73		34.23		27.26		66.44		47.72		34.23		27.25	
Jadeplant	55.99	±0.63	41.39	±0.91	30.94	±0.98	24.68	±0.91	56.89	±0.63	42.05	±0.92	31.43	±1.00	25.07	±0.92
Error wrt. PMF(%)	90.69		84.32		78.20		73.54		90.70		84.32		78.20		73.55	
Motorcycle	51.31	±0.18	27.90	±0.27	16.35	±0.34	11.16	±0.31	52.44	±0.19	28.51	±0.28	16.72	±0.34	11.40	±0.32
Error wrt. PMF(%)	81.92		65.70		58.37		55.52		81.93		65.69		58.38		55.51	
Recycle	46.19	±0.30	28.41	±0.37	17.25	±0.41	10.78	±0.32	46.98	±0.30	28.90	±0.37	17.54	±0.41	10.96	±0.33
Error wrt. PMF(%)	73.58		59.35		46.62		36.76		73.58		59.35		46.62		36.76	
Shelves	68.54	±0.26	56.51	±0.29	47.13	±0.46	39.52	±0.29	76.33	±0.29	62.94	±0.33	52.49	±0.52	44.02	±0.33
Error wrt. PMF(%)	93.47		89.06		83.24		77.75		93.47		89.07		83.24		77.75	
Teddy	50.34	±0.09	22.56	±0.32	10.81	±0.59	6.84	±0.59	51.54	±0.09	23.10	±0.33	11.07	±0.60	7.01	±0.61
Error wrt. PMF(%)	92.47		84.03		82.69		80.86		92.47		84.03		82.68		80.92	
Vintage	62.70	±0.76	45.81	±1.10	34.32	±0.96	26.78	±1.05	64.27	±0.78	46.96	±1.13	35.18	±0.99	27.45	±1.08
Error wrt. PMF(%)	92.16		88.36		85.38		84.36		92.16		88.36		85.39		84.36	

A.2.2 Errors (% of Bad Pixels) of PMF+C Compared to PMF (Plots)



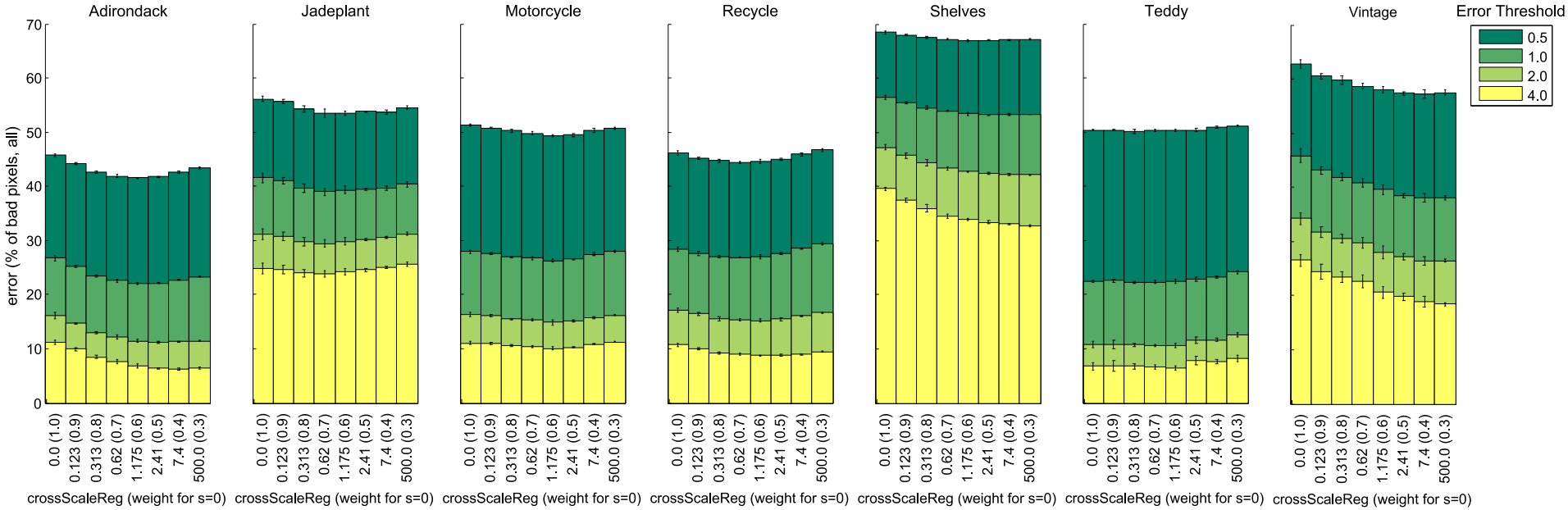
A.3 Cross-Scale Regulation

A.3.1 Errors (% of Bad Pixels) of PMF+C+CS (Table)

