

Dense Stereo Matching for Urban Outdoor Scenes

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Visual Computing

eingereicht von

Katrin Lasinger

Matrikelnummer 0726670

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dr. Margrit Gelautz
Externe Betreuung: Prof. Dr. Konrad Schindler
Silvano Galliani

Wien, 01.12.2014

(Unterschrift Verfasserin)

(Unterschrift Betreuung)

Dense Stereo Matching for Urban Outdoor Scenes

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Visual Computing

by

Katrin Lasinger

Registration Number 0726670

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Dr. Margrit Gelautz
External Advisors: Prof. Dr. Konrad Schindler
Silvano Galliani

Vienna, 01.12.2014

(Signature of Author)

(Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Katrin Lasinger
Feldgasse 34, 3425 Langenlebar

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

(Ort, Datum)

(Unterschrift Verfasserin)

Acknowledgements

I am using this opportunity to express my gratitude to all the people who supported me during the creation of this master thesis. Special thanks go to my supervisor Silvano Galliani who supported me throughout the whole process of my thesis with helpful insights and feedback. Further, I would like to thank Prof. Konrad Schindler for giving me the opportunity to work on this project and for many fruitful discussions and suggestions. I would also like to thank Christoph Vogel for his help with the Kitti Vision Benchmark and for useful comments. I want to thank the whole group of Photogrammetry and Remote Sensing at ETH Zurich for giving me such a warm welcome in Switzerland and for the great experiences we shared, both research related and non-research related.

Furthermore, I would like to thank Jafar Amiri Parian and PhotoCore for providing high resolution panoramic image sequences that are used in our qualitative evaluation. I also want to thank Markus Gerke from the University of Twente for the provision of aerial image data.

In addition, I want to thank the people who supported me during the past few months aside from my scientific life. I am deeply grateful for the unconditional support of my family for all my decisions and ventures. Thank you for always standing by me and being there whenever I need you. I also want to thank all the new friends and amazing flatmates I met in Zurich in the past nine months. Thank you Alessandro, Mado and all the others who enriched my life in Switzerland in so many different ways.

Abstract

Dense stereo matching is an active research topic in the area of Computer Vision. Depth information is extracted from a dense correspondence search between two or more images of the same scene, taken from different camera positions. Extracted depth information can be used for various applications such as robotic navigation, automated driving or 3D reconstruction of objects and buildings.

In this work we will focus on dense stereo matching for urban outdoor environments. We start from the recently published PatchMatch Stereo approach by Bleyer et al. [8] since it seems suitable for our purpose in terms of memory consumption and scalability for high resolution images. We further extend their idea to multi-view stereo. Our algorithm is tested on different urban outdoor image sets, including image pairs from cameras mounted on a car, panoramic images of urban areas as well as multi-view data from historic sites and aerial image data. For the correspondence search, experiments with different cost functions are performed.

PatchMatch Stereo is a local stereo matching approach that estimates a 3D plane at each pixel position, hence, extracting not only disparity values but also surface normals. The PatchMatch Stereo algorithm is based on a randomized approximate correspondence search. Initially a random plane is selected for each pixel position. Good plane estimates are then propagated to neighboring pixels and further refined in an iterative process.

We transform the PatchMatch Stereo approach to scene space in order to directly estimate depth values and work with non-rectified images. Mapping from one image to another is facilitated by plane induced homographies, utilizing the estimated planes (normal and depth) at each pixel position. Processing in scene space allows us to directly combine multiple images.

The major contribution of our work is a multi-view stereo matching approach. The use of more than two images facilitates the handling of partially occluded image regions and therefore leads to more robust results. Our approach is quantitatively evaluated on existing benchmark data for two-view and multi-view image sequences. Results are compared with reported values of state-of-the-art stereo matching methods.

Kurzfassung

Dichtes Stereomatching ist ein aktives Forschungsgebiet im Bereich der Computer Vision. Ziel ist es, Tiefeninformationen aus zwei oder mehr 2D-Bildern einer Szene zu extrahieren. Hierfür wird eine Korrespondenzsuche über alle Pixel der verwendeten Bilder angewandt. Ermittelte Tiefeninformation kann für verschiedene Anwendungen verwendet werden. Beispiele sind die automatisierte Navigation von Robotern und Autos oder die 3D-Rekonstruktion von Gegenständen und Gebäuden.

In dieser Arbeit werden wir uns auf dichtes Stereomatching für urbane Outdoor-Bereiche konzentrieren. Der kürzlich publizierte PatchMatch Stereo Ansatz von Bleyer et al. [8] scheint in Hinsicht auf Speicherverbrauch und Skalierbarkeit für hochauflösende Bilder für unsere Zwecke geeignet. Wir starten von dieser Idee und erweitern den Ansatz, um die Verarbeitung von mehr als zwei Bildern zu ermöglichen. Wir testen unseren Algorithmus an verschiedenen Bilddaten im urbanen Außenbereich: Stereobilder aufgenommen von einem fahrenden Auto, Panoramabilder aus städtischen Gebieten, Bildsequenzen von historischen Stätten und Luftbilder. Für die Korrespondenzsuche werden Experimente mit unterschiedlichen Kostenfunktionen durchgeführt.

PatchMatch Stereo ist ein lokaler Stereomatching Ansatz, der an jeder Pixelposition eine 3D-Ebene schätzt. Dadurch werden nicht nur Disparitätswerte sondern auch Oberflächennormalen ermittelt. Der PatchMatch Stereo Algorithmus basiert auf einer randomisierten, approximierten Korrespondenzsuche. Zunächst wird für jede Pixelposition eine zufällige Ebene gewählt. Gute Ebenenschätzungen, die niedrige Matching-Kosten aufweisen, werden daraufhin an benachbarte Pixel weitergegeben und in einem iterativen Prozess weiter verfeinert.

Wir transformieren den PatchMatch Stereoansatz vom Disparitätsraum in den 3D Szenenraum, um eine direkte Bestimmung von Tiefenwerten zu ermöglichen. Dies ermöglicht zusätzlich das Arbeiten mit nicht-rektifizierten Bildpaaren. Die Abbildung von einem Kamerabild zum anderen wird durch Ebenen-induzierte Homographien ermöglicht. Hierfür wird die geschätzte Ebene (Normale und Tiefenwert) an jeder Pixelposition verwendet. Das Arbeiten im Szenenraum ermöglicht die direkte Verarbeitung von mehr als zwei Bildern, da keine Rektifizierung notwendig ist.

Dies führt zum Hauptbeitrag dieser Arbeit: ein Multi-View Stereo Matching-Ansatz. Die Verwendung von mehr als zwei Bildern erleichtert die Handhabung von teilweise verdeckten Bildbereichen und führt dadurch zu robusteren Ergebnissen. Unser Ansatz wird quantitativ auf bestehenden Benchmarks für 2-View und Multi-View Bildsequenzen ausgewertet. Die Ergebnisse werden des Weiteren mit anderen State-of-the-Art Stereomatching Methoden verglichen.