Image Pair	CS reg	CS w_0	all ± std								nonoccl ± std							
			0.5		1.0		2.0		4.0		0.5		1.0		2.0		4.0	
Adirondack	0	1	45.73	±0.35	26.77	±0.51	16.09	±0.54	11.18	±0.43	45.90	±0.35	26.86	±0.51	16.16	±0.54	11.22	±0.43
	0.123	0.9	44.20	±0.26	25.14	±0.22	14.65	±0.17	9.89	±0.27	44.37	±0.26	25.24	±0.22	14.71	±0.18	9.93	±0.27
	0.313	0.8	42.63	±0.13	23.38	±0.23	12.96	±0.26	8.48	±0.25	42.80	±0.14	23.47	±0.23	13.01	±0.26	8.51	±0.25
	0.62	0.7	41.92	±0.28	22.63	±0.30	12.11	±0.37	7.58	±0.34	42.08	±0.28	22.72	±0.31	12.16	±0.37	7.61	±0.34
	1.175	0.6	41.57	±0.13	22.12	±0.20	11.45	±0.25	6.81	±0.35	41.72	±0.14	22.20	±0.20	11.49	±0.25	6.83	±0.35
	2.41	0.5	41.80	±0.12	22.14	±0.12	11.14	±0.17	6.35	±0.15	41.96	±0.12	22.22	±0.11	11.19	±0.17	6.37	±0.15
	7.4	0.4	42.66	±0.24	22.70	±0.12	11.29	±0.08	6.23	±0.15	42.82	±0.24	22.78	±0.13	11.33	±0.08	6.26	±0.15
	500	0.3	43.37	±0.17	23.30	±0.11	11.43	±0.08	6.34	±0.21	43.54	±0.17	23.39	±0.11	11.48	±0.08	6.37	±0.21
Jadeplant	0	1	56.16	±0.59	41.60	±0.89	31.12	±1.04	24.79	±1.00	57.06	±0.60	42.27	±0.91	31.61	±1.05	25.19	±1.02
	0.123	0.9	55.69	±0.40	41.07	±0.61	30.82	±0.78	24.67	±0.77	56.58	±0.41	41.72	±0.62	31.31	±0.79	25.06	±0.78
	0.313	0.8	54.31	±0.61	39.70	±0.82	29.72	±0.76	23.95	±0.63	55.17	±0.62	40.33	±0.83	30.19	±0.78	24.34	±0.65
	0.62	0.7	53.53	±0.76	38.99	±0.70	29.45	±0.67	23.86	±0.56	54.39	±0.77	39.61	±0.72	29.92	±0.68	24.25	±0.57
	1.175	0.6	53.56	±0.40	39.32	±0.67	29.79	±0.67	24.19	±0.57	54.42	±0.41	39.95	±0.68	30.26	±0.68	24.57	±0.58
	2.41	0.5	53.81	±0.17	39.38	±0.22	30.08	±0.19	24.60	±0.29	54.66	±0.17	40.01	±0.22	30.57	±0.20	24.99	±0.30
	7.4	0.4	53.74	±0.36	39.63	±0.33	30.58	±0.24	25.00	±0.21	54.60	±0.37	40.25	±0.34	31.07	±0.24	25.39	±0.22
	500	0.3	54.55	±0.32	40.34	±0.45	31.23	±0.34	25.66	±0.41	55.42	±0.32	40.98	±0.45	31.73	±0.34	26.07	±0.41
Motorcycle	0	1	51.34	±0.20	27.91	±0.31	16.29	±0.35	11.07	±0.26	52.47	±0.20	28.52	±0.32	16.65	±0.36	11.31	±0.27
	0.123	0.9	50.79	±0.17	27.56	±0.23	16.11	±0.14	11.01	±0.16	51.92	±0.17	28.17	±0.24	16.47	±0.14	11.25	±0.17
	0.313	0.8	50.27	±0.29	26.92	±0.16	15.45	±0.17	10.57	±0.16	51.38	±0.29	27.52	±0.16	15.79	±0.17	10.80	±0.16
	0.62	0.7	49.81	±0.27	26.75	±0.27	15.36	±0.24	10.43	±0.18	50.91	±0.27	27.34	±0.27	15.70	±0.24	10.67	±0.18
	1.175	0.6	49.27	±0.18	26.27	±0.36	14.84	±0.41	10.01	±0.32	50.37	±0.18	26.85	±0.37	15.17	±0.42	10.24	±0.33
	2.41	0.5	49.46	±0.29	26.59	±0.11	15.13	±0.14	10.26	±0.14	50.56	±0.29	27.17	±0.11	15.47	±0.15	10.48	±0.15
	7.4	0.4	50.28	±0.34	27.44	±0.28	15.80	±0.18	10.84	±0.11	51.39	±0.35	28.04	±0.29	16.15	±0.18	11.08	±0.11
	500	0.3	50.63	±0.20	27.96	±0.15	16.19	±0.11	11.19	±0.09	51.75	±0.20	28.57	±0.16	16.55	±0.11	11.44	±0.09

Recycle	0	1	46.18	±0.35	28.38	±0.42	17.17	±0.42	10.70	±0.32	46.97	±0.35	28.86	±0.42	17.46	±0.43	10.88	±0.32
	0.123	0.9	45.20	±0.24	27.56	±0.32	16.42	±0.27	10.01	±0.27	45.97	±0.24	28.03	±0.33	16.70	±0.28	10.18	±0.28
	0.313	0.8	44.73	±0.25	27.04	±0.15	15.49	±0.34	9.20	±0.25	45.50	±0.25	27.50	±0.15	15.75	±0.35	9.36	±0.26
	0.62	0.7	44.33	±0.18	26.86	±0.12	15.34	±0.20	8.95	±0.17	45.09	±0.18	27.32	±0.13	15.60	±0.21	9.10	±0.18
	1.175	0.6	44.60	±0.45	27.01	±0.33	15.22	±0.28	8.74	±0.16	45.36	±0.45	27.47	±0.34	15.48	±0.29	8.89	±0.16
	2.41	0.5	44.94	±0.15	27.53	±0.24	15.46	±0.25	8.73	±0.22	45.71	±0.16	28.00	±0.24	15.73	±0.26	8.88	±0.22
	7.4	0.4	45.89	±0.24	28.48	±0.17	16.07	±0.09	8.97	±0.12	46.68	±0.25	28.97	±0.17	16.35	±0.09	9.12	±0.12
	500	0.3	46.71	±0.31	29.37	±0.26	16.62	±0.10	9.41	±0.09	47.50	±0.31	29.87	±0.27	16.91	±0.10	9.57	±0.09
Shelves	0	1	68.51	±0.29	56.46	±0.32	47.20	±0.51	39.56	±0.33	76.29	±0.32	62.88	±0.35	52.56	±0.56	44.06	±0.36
	0.123	0.9	67.98	±0.11	55.42	±0.20	45.70	±0.44	37.53	±0.39	75.72	±0.12	61.72	±0.22	50.91	±0.49	41.80	±0.42
	0.313	0.8	67.53	±0.17	54.55	±0.33	44.44	±0.56	35.97	±0.67	75.21	±0.19	60.75	±0.37	49.49	±0.63	40.07	±0.75
	0.62	0.7	67.23	±0.06	53.95	±0.12	43.33	±0.34	34.55	±0.45	74.88	±0.07	60.09	±0.13	48.26	±0.38	38.48	±0.50
	1.175	0.6	66.93	±0.14	53.46	±0.30	42.77	±0.07	33.92	±0.13	74.54	±0.15	59.54	±0.34	47.63	±0.08	37.78	±0.15
	2.41	0.5	66.97	±0.12	53.23	±0.14	42.46	±0.23	33.42	±0.24	74.58	±0.13	59.28	±0.16	47.28	±0.25	37.22	±0.27
	7.4	0.4	67.04	±0.19	53.36	±0.18	42.21	±0.18	33.05	±0.17	74.66	±0.21	59.43	±0.20	47.02	±0.20	36.80	±0.19
	500	0.3	67.20	±0.04	53.31	±0.07	42.14	±0.16	32.74	±0.17	74.84	±0.05	59.38	±0.08	46.93	±0.18	36.47	±0.19
Teddy	0	1	50.34	±0.10	22.46	±0.27	10.68	±0.59	6.73	±0.62	51.53	±0.10	23.00	±0.27	10.94	±0.60	6.90	±0.64
	0.123	0.9	50.37	±0.10	22.55	±0.30	10.75	±0.85	6.77	±0.95	51.57	±0.11	23.08	±0.31	11.01	±0.86	6.93	±0.97
	0.313	0.8	50.14	±0.37	22.24	±0.26	10.70	±0.24	6.73	±0.56	51.33	±0.38	22.77	±0.27	10.95	±0.24	6.89	±0.57
	0.62	0.7	50.27	±0.20	22.34	±0.22	10.61	±0.24	6.69	±0.40	51.47	±0.20	22.88	±0.22	10.86	±0.24	6.85	±0.42
	1.175	0.6	50.35	±0.14	22.44	±0.33	10.57	±0.44	6.46	±0.42	51.55	±0.14	22.98	±0.34	10.82	±0.45	6.61	±0.43
	2.41	0.5	50.40	±0.26	22.85	±0.35	11.64	±0.61	7.78	±0.82	51.59	±0.27	23.39	±0.36	11.91	±0.62	7.97	±0.84
	7.4	0.4	50.83	±0.19	23.22	±0.19	11.63	±0.29	7.65	±0.37	52.05	±0.20	23.78	±0.20	11.91	±0.30	7.83	±0.38
	500	0.3	51.21	±0.17	24.14	±0.26	12.63	±0.38	8.23	±0.53	52.43	±0.18	24.72	±0.27	12.94	±0.39	8.43	±0.55
Vintage	0	1	62.77	±0.86	45.84	±1.27	34.24	±1.09	26.54	±1.05	64.35	±0.88	46.99	±1.30	35.10	±1.12	27.21	±1.08
	0.123	0.9	60.55	±0.54	43.14	±0.72	31.76	±1.00	24.48	±1.36	62.07	±0.55	44.23	±0.73	32.56	±1.02	25.09	±1.39
	0.313	0.8	59.77	±0.76	41.92	±0.74	30.53	±0.73	23.36	±0.96	61.27	±0.79	42.97	±0.76	31.30	±0.74	23.94	±0.99
	0.62	0.7	58.70	±0.53	40.75	±0.79	29.74	±1.00	22.65	±1.11	60.17	±0.54	41.77	±0.81	30.49	±1.02	23.21	±1.14
	1.175	0.6	58.09	±0.63	39.56	±0.87	27.98	±1.09	20.63	±1.07	59.55	±0.65	40.55	±0.89	28.69	±1.12	21.14	±1.09
	2.41	0.5	57.35	±0.23	38.53	±0.37	27.11	±0.58	19.79	±0.72	58.79	±0.24	39.50	±0.38	27.79	±0.59	20.29	±0.73
	7.4	0.4	57.28	±0.74	38.11	±0.82	26.42	±0.86	18.87	±0.91	58.72	±0.76	39.07	±0.84	27.09	±0.88	19.34	±0.94
	500	0.3	57.48	±0.47	38.12	±0.44	26.45	±0.27	18.42	±0.33	58.92	±0.48	39.08	±0.45	27.11	±0.28	18.88	±0.34