Contents

1	Introduction	1
1.1	Problem Statement	1
1.2	Test Data	3
1.3	Methodological Approach	5
1.4	Structure of the Work	5
2	Stereo Geometry	7
2.1	Camera Geometry	7
2.2	Epipolar Geometry	9
2.3	Plane-Induced Homographies	12
2.4	Summary	14
3	State of the Art	15
3.1	Local Stereo Matching	16
3.2	Cost Computation	18
3.3	PatchMatch Stereo	19
3.4	Panoramic and High-Resolution Images	23
3.5	Multi-View Stereo	24
3.6	Summary	25
4	Multi-View Stereo Approach	27
4.1	Two-View Stereo Matching	27
4.2	Cost Functions	30
4.3	Scene Space PatchMatch Stereo	30
4.4	Multi-View PatchMatch Stereo	32
4.5	Summary	35
5	Results	37
5.1	Two-view Stereo	37
5.2	Multi-view Stereo	52
5.3	Summary	59
6	Conclusion	61

Introduction

Depth information is a useful cue for multiple vision related applications. For example, it can be used in autonomous navigation systems or as additional cue for object detection and scene understanding. In urban outdoor scenes, accurate 3D measurement of buildings and urban environment is relevant in areas such as surveying, architecture, civil engineering and cultural heritage. Furthermore, depth information extracted from cameras mounted on a car can be used to warn drivers about potential obstacles or for autonomous driving by capturing the environment around the car.

Different technologies for depth extraction exist, varying in application area, processing time and price. In this work, we will focus on dense stereo matching where depth information is extracted from two or more 2D images. Hence, no special hardware is required for data acquisition. Depth information from stereo image pairs can be extracted in a similar way as humans perceive binocular depth from their left and right eyes [25]. In two horizontally displaced views far away objects will be roughly at the same position in both images. However, the closer the objects are, the farther apart their positions will be in the two images. This can be tested for example by placing a finger in front of the face and alternately closing the left and right eye. The position of the finger will jump left and right relative to the farther away background [54].

When processing images, the pixel distance of corresponding points in the two image views is commonly called *disparity*. If the camera setup (including internal parameters as well as relative location and orientation of the two cameras) is known, the depth value can be computed from the disparity. The difficult part of dense stereo matching is to find corresponding pixels among the considered camera images.

1.1 Problem Statement

The focus of this master thesis lies on dense stereo matching for urban outdoor scenes. Thus, for each pixel in one image the corresponding pixel position in the other image views needs to be found. In order to find corresponding pixels a cost function is defined which measures the

dissimilarity of two pixels or pixel patches. Compared to indoor scenes, common challenging issues in outdoor scenes are changing lighting conditions as well as sky and cluttered vegetation areas. Hence, robust cost functions in terms of illumination changes are needed.

A problem of two-view stereo is that non-overlapping or occluded areas cannot be recovered. Figure 1.1 shows an exemplary image pair from the Middlebury Stereo Vision Benchmark [45] together with the corresponding occlusion map for the left image. Image regions that are either occluded by other objects in the scene or not visible due to the changed view point of the second camera are marked in black. To solve this issue, stereo information from multiple views needs to be combined.

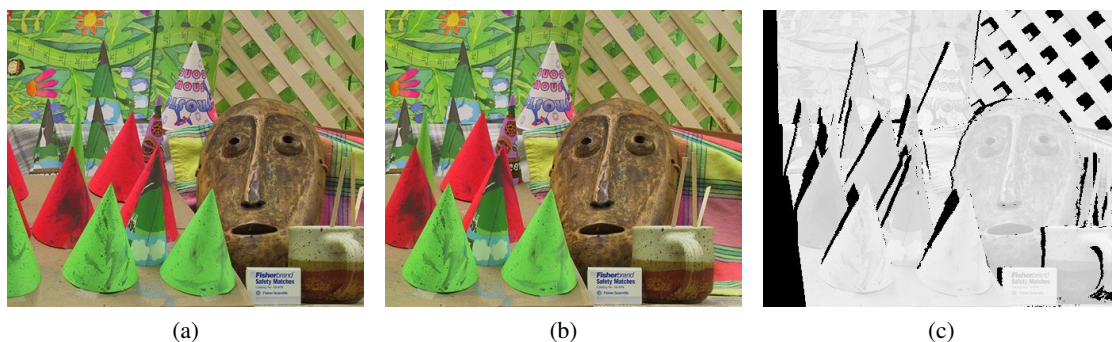


Figure 1.1: Occluded regions of a stereo image pair: (a) left image, (b) right image, (c) occlusion map (black) overlaid on grayscale version of left image. Images taken from Middlebury Stereo Vision Benchmark [45].

We aim to extend the PatchMatch Stereo algorithm by Bleyer et al. [8] in order to fulfill the requirements of urban outdoor scenes. PatchMatch Stereo is a local stereo matching algorithm. Hence, for a pixel only its local neighborhood is taken into consideration during dissimilarity estimation for the correspondence search. The most similar pixel patch in the second image in terms of color and gradient information is considered for disparity estimation. A randomized approximate nearest neighbor search facilitates the handling of continuous disparity values and the additional estimation of a surface normal at each pixel position allows the support of slanted surfaces. Hence, reconstructions are not biased towards fronto-parallel planes, which is often the case in traditional stereo methods. The normal estimation can furthermore be used as additional cue for 3D surface reconstruction. Further benefits of the approach are the low memory consumption and its linear scale in computation time with respect to the number of pixels. This makes the algorithm also suitable for high resolution images.

We aim to tackle the mentioned problems for urban outdoor scenes by extending the original PatchMatch Stereo algorithm in two directions. First, we utilize census transform [63] as cost functions for the correspondence search, which has been proven to perform well in outdoor scenarios. We will evaluate its performance compared to the original cost function used by Bleyer et al. on different outdoor datasets. Second, we will reformulate the PatchMatch Stereo approach in scene space which allows us to directly work with depth values instead of disparity values. This will prevent the need for rectification of image pairs (mapping epipolar lines to

horizontal scanlines). Furthermore, planes will be estimated in 3D scene space instead of the disparity space. Since we work in a common scene coordinate system, information from multiple views can be directly fused into a single depth map by combining cost values from different views. This leads to our multi-view stereo approach. The benefit of a multi-view approach is that partial occlusions can be directly resolved by valid matches with other image views. Areas that are occluded in one image view might be still matchable in other views. Furthermore, redundant information, e.g. consistent matches over multiple views, can help to reduce noise or to measure reliability of matches.

1.2 Test Data

For a qualitative analysis of our approach we will use images from a high resolution panoramic camera as well as aerial image data. Both datasets come without ground truth information. The high resolution panoramic images (about 300 megapixels per image) are obtained from a 360° panorama camera with an underlying cylindrical projection. Special issues of this specific type of camera are the non-linear epipolar geometry, induced by the cylindrical projection, and potentially high memory consumption due to the high resolution, inducing also a large disparity range. We will not address the specific geometry of the camera in this thesis but rather work directly on rectified images, where correspondence search can be performed along the horizontal scanline. An example of two panoramic images, viewing the same scene, is shown in Figure 1.2.



Figure 1.2: Two panoramic images of an urban scene from horizontally displaced views.

Our second used dataset for qualitative evaluation consists of aerial image sequences obtained from cameras mounted on an airplane. Since more than two views of a scene are provided, we will use this dataset to evaluate our multi-view stereo matching implementation. Oblique aerial image data is provided by Slagboom & Peeters¹. A special characteristic of this dataset,

¹<http://www.slagboomenpeeters.com/>

compared to other image data used in our evaluation, is the elevated viewing position. One of the aerial image sequences is shown in Figure 1.3.



Figure 1.3: Sequence of aerial image data captured from an airplane.

Additionally, we will quantitatively evaluate our algorithm on benchmark datasets for urban outdoor scenes. An evaluation will be performed on the Kitti Vision Benchmark Suite [17] as well as the Multi-View Stereo dataset by Strecha et al. [51]. Images in the Kitti Vision Benchmark were captured from a driving car. Hence, dominant objects in the images are streets, houses, cars and vegetation. An exemplary image pair is depicted in Figure 1.4. The Strecha dataset contains outdoor images of a castle and its surroundings from multiple views. Images from one of the provided sequences are shown in Figure 1.5. Both datasets provide internal and external camera calibration data and ground truth data captured with a laser scanner. We will compare quantitative results of our algorithm on those datasets with reported results from other approaches. Furthermore, a comparison between our two-view and our multi-view matching approach will be given for the Kitti dataset.



Figure 1.4: Image pair from Kitti Vision Benchmark Suite [17].

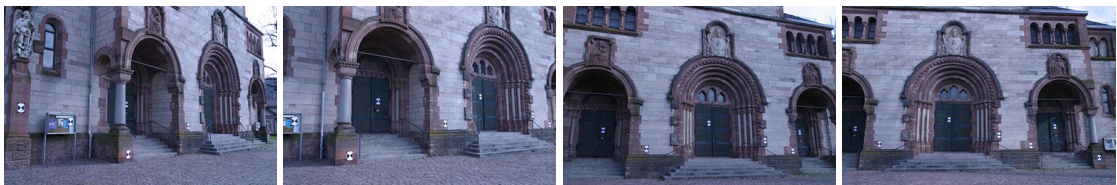


Figure 1.5: Image sequence from Multi-View Stereo dataset by Strecha et al. [51].

1.3 Methodological Approach

As mentioned before, a promising local stereo matching approach, especially in terms of memory consumption, is the PatchMatch Stereo algorithm by Bleyer et al. [8]. The algorithm is based on a randomized correspondence algorithm for approximate nearest neighbor search by Barnes et al. [4]. Concerning memory consumption, per pixel only the current best plane (disparity value plus normal) and the current cost need to be stored.

Hence, as a first step we have reimplemented the PatchMatch Stereo algorithm and tested it on the Kitti Vision benchmark as well as on high-resolution panoramic images. Additionally, different cost functions were tested. E.g. Census Transform [63] has proven to be robust in challenging outdoor lighting situations [42]. We further investigated potential speed-up options. Bao et al. [3] use a self-similarity measurement to consider only the n most similar neighboring pixels instead of the whole support window. Zinner et al. [67] introduced Sparse Census Transform, an adaption of the original Census Transform approach in order to speed up computation time.

As a major contribution of our work we moved from disparity space to scene space. Benefits of working directly in scene space are that no rectification of the input images is required and that depth and plane estimation are directly performed in scene space. Furthermore, we can extend this approach to multiple views, since calculations are performed in a common coordinate system. Plane-induced homographies can be used to transform a 2D pixel coordinate from one camera view to another. In order to handle multiple views, the cost computation needs to be extended accordingly. Hence, cost values between multiple views need to be fused. A simple fusing approach is to accumulate over all two-view cost values. However, more sophisticated ways to fuse the data are also investigated in this thesis. The quantitative performance of the algorithm is evaluated on existing benchmarks for stereo vision.

1.4 Structure of the Work

After this brief introduction into the topic, in Chapter 2 we will give a detailed description of the relevant geometric properties for stereo reconstruction. We will cover camera geometry and epipolar geometry. Moreover, plane-induced homographies will be explained which are used to map points between different image views. In Chapter 3 an overview of existing stereo matching approaches is given with a special focus on the PatchMatch Stereo algorithm. The methodological approach of this thesis is presented in Chapter 4, describing the implementation of the two-view approach and its extension to scene space and multi-view stereo matching. An extensive evaluation of the results is conducted in Chapter 5. Finally, the conclusion and an outlook for future work are provided in Chapter 6.

Stereo Geometry

This chapter introduces the mathematical concepts used for our dense stereo matching approach. First of all, in Section 2.1 we define the used camera model and its geometric principles. The mapping from 3D scene space to 2D image space as well as the definition of depth will be given. Subsequently, the epipolar geometry between two camera views will be described in Section 2.2. Epipolar geometry defines the geometric relation between two camera views and the thereby induced constraints in the correspondence search space. Finally, we introduce the concept of plane-induced homographies in Section 2.3 in order to map image points from one view to the other. The content of this chapter is mainly based on Hartley and Zisserman [19], Chapters 6, 9 and 12.

2.1 Camera Geometry

In this section we will focus on finite projective cameras, which can be used to model most commonly available still cameras. The simplest model of a finite camera is the pinhole camera model. The camera is defined by a focal length f , a center of projection or camera center \mathbf{C} and a principal axis. No lenses are considered in this simple model. A graphical representation of the basic pinhole camera model is depicted in Figure 2.1. The focal length specifies the distance between the center of projection and the image plane onto which the 3D scene is projected. The image plane is orthogonal to the principal axis. The intersection point between the principal axis and the image plane is called principal point \mathbf{p} . We refer to the local coordinate system of the image plane as the *image space*. The principal point is not necessarily located at the coordinate origin but might have a small offset from the origin.