A.3.2 Errors (% of Bad Pixels) of PMF+C+CS (Plots)



Appendix B

Implementation Details

B.1 Parameters

Parameter Name	Comment	Default Value
`dispScale`	The scale factor for the error with respect to the ground truth, depending on the used resolution.	2.0
`costScale`	Scale factor for displaying the cost image.	255.0
`useColorGuide`	Flag for using a color image as guide for the Guided Filter (alternative: gray-value image).	true
`useBestPixel`	Keep track of the pixel with the lowest associated cost per superpixel and use its label whenever a random label should be picked in the PatchMatch algorithm.	false
`numRandomPicks`	Number of randomly drawn pixels per superpixel from which the label with the lowest cost is chosen whenever a random label should be picked in the PatchMatch algorithm.	1
`interleavedMagic`	Flag for alternating between left and right image when executing the 3 label testing and propagation steps (neighborhood,random,view) (alternative: do all 3 for left image first, then all 3 for right image).	false
`maxDisp`	Upper limit for the disparity search range. In case of using the Middlebury data set, a value is provided for each image pair.	-
`minDisp`	Lower limit for the disparity search range.	0.0
`maxcost`	Maximum possible cost, used for cost normalization.	[Depends on cost function weights and thresholds.]
`alpha`	Factor for distance reduction in the PatchMatch random search.	0.5
`radius`	The width of the 'border' around each $B(k)$, determining $R(k)$.	9

`crossCheckLimit`	The threshold for the left-right cross-check.	1.0
`useCensus`	Flag for using the Census transform in the cost function (alternative: gradient).	true
`lambdaCensus`	For controlling the Census transform.	10.0
`withCrossScaleAggregaton`	Flag for using cross-scale regulation of costs.	false
`numScales`	Number of used scales if cross-scale regulated.	5
`crossScaleReg`	Strength of cross-scale regulation of costs.	5.0
`enrichmentNumberAppearanceNeighbors`	Maximum number of added appearance neighbors per superpixel.	3
`enrichmentRadiusAppearanceNeighbors`	Maximum distance of searching for appearance neighbors (path length).	3
`enrichmentSimilarityThreshold`	Threshold for finding appearance neighbors, [0,1].	0.85
`enrichmentSigmaS (initializationSigmaS)`	Parameter in similarity computation in the enrichment (initialization) step.	max. double val (100.0)
`enrichmentSigmaR (initializationSigmaR)`	Parameter in similarity computation in the enrichment (initialization) step.	0.1 (0.1)
`enrichmentRandomPairSamplingRatio (initializationRandomPairSamplingRatio)`	Determines how many pixel samples per superpixel will be used for the similarity computation in the enrichment (inititialization) step.	0.1 (0.1)
`useMatchingInitialization`	Flag for using the loose superpixel matching for initializing the labels (alternative: random initialization).	true
`epsBase`	Guided Filter parameter for smoothing strength.	0.01^2
`filterSize`	Kernel width of the Guided Filter. Depends on `radius`.	19
`medianFilterRadius`	Kernel width of the postprocessing filter.	17

`gamma`	Weight computation parameter for median filter in postprocessing.	$\frac{10.0}{255.0}$
`useBilateralFilterWeights`	Flag for using Bilateral Filter weights in postprocessing (alternative: custom weight computation)	true
`bilatSigmaS`	Parameter for Bilateral Filter weights in postprocessing.	7.0
`bilatSigmaR`	Parameter for Bilateral Filter weights in postprocessing.	$\frac{20.0}{255.0}$
`beta`	Weight for Census-related part of the cost function.	0.8
`gamma1`	Truncation threshold for intensity difference in the cost function.	0.039
`gamma2`	Truncation threshold for gradient difference in the cost function.	0.008

B.2 Third-Party Components

The proposed method implementation is done in Java SE 8 [4] and uses the following third-party components and libraries:

- ImageJ (image processing program) [3]
- jSLIC v.0.30 (superpixel segmentation, a plugin for ImageJ) [18]
- PortableFloatMap class for reading and writing pfm-files [54] (modified)
- GuidedFilter class [22] (ported from C++ to Java, modified)
- EJML v.0.29 (linear algebra library) [1]

Bibliography

[1] EJML (Efficient Java Matrix Library). http://ejml.org. Accessed: 2016-11-24.

[2] HCI Robust Vision Challenge. https://hciweb.iwr.uni-heidelberg.de/benchmarks/Challenging_Data_for_Stereo_and_Optical_Flow. Accessed: 2016-11-18.

[3] ImageJ. http://imagej.net/. Accessed: 2016-11-24.

[4] Java SE 8. http://www.oracle.com/technetwork/java/javase/overview/index.html. Accessed: 2016-11-24.

[5] KITTI Vision Benchmark. http://www.cvlibs.net/datasets/kitti/. Accessed: 2016-11-18.

[6] Matlab Image Processing Toolbox™. https://de.mathworks.com/products/image/. Accessed: 2016-11-24.

[7] Middlebury Stereo Evaluation. http://vision.middlebury.edu/stereo/. Accessed: 2016-11-18.

[8] OpenCV - Open Source Computer Vision Library. http://opencv.org/. Accessed: 2016-11-24.

[9] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Susstrunk, S. SLIC superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 34*, 11 (Nov. 2012), 2274–2282.

[10] Antunes, M., and Barreto, J. P. Efficient stereo matching using histogram aggregation with multiple slant hypotheses. In *Pattern Recognition and Image Analysis - 6th Iberian Conference, IbPRIA 2013, Funchal, Madeira, Portugal, June 5-7, 2013. Proceedings* (2013), pp. 1–10.

[11] Barnes, C., Shechtman, E., Goldman, D. B., and Finkelstein, A. The generalized PatchMatch correspondence algorithm. In *European Conference on Computer Vision* (Sept. 2010).

[12] Besse, F., Rother, C., Fitzgibbon, A. W., and Kautz, J. PMBP: PatchMatch belief propagation for correspondence field estimation. *International Journal of Computer Vision 110*, 1 (2014), 2–13.