Camera Parameters

In order to locate the camera in 3D world space we define the *extrinsic camera parameters*. They are given by a translation of the camera center \mathbf{C} from the world space origin and rotation of the principal axis. The rotation can be expressed by a 3x3 rotation matrix \mathcal{R} . The 3D coordinates of

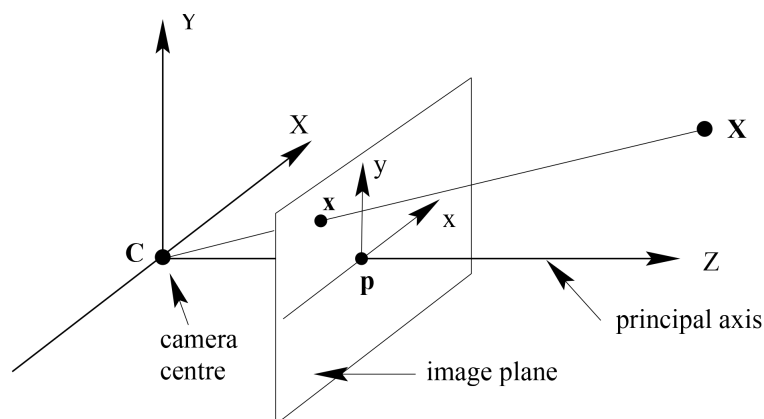


Figure 2.1: Geometric representation of a pinhole camera with the camera center in the coordinate origin and the principal axis pointing in the direction of the z axis. A 3D point \mathbf{X} in world space is projected onto the image plane at the 2D image coordinate \mathbf{x} . Image taken from Hartley and Zisserman [19].

the camera center represent the translation of the camera. Instead of the explicit representation of the camera center one can also state the translation vector $\mathbf{t} = -\mathcal{R}\mathbf{C}$.

We refer to the *intrinsic parameters* of a camera as the internal geometric setup of the camera. This includes the focal length f and the 2D coordinates of the principal point in image space ($\mathbf{p} = (p_x, p_y)^T$). In order to extend the basic pinhole model to CCD cameras, we additionally need to consider the non-squared dimensions of pixels. Thus, we introduce a scale factor for each axial direction (m_x, m_y). Focal length and principal point are represented in pixel dimensions as $\alpha_x = fm_x, \alpha_y = fm_y$ and $x_0 = m_x p_x, y_0 = m_y p_y$. For general finite projective cameras a skew parameter s is further introduced. The intrinsic parameters of the camera can be combined in a 3x3 camera calibration matrix \mathcal{K} :

$$\mathcal{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (2.1)$$

Extrinsic and intrinsic camera parameters can be combined into a 3x4 homogeneous *camera projection matrix* $\mathcal{P} = \mathcal{K}[\mathcal{R}|\mathbf{t}]$. Note that the camera projection matrix does not take into account nonlinear intrinsic camera parameters such as radial lens distortion. Hence, images need to be corrected accordingly in order to resemble the result obtained under a linear projective camera model. E.g. a straight line method [11] can be used to determine the distortion function. Lens correction can be also performed jointly with rectification if rectified images are required for the stereo matching process (see Section 2.2 for details on rectification).

2D to 3D Mapping

The camera projection matrix maps from 3D world coordinate space to the 2D image coordinate space of the camera:

$$\mathbf{x} = w(x, y, 1)^T = \mathcal{P}\mathbf{X}. \quad (2.2)$$

The values x and y represent the 2D coordinates in image space. If the camera matrix is normalized the value w is the depth of the point \mathbf{X} . A camera is normalized if the determinant of the first 3x3 submatrix \mathcal{M} of \mathcal{P} is greater than zero and the length of the third row of this submatrix \mathbf{m}^3 is one. In the scope of this thesis we only deal with normalized camera matrices. In any case, normalization can be achieved by multiplying the matrix by an appropriate factor. The depth is defined as the distance of a point to the principal plane of the camera. The principal plane is parallel to the image plane (normal to the principal axis), passing through the camera center.

In the same way as we project 3D points to the image plane we can back-project 2D image points to rays in 3D. We can represent the back-projected line as the join of two points on the ray: the camera center \mathbf{C} and the image point \mathbf{x} back-projected onto the plane at infinity. When writing $\mathcal{P} = [\mathcal{M}|\mathbf{p}_4]$, the camera center is given by $\mathbf{C} = -\mathcal{M}^{-1}\mathbf{p}_4$. Furthermore, the point at infinity can be written as $D = ((\mathcal{M}^{-1}\mathbf{x})^T, 0)^T$. The ray is then defined as the join of these two points:

$$\mathbf{X}(\omega) = \omega \begin{pmatrix} \mathcal{M}^{-1}\mathbf{x} \\ 0 \end{pmatrix} + \begin{pmatrix} -\mathcal{M}^{-1}\mathbf{p}_4 \\ 1 \end{pmatrix} = \begin{pmatrix} \mathcal{M}^{-1}(\omega\mathbf{x} - \mathbf{p}_4) \\ 1 \end{pmatrix}. \quad (2.3)$$

The original 3D position \mathbf{X} can only be recovered if the depth ω of the 2D point is known.

2.2 Epipolar Geometry

The epipolar geometry describes the geometric relationship of two camera views. It depends on the intrinsic parameters of the cameras and their relative pose to each other. Figure 2.2 depicts the basic setup of the epipolar geometry. The two cameras are represented by their camera centers \mathbf{C} and \mathbf{C}' and the corresponding image planes. The connection between the camera centers is called baseline. The epipole \mathbf{e} of a camera is the intersection point between its image plane and the baseline.

For an image point \mathbf{x} of a camera image an epipolar plane can be constructed, passing through the image point, the epipole and the camera center. Or in other words, the plane is determined by the baseline and the ray from the camera center to the point \mathbf{X} in 3D scene space, passing through the image point \mathbf{x} . The intersecting line of the image plane and the epipolar plane is called epipolar line l . All epipolar lines pass through the epipole. A corresponding image point \mathbf{x}' in the second camera view has to fall on its epipolar line l' defined by the epipolar plane. Hence, the search space for a corresponding point is constrained to a one dimensional search along the epipolar line. The correct position on the line depends on the depth of the point in 3D scene space.

An important factor for the obtaining of reliable stereo matches is the baseline length between two cameras. Due to the different viewing angles, wide baselines can lead to different appearances of corresponding points in the two views. Furthermore, occlusions are more likely to occur at wider baselines. On the other hand, a narrow baseline results in a less accurate depth estimation, especially for distant regions. Hence, a good trade-off for the baseline length is desirable, although not always possible due to technical limitations.

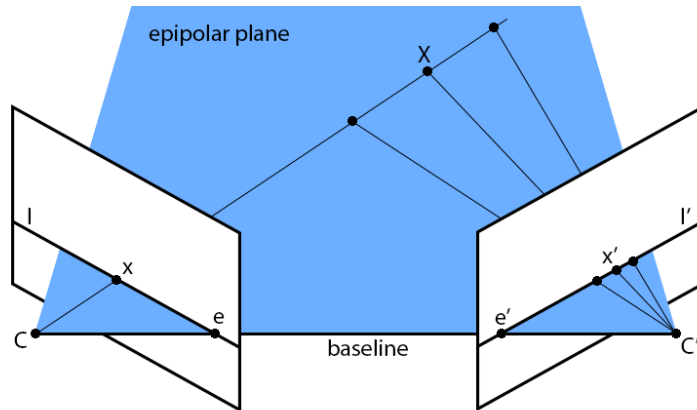


Figure 2.2: Epipolar geometry setup of two cameras (represented by their image planes and camera centers C and C'). The correspondence search for an image point x in the second camera image can be constrained to a one dimensional search on the epipolar line l' .

Rectification

A common transformation used to simplify the search for correspondences is called rectification. Rectification maps epipolar lines to horizontal scanlines [54]. This can be achieved by virtually rotating and translating the two cameras in such a way that their viewing directions are parallel to each other and perpendicular to the baseline. Furthermore, the cameras need to be horizontally aligned and their focal lengths need to be the same (can be adjusted by rescaling the image). Hence, they share a common image plane and their epipoles lie at infinity. This results in horizontal epipolar lines, simplifying the correspondence search problem. The rectification process is depicted in Figure 2.3.

For rectified images, the pixel distance between two corresponding points in the two images is called *disparity*. When searching for a corresponding point in the right image we can start at the pixel position of the left image point (disparity 0) and walk along the horizontal scanline to the left until the image border or the maximal considered disparity value is reached. The disparity range depends on the scene (closer objects result in higher disparity values) and the distance between the two cameras (baseline). For an image point (x, y) in the left image and a certain disparity value d the corresponding image point (x', y') in the right image can be obtained with the following formula:

$$(x', y') = (x - d, y). \quad (2.4)$$

Similarly, when moving from the right image to the left image the disparity value has to be added to the x coordinate. In Figure 2.4 we illustrate the setup of a rectified image pair and a horizontal scanline. The house is horizontally displaced in the two image views, whereas the vertical location does not change. Furthermore, the far away sun stays at the same image location in both views. The bigger the disparity value, the closer the object is to the camera. Respectively, farther away points have a smaller pixel distance.

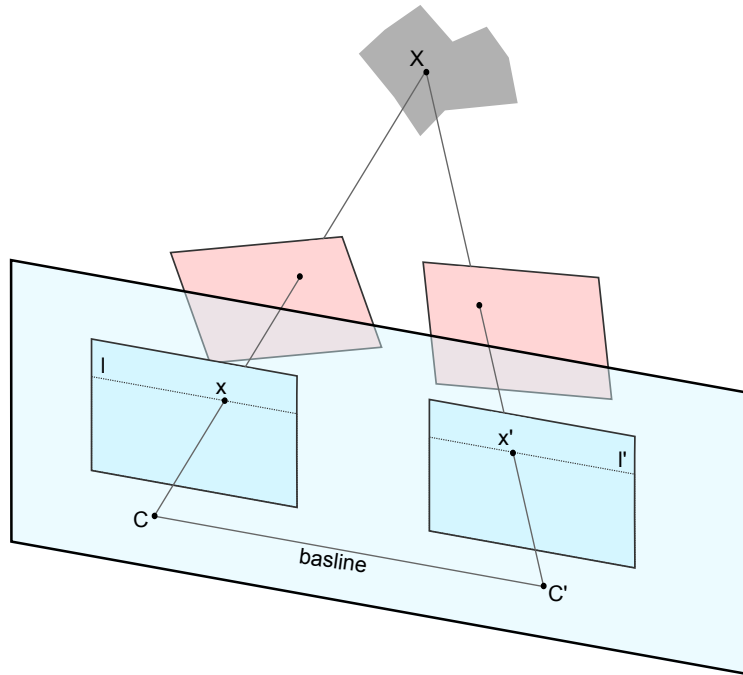


Figure 2.3: Image rectification of stereo image pair by projecting the two camera views onto a common image plane. Original views are depicted in red and their rectified projections are shown in blue. The baseline between the two cameras is parallel to the common image plane and, hence, the epipoles lie at infinity.

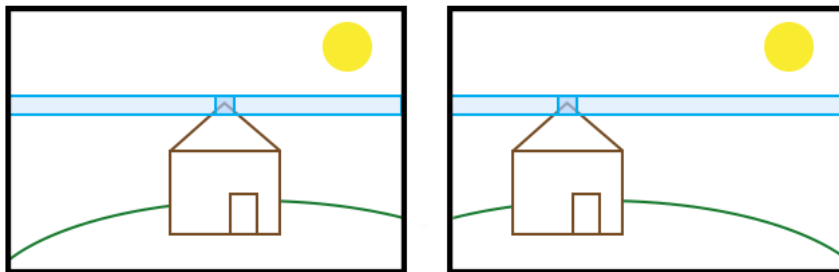


Figure 2.4: Example of a rectified image pair with a horizontal epipolar scanline in blue.

Depth vs. Disparity

The depth of a point in the scene is inversely proportional to the disparity value. If the distance between the cameras and the focal length f are known for a rectified image pair, the depth can be computed with the following formula:

$$depth = f \frac{baseline}{disparity}. \tag{2.5}$$

The mapping between disparity and depth values is in general non-linear. This is due to the fact that the step size of disparity values is uniform along the epipolar line in an image view. However, this uniform mapping in image space does not lead to a uniform mapping in 3D space. Depth resolution decreases for farther away points. This effect is depicted in Figure 2.5. The uniform sampling along the epipolar line leads to a non-uniform sampling in the depth interval.

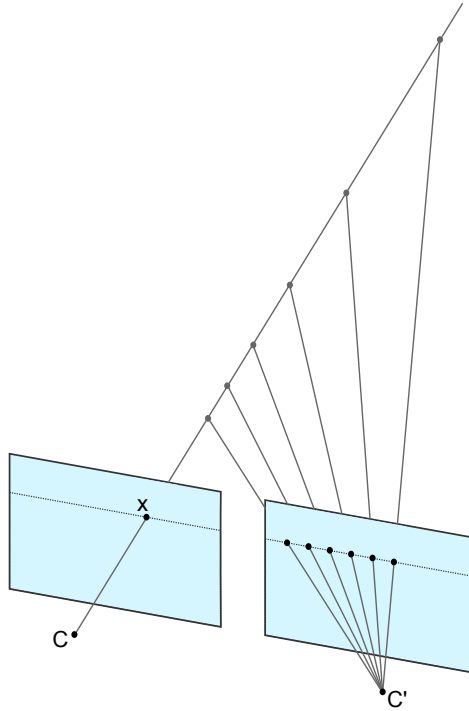


Figure 2.5: Stereo setup with a uniformly sampled epipolar line resulting in a non-uniformly sampled depth interval. Image adapted from Tola et al. [56].

2.3 Plane-Induced Homographies

As stated in the previous section, the search for corresponding points within two stereo views can be limited to a one-dimensional search along the epipolar line. For rectified images these epipolar lines correspond to horizontal scanlines. Hence, correspondence search is performed along these scanlines and the disparity is determined by the horizontal pixel distance between two corresponding points (see Equation 2.4).