[13] Bleyer, M., and Gelautz, M. A layered stereo algorithm using image segmentation and global visibility constraints. In *International Conference on Image Processing*. IEEE, 2004, pp. 2997–3000.

[14] Bleyer, M., Rhemann, C., and Rother, C. PatchMatch Stereo - stereo matching with slanted support windows. In *Proceedings of the British Machine Vision Conference* (2011), BMVA Press, pp. 14.1–14.11. http://dx.doi.org/10.5244/C.25.14.

[15] Bleyer, M., Rother, C., and Kohli, P. Surface stereo with soft segmentation. In *CVPR* (2010), IEEE Computer Society, pp. 1570–1577.

[16] Bleyer, M., Rother, C., Kohli, P., Scharstein, D., and Sinha, S. N. Object stereo - joint stereo matching and object segmentation. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011* (2011), pp. 3081–3088.

[17] Bobick, A. F., and Intille, S. S. Large occlusion stereo. *International Journal of Computer Vision 33* (1999), 181–200.

[18] Borovec, J. jSLIC superpixels. `http://imagej.net/CMP-BIA_tools#jSLIC_-_superpixels`. Accessed: 2016-11-24.

[19] Boykov, Y., Ulen, J., and Olsson, C. In defense of 3D-label stereo. *2013 IEEE Conference on Computer Vision and Pattern Recognition 00*, undefined (2013), 1730–1737.

[20] Boykov, Y., Veksler, O., and Zabih, R. A variable window approach to early vision. *IEEE Trans. Pattern Anal. Mach. Intell. 20*, 12 (1998), 1283–1294.

[21] Buades, A., and Facciolo, G. Reliable multiscale and multiwindow stereo matching. *SIAM J. Imaging Sciences 8*, 2 (2015), 888–915.

[22] Çetin, A. GuidedFilter class. `https://github.com/atilimcetin/guided-filter`. Accessed: 2016-11-24.

[23] Do, M. N. Cross-based local multipoint filtering. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Washington, DC, USA, 2012), CVPR '12, IEEE Computer Society, pp. 430–437.

[24] Dumont, M., Goorts, P., Maesen, S., Degraen, D., Bekaert, P., and Lafruit, G. Iterative refinement for real-time local stereo matching. In *2014 International Conference on 3D Imaging, IC3D 2014, Liège, Belgium, December 9-10, 2014* (2014), pp. 1–8.

[25] Fusiello, A., Roberto, V., and Trucco, E. Efficient stereo with multiple windowing. In *1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Proceedings* (1997), Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 858–863. 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 97), SAN JUAN, PR, JUN 17-19, 1997.

[26] Gerrits, M., and Bekaert, P. Local stereo matching with segmentation-based outlier rejection. In *The 3rd Canadian Conference on Computer and Robot Vision, 2006.* (June 2006), IEEE, p. 66.

[27] Gupta, A., Efros, A. A., and Hebert, M. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV European Conference on Computer Vision* (2010).

[28] Hartley, R., and Zisserman, A. *Multiple View Geometry in Computer Vision.*, 2 ed. Cambridge University Press, 2003.

[29] He, K., Sun, J., and Tang, X. Guided image filtering. In *Proceedings of the 11th European Conference on Computer Vision: Part I* (Berlin, Heidelberg, 2010), ECCV'10, Springer-Verlag, pp. 1–14.

[30] Heise, P., Klose, S., Jensen, B., and Knoll, A. PM-Huber: PatchMatch with Huber regularization for stereo matching. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013* (2013), pp. 2360–2367.

[31] Hirschmüller, H. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02* (Washington, DC, USA, 2005), CVPR '05, IEEE Computer Society, pp. 807–814.

[32] Hirschmüller, H., Innocent, P. R., and Garibaldi, J. Real-time correlation-based stereo vision with reduced border errors. *Int. J. Comput. Vision 47*, 1-3 (Apr. 2002), 229–246.

[33] Hirschmüller, H., and Scharstein, D. Evaluation of cost functions for stereo matching. In *CVPR* (2007), IEEE Computer Society.

[34] Hirschmüller, H., and Scharstein, D. Evaluation of stereo matching costs on images with radiometric differences. *IEEE Trans. Pattern Anal. Mach. Intell. 31*, 9 (Sept. 2009), 1582–1599.

[35] Hong, L., and Chen, G. Segment-based stereo matching using graph cuts. In *CVPR* (2004), pp. 74–81.

[36] Horbach, M. Real-time stereo matching on GPU., 2016.