For a general setup (non-rectified images), epipolar lines do no longer correspond to horizontal scanlines. Thus, the correspondence search along epipolar lines needs to be adapted accordingly. Given an image point \mathbf{x} and a depth estimate ω for one image view we want to determine the location of the corresponding image point \mathbf{x}' in another view. This can be achieved by determining the 3D location \mathbf{X} according to Equation 2.3 and mapping the 3D point back

to the desired 2D image view utilizing Equation 2.2. However, if a plane π is given in the 3D space, we can directly compute the corresponding image point \mathbf{x}' for the desired view, without the need of computing the 3D point. The 3D plane π and the camera parameters of the two images induce a homography \mathcal{H} . This homography allows a direct mapping between the two image planes. The setup is depicted in Figure 2.6.

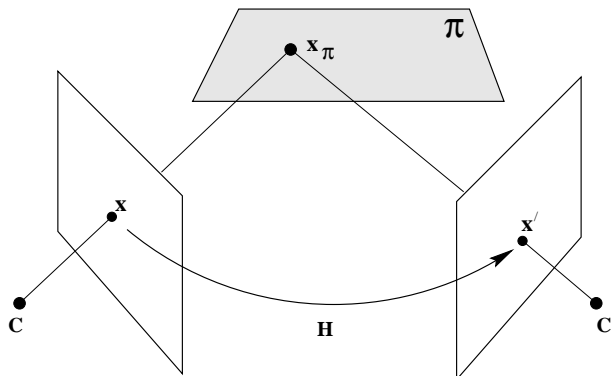


Figure 2.6: Homography \mathcal{H} from image point \mathbf{x} to \mathbf{x}' in the other image plane, induced by a plane π in scene space. Image taken from Hartley and Zisserman [19].

We consider only the case of one projection matrix having the form $\mathcal{P} = \mathcal{K}[\mathcal{I}|\mathbf{0}]$. This means the camera is located at the origin of the coordinate system and has zero rotation. We call this camera the *reference camera*. A general setup can be translated to this special form by transforming the two cameras accordingly.

The plane in scene space is defined as $\pi = (\mathbf{n}^T, d)^T$, with \mathbf{n} being the unit normal vector of the plane and d the distance of the plane to the coordinate origin (position of the reference camera center). With the projection matrix of the second camera being of the form $\mathcal{P}' = \mathcal{K}'[\mathcal{R}|\mathbf{t}]$, the resulting homography is defined as:

$$\mathcal{H}_\pi = \mathcal{K}'(\mathcal{R} - \mathbf{t}\mathbf{n}^T/d)\mathcal{K}^{-1}. \quad (2.6)$$

Hence, we can use the following formula to get from a pixel position \mathbf{x} in one view and a certain plane π in scene space to the corresponding pixel position \mathbf{x}' in the second view:

$$\mathbf{x}' = \mathcal{H}_\pi \mathbf{x}. \quad (2.7)$$

Multiple cameras

This setup can be extended to multiple cameras by computing the corresponding homographies for all cameras (see Figure 2.7). E.g. the homography $\mathcal{H}_{\pi,1}$ for the plane π from reference camera C_r to camera C_1 can be computed from the relative rotation and translation of camera C_1 to camera C_r and the two calibration matrices \mathcal{K} and \mathcal{K}' with Equation 2.6. In order to move from a target camera to the reference camera the corresponding homography can be inverted (e.g. $\mathcal{H}_{\pi,1}^{-1}$). Hence, movement between two arbitrary cameras is possible. E.g. to move from camera C_3 to camera C_2 the homography $\mathcal{H} = \mathcal{H}_{\pi,2}\mathcal{H}_{\pi,3}^{-1}$ can be used.

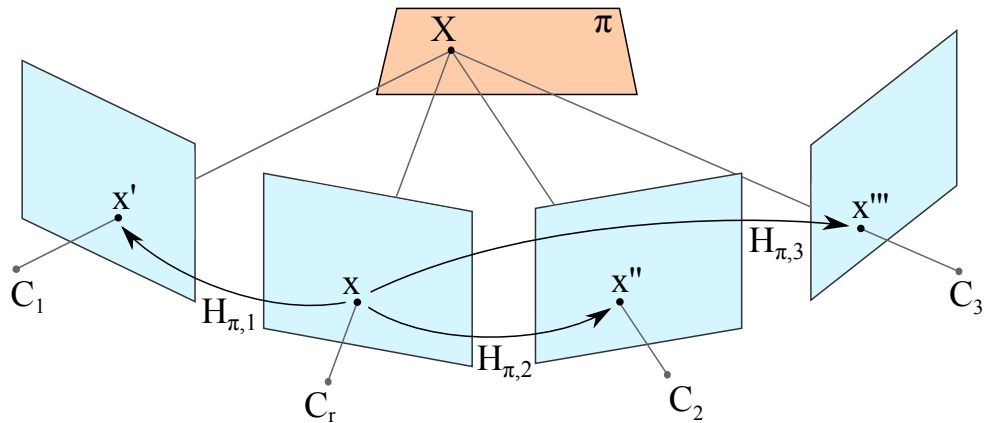


Figure 2.7: Multi-view setup with four cameras and homographies from reference camera C_r to three other cameras.

2.4 Summary

In this chapter we have covered the basic geometric principles that are necessary for our stereo matching approach. We have introduced the camera projection matrix \mathcal{P} that defines the camera location and orientation in 3D world space and the intrinsic camera parameters, such as focal length and principal point offset. Furthermore, we have described the epipolar geometry of two camera views. The epipolar geometry allows us to limit the search space for corresponding points in image pairs. Finally, the concept of plane-induced homographies was introduced in order to map a image point from one camera view to arbitrary other views, based on a plane defined in 3D space.

State of the Art

In dense stereo matching, for each pixel in one view we want to find the corresponding pixel location in another view. Based on the offset in the pixel locations (disparity), the depth value can be computed. Most two-view stereo algorithms work with rectified images in order to simplify the correspondence search to a one dimensional search on a horizontal scanline. In 2002, Scharstein and Szeliski [45] reviewed existing dense two-frame stereo matching approaches and introduced the Middlebury Stereo Vision Benchmark. Although new methods have been presented since then, the basic principles still apply for current approaches. Four main steps can be recognized in stereo matching methods: matching cost computation, cost aggregation, disparity computation and optimization, and disparity refinement. Furthermore, methods can be divided into local and global stereo matching approaches.