[37] Hosni, A., Bleyer, M., Gelautz, M., and Rhemann, C. Local stereo matching using geodesic support weights. In *Proceedings of the 16th IEEE International Conference on Image Processing* (Piscataway, NJ, USA, 2009), ICIP'09, IEEE Press, pp. 2069–2072.

[38] Huang, X., Yuan, C., and Zhang, J. Graph cuts stereo matching based on PatchMatch and ground control points constraint. In *Advances in Multimedia Information Processing - PCM 2015 - 16th Pacific-Rim Conference on Multimedia, Gwangju, South Korea, September 16-18, 2015, Proceedings, Part II* (2015), pp. 14–23.

[39] Huang, X., Zhang, J., Wu, Q., Yuan, C., and Fan, L. Dense correspondence using non-local DAISY forest. In *2015 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2015, Adelaide, Australia, November 23-25, 2015* (2015), pp. 1–8.

[40] Jiao, J., Wang, R., Wang, W., Dong, S., Wang, Z., and Gao, W. Cost-volume filtering-based stereo matching with improved matching cost and secondary refinement. In *2014 IEEE International Conference on Multimedia and Expo (ICME)* (July 2014), pp. 1–6.

[41] Kanade, T., and Okutomi, M. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE Trans. Pattern Anal. Mach. Intell. 16*, 9 (Sept. 1994), 920–932.

[42] Klaus, A., Sormann, M., and Karner, K. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *Proceedings of the 18th International Conference on Pattern Recognition -*

Volume 03 (Washington, DC, USA, 2006), ICPR '06, IEEE Computer Society, pp. 15–18.

[43] Kowalczuk, J., Psota, E., and Pérez, L. C. Real-time stereo matching on CUDA using an iterative refinement method for adaptive support-weight correspondences. *IEEE Trans. Circuits Syst. Video Techn. 23*, 1 (2013), 94–104.

[44] Li, Y., Min, D., Brown, M. S., Do, M. N., and Lu, J. SPM-BP: sped-up PatchMatch belief propagation for continuous MRFs. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015* (2015), pp. 4006–4014.

[45] Lu, J., Yang, H., Min, D., and Do, M. N. PatchMatch Filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2013).

[46] Manduchi, R., and Tomasi, C. Distinctiveness maps for image matching. In *Image Analysis and Processing, 1999. Proceedings. International Conference on* (1999), pp. 26–31.

[47] Mei, X., Sun, X., Dong, W., Wang, H., and Zhang, X. Segment-tree based cost aggregation for stereo matching. In *CVPR* (2013), IEEE Computer Society, pp. 313–320.

[48] Mei, X., Sun, X., Zhou, M., Jiao, S., Wang, H., and Zhang, X. On building an accurate stereo matching system on graphics hardware. In *ICCV Workshops* (2011), IEEE, pp. 467–474.

[49] Min, D., Lu, J., and Do, M. N. A revisit to cost aggregation in stereo matching: How far can we reduce its computational redundancy? In *ICCV* (2011), D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, Eds., IEEE Computer Society, pp. 1567–1574.

[50] Min, D. B., and Sohn, K. Cost aggregation and occlusion handling with WLS in stereo matching. *IEEE Trans. Image Processing 17*, 8 (2008), 1431–1442.

[51] Mordohai, P. The self-aware matching measure for stereo. In *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009* (2009), pp. 1841–1848.

[52] Mordohai, P., and Hu, X. A quantitative evaluation of confidence measures for stereo vision. *IEEE Transactions on Pattern Analysis Machine Intelligence 34*, undefined (2012), 2121–2133.

[53] Nalpantidis, L., and Gasteratos, A. Biologically and psychophysically inspired adaptive support weights algorithm for stereo correspondence. *Robotics and Autonomous Systems 58*, 5 (2010), 457–464.

[54] Nayuki. PortableFloatMap class. `https://www.nayuki.io/page/portable-floatmap-format-io-java`. Accessed: 2016-11-24.

[55] Rhemann, C., Hosni, A., Bleyer, M., Rother, C., and Gelautz, M. Fast cost-volume filtering for visual correspondence and beyond. In *IEEE Computer Vision and Pattern Recognition (CVPR)* (2011).

[56] Saygili, G., van der Maaten, L., and Hendriks, E. A. Stereo similarity metric fusion using stereo confidence. In *22nd International Conference on Pattern Recognition, ICPR 2014, Stockholm, Sweden, August 24-28, 2014* (2014), pp. 2161–2166.

[57] Saygili, G., van der Maaten, L., and Hendriks, E. A. Adaptive stereo similarity fusion using confidence measures. *Comput. Vis. Image Underst. 135*, C (June 2015), 95–108.

[58] Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nesic, N., Wang, X., and Westling, P. High-resolution stereo datasets with subpixel-accurate ground truth. In *GCPR* (2014), vol. 8753 of *Lecture Notes in Computer Science*, Springer, pp. 31–42.

[59] Scharstein, D., and Pal, C. Learning conditional random fields for stereo. In *CVPR* (2007), IEEE Computer Society.

[60] Scharstein, D., and Szeliski, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision 47*, 1-3 (Apr. 2002), 7–42.

[61] Scharstein, D., and Szeliski, R. High-accuracy stereo depth maps using structured light. In *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 2003), CVPR'03, IEEE Computer Society, pp. 195–202.

[62] Tomasi, C., and Manduchi, R. Bilateral filtering for gray and color images. In *Proceedings of the Sixth International Conference on Computer Vision* (Washington, DC, USA, 1998), ICCV '98, IEEE Computer Society, pp. 839–.

[63] Tombari, F., Mattoccia, S., and Di Stefano, L. Segmentation-based adaptive support for accurate stereo correspondence. In *Proceedings of the 2Nd Pacific Rim Conference on Advances in Image and Video Technology* (Berlin, Heidelberg, 2007), PSIVT'07, Springer-Verlag, pp. 427–438.

[64] Tombari, F., Mattoccia, S., di Stefano, L., and Addimanda, E. Near real-time stereo based on effective cost aggregation. In *19th International Conference on Pattern Recognition (ICPR 2008), December 8-11, 2008, Tampa, Florida, USA* (2008), pp. 1–4.

[65] Unger, C., and Ilic, S. A stochastic cost function for stereo vision. In *Proceedings of the British Machine Vision Conference* (2014), BMVA Press.

[66] Van Meerbergen, G., Vergauwen, M., Pollefeys, M., and Van Gool, L. A hierarchical symmetric stereo algorithm using dynamic programming. *Int. J. Comput. Vision 47*, 1-3 (Apr. 2002), 275–285.

[67] Veksler, O. Fast variable window for stereo correspondence using integral images. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (2003), pp. 556–561.

[68] Wang, Z.-F., and Zheng, Z.-G. A region based stereo matching algorithm using cooperative optimization. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008.* (June 2008), IEEE, pp. 1–8.

[69] Woodford, O., Torr, P., Reid, I., and Fitzgibbon, A. Global stereo reconstruction under second-order smoothness priors. *IEEE Trans. Pattern Anal. Mach. Intell. 31*, 12 (Dec. 2009), 2115–2128.

[70] Xu, S., Zhang, F., He, X., Shen, X., and Zhang, X. PM-PM: PatchMatch with Potts model for object segmentation and stereo matching. *IEEE Trans. Image Processing 24*, 7 (2015), 2182–2196.

[71] Yang, Q., Wang, L., and Yang, R. Real-time global stereo matching using hierarchical belief propagation. In *Proceedings of the British Machine Vision Conference* (2006), BMVA Press, pp. 101.1–101.10. doi:10.5244/C.20.101.

[72] Yang, Q., Wang, L., Yang, R., Stewenius, H., and Nister, D. Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 31*, 3 (Mar. 2009), 492–504.

[73] Yoon, K.-J., and Kweon, I. S. Adaptive support-weight approach for correspondence search. *IEEE Trans. Pattern Anal. Mach. Intell. 28*, 4 (Apr. 2006), 650–656.

[74] Zhang, C., Li, Z., Cai, R., Chao, H., and Rui, Y. *As-Rigid-As-Possible Stereo under Second Order Smoothness Priors.* Springer International Publishing, Cham, 2014, pp. 112–126.

[75] Zhang, K., Fang, Y., Min, D., Sun, L., Yang, S., Yan, S., and Tian, Q. Cross-scale cost aggregation for stereo matching. In *CVPR* (2014).

[76] Zhang, K., Lu, J., and Lafruit, G. Cross-based local stereo matching using orthogonal integral images. *IEEE Trans. Circuits Syst. Video Techn. 19*, 7 (2009), 1073–1079.

[77] Zhang, Y., Gong, M., and Yang, Y.-H. Local stereo matching with 3D adaptive cost aggregation for slanted surface modeling and sub-pixel accuracy. In *ICPR* (2008), IEEE Computer Society, pp. 1–4.