The matching cost indicates the dissimilarity of two corresponding pixels in the image pair. E.g. the absolute or squared difference of intensity or color values can be computed. For cost aggregation, the matching cost is accumulated over a support window of neighboring pixels. Cost aggregation is commonly used for local stereo matching approaches, but can also be applied for global methods. In many cases, cost computation and aggregation are implicitly combined. This is for example the case for normalized cross correlation or census transform [63]. For the disparity computation and optimization step, local methods select the pixel location with the minimum overall cost (winner-take-all). Global methods include explicit smoothing assumptions, e.g. by solving an energy minimization problem, with the energy function consisting of a data and a smoothness term. The data term again describes the dissimilarity of pixels or image patches and the smoothness term enforces smoothness in the disparity variation of neighboring pixels. The final disparity refinement step can include sub-pixel refinement or cross-checking of left and right disparity maps in order to detect occluded or inconsistent areas. For many approaches, a clear classification into local or global methods is not possible. E.g. the semi-global matching approach by Hirschmüller [21] combines concepts of both local and global stereo matching approaches. A global optimization is approximated by pathwise optimization of a global cost function.

We chose a local stereo matching approach for our implementation since it seems to be better scalable in terms of memory consumption and runtime for high-resolution images compared to global approaches. Hence, in the following section we will focus on local stereo matching approaches. In Section 3.2 we will compare different cost computation methods that are relevant for urban outdoor scenes. Particularly interesting for our purpose is the PatchMatch Stereo algorithm by Bleyer et al. [8], which will be described in detail in Section 3.3. Furthermore, we will refer to related work in the field of panoramic and high resolution stereo matching in Section 3.4. Finally, related multi-view stereo matching approaches will be reviewed in Section 3.5.

3.1 Local Stereo Matching

Stereo matching is based on the idea that corresponding points in two images are locally similar in their appearance [32]. In local stereo matching approaches the disparity selection of a pixel depends solely on the similarity of two support windows around potential corresponding pixels in the two image views. Commonly, the support window is a square pixel patch centered around the current pixel of interest. In the second view a support window is selected for the corresponding pixel at a certain disparity value. The dissimilarity or cost between the two support windows is computed based on a cost function, e.g. sum of squared intensity differences [45].

A traditional way to select the best disparity value is to compute a cost volume or disparity space image (DSI) [28, 61]. It can be seen as a three dimensional data structure $C(x, y, d)$ that stores the cost for a point pair with the coordinates (x, y) in the left and $(x - d, y)$ in the right rectified image. Hence, we get a cost value for each considered disparity value d for an image point (x, y) in the reference image. Using the winner-take-all strategy, the disparity value with the minimum cost is selected for an image point. Global methods introduce an additional smoothness cost and use global optimization to solve the disparity search problem, usually resulting in high computational complexity [45].

One drawback of the computation of a cost volume is that only a discrete number of disparity values can be handled. Disparity range and step size are limited by the available memory space. Some methods therefore use sub-pixel refinement as a post-processing step in order to obtain smooth and continuous disparity values (e.g. [44, 60]). A further possible post-processing step is the cross-checking of the left-to-right and right-to-left disparity maps to detect inconsistent areas that might occur due to occlusions [14]. Additionally, a filter can be applied in order to reduce noise in the disparity image. Potential noise reduction filters are the bilateral filter [57] and the median filter [45].

Besides discretized disparity values, another drawback of common local stereo matching approaches is the bias towards fronto-parallel surfaces. If a constant disparity for a local support window is assumed, reconstruction will be biased towards surfaces parallel to the image plane. This, especially in combination with discrete disparity steps, can lead to staircase effects when slanted surfaces are reconstructed (e.g. the ground plane of a scene). The problem is depicted in Figure 3.1a. With discrete disparity values and fronto-parallel support windows the true shape of the surface cannot be recovered correctly. Instead, the curved surface is represented by a flat plane at disparity value 1. Figure 3.2b illustrates the resulting staircase effect on a corridor scene when using discrete disparity values and no support for slanted surfaces.

Slanted support windows can be used to overcome the mentioned problem (see Figure 3.1b). Sub-pixel disparity changes can be recovered and curved regions are approximated more accurately. However, handling slanted support windows and continuous disparity values requires a bigger cost volume (including also plane parameters) and a reasonable selection of candidate planes (e.g. [16, 65]). Bleyer et al. [8] presented an alternative approach to solve this problem by utilizing a randomized approximate correspondence search. Their approach circumvents the necessity of computing a full cost volume and allows the estimation of continuous disparity values as well as the support of slanted surfaces. Without the need of computing a cost volume, the approach is suitable for high resolution images and big disparity ranges. The method will be discussed in detail in Section 3.3. A reconstruction of the corridor scene utilizing the Patch-Match Stereo approach is shown in Figure 3.2c. Note that the staircase effects at the floor and walls disappear and also curved objects are reconstructed reasonably.

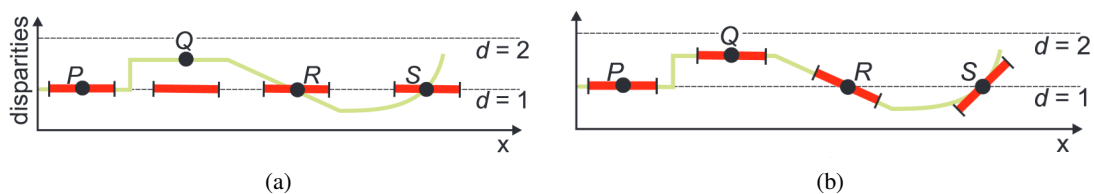


Figure 3.1: Comparison of (a) fronto-parallel support windows at integer disparities and (b) slanted support windows at continuous disparities. The red bars represent the support windows and the green curve shows the true surface that we want to reconstruct (in 1D). Images taken from Bleyer et al. [8].

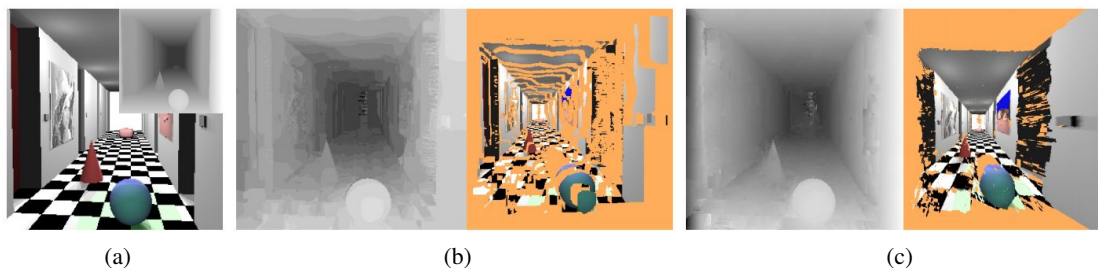


Figure 3.2: Illustrating staircase effect for slanted surfaces: (a) Original left image and ground truth disparity map of a scene with slanted ground floor and walls as well as rounded objects (b) Reconstruction with integer disparity values and fronto-parallel bias (c) Reconstruction with slanted support windows. Images taken from Bleyer et al. [8].

3.2 Cost Computation

The matching cost denotes the dissimilarity of two pixels or image patches. For local stereo matching approaches the cost is usually aggregated over a local support window. This leads to implicit smoothing of the depth estimation, since a proposed depth or disparity value needs the support of the majority of pixels in the image patch. Different methods exist to measure this cost value. Classical similarity measures are commonly based on sum of absolute or squared differences (SAD or SSD) or normalized cross-correlation (NCC) [2, 45]. SAD and SSD aggregate respectively absolute and squared intensity or color differences between two image patches. Due to the normalization, the NCC score is more robust to exposure changes between images than SSD or SAD. The NCC score indicates the correlation between two patches. Its value is within the range $[-1, 1]$, with bigger values meaning higher correlation. For the actual cost value one can use for example the negative NCC score.

Nonparametric cost functions

Nonparametric techniques, such as rank transform and census transform form another group of similarity measures [63]. Instead of relying on intensity values themselves they are based on the relative ordering of intensities. Hence, the method is robust to changes in image gain and bias (since no absolute intensity values are used) and performance near object boundaries can be improved (due to intrinsic outlier handling). The rank transform computes the number of pixels in a patch whose intensity value is lower than the center intensity value. Census transform additionally includes location information by computing a signature string for an image patch with value 0 if a pixel's intensity is lower than the center pixel intensity and 1 otherwise. The Hamming distance between the signature strings of two image patches indicates their dissimilarity. Stein [50] extended this approach by a third state for similar intensity values. Therefore, an additional parameter ϵ is introduced. If the intensity difference between two pixel intensities is less than ϵ , these pixels are considered as similar. Hence, the following formula is used to compute the digit for pixel p' on a signature string with center pixel p :

$$\xi(p, p') = \begin{cases} 0 & p - p' > \epsilon \\ 1 & |p - p'| \leq \epsilon \\ 2 & p' - p > \epsilon \end{cases} \quad (3.1)$$

Another adaption of the original census transform is the sparse census transform [67], where only every second row and column is processed. This allows a faster computation and thus larger window sizes. The authors claim that the accuracy is nearly as good as with the original results. Fife and Archibald [13] confirmed these results and experimented with additional sparse patterns and variants.

Adaptive support window

Classical similarity measures for support windows have problems at depth discontinuities, since objects with different disparity values move by a different amount of pixels in the stereo image

pair. To overcome this problem, Yoon and Kweon introduced the weighted support window [62]. The weight of each pixel in the support window is based on its photometric and geometric distance to the center pixel. The underlying assumption is that pixels with a similar color or intensity value are more likely to belong to the same object than pixels with different values. If pixels belong to the same object, there is a higher chance that their disparity values are the same (or they lie on the same plane in case of slanted support windows). Hence, these pixels get a higher weight in order to have a higher influence on the result.

The adaptive weight function proposed by Yoon and Kweon is similar to bilateral filtering [57]. An extensive study on different adaptive support weight approaches for local stereo matching was performed by Hosni et al. [24]. They evaluated the quantitative and runtime performance of different approaches on a matching algorithm with fronto-parallel windows and on PatchMatch Stereo (introduced in Section 3.3), which uses slanted support windows. For the PatchMatch Stereo approach, where no full cost volume computation is required, the bilateral weights performed best in their study.

In the PatchMatch Stereo approach a weight function similar to the original approach proposed by Yoon and Kweon was used. However, this weight function is solely dependent on RGB color information, since additional position information did not show a significant improvement. The following weight function was used:

$$w(p, q) = e^{-\frac{\|I_p - I_q\|}{\gamma}}. \quad (3.2)$$

The parameter γ is defined by the user and $\|I_p - I_q\|$ is the L_1 -distance of the RGB color values of the pixels p and q . The authors suggested a value of 10 for the parameter γ with the RGB values being in the range of $[0, 255]$ per color component. Furthermore, the dissimilarity function used by Bleyer et al. combines truncated absolute differences of color and gradient information [9, 10, 43]. The pixel dissimilarity between two pixels q and q' is computed as follows:

$$\rho(q, q') = (1 - \alpha) \cdot \min(\|I_q - I_{q'}\|, \tau_{col}) + \alpha \cdot \min(\|\nabla I_q - \nabla I_{q'}\|, \tau_{grad}). \quad (3.3)$$

The user-defined parameter α denotes the respective weighting of color and gradient dissimilarity. The parameters τ_{col} and τ_{grad} are used to truncate the dissimilarity values in order to increase robustness to outliers. The components of the two dimensional gradient vectors ∇I_q and $\nabla I_{q'}$ are in the range of $[0, 255]$.

A drawback of adaptive support windows is that intensity or color edges do not necessarily correspond with disparity edges. This is for example the case for highly textured surfaces. Hence, potentially valid information from differently textured areas of the same object get a low weighting. Nevertheless, on average the use of adaptive support windows seems to significantly improve performance at disparity discontinuities.

3.3 PatchMatch Stereo

In 2011, Bleyer et al. [8] introduced PatchMatch Stereo, a local stereo matching approach based on approximate nearest neighbor search. Two major benefits of this approach are the support

of slanted surfaces and the direct estimation of floating point disparity values. Furthermore, instead of computing a memory-demanding cost volume for the full disparity range, only a limited number of 3D planes is tested at each pixel position and only the current best plane and cost is stored. The randomized disparity estimation is based on an approximate nearest neighbor search where good guesses are propagated to neighbors. The mentioned advantages of the algorithm are also beneficial for our problems. The disparity d for a point p can be computed from a 3D plane π with the following formula:

$$d(\mathbf{p}, \pi) = a_\pi p_x + b_\pi p_y + c_\pi. \quad (3.4)$$

The variables a_π , b_π and c_π represent the three plane parameters of π and the parameters p_x and p_y are the x and y coordinates of the image point p . In the following subsections we will explain the basic idea of the randomized correspondence search and describe the PatchMatch Stereo algorithm in detail. Furthermore, some adaptations of the PatchMatch Stereo algorithm will be reviewed.

Randomized Correspondence Search

The search for the best approximating plane is based on the approximate nearest neighbor search algorithm PatchMatch by Barnes et al. [4]. The idea was originally introduced for interactive image editing in order to quickly find corresponding patches in an image. The search of nearest neighbors for an entire image traditionally requires a high computational cost, inhibiting a real-time execution. The authors present an approximate nearest neighbor search based on the assumption that there exist large coherent areas in the image. If one image patch finds a good match, it can propagate its findings to neighboring patches. The three basic phases of the algorithm are depicted in Figure 3.3. First, all patches are initialized with random assignments. The subsequent iteration phase consists of a propagation step and a random search step. In the propagation step patches propagate their matches to local neighbors, if those matches improve the matching cost of the neighbors. In the random search step other random candidates are tested within a search radius around the current best match.

Algorithm

Bleyer et al. adapted the PatchMatch algorithm for the correspondence search between stereo image pairs. In order to support slanted surfaces, the randomized search is used to find a best approximating 3D plane for each pixel location. A cost function needs to be defined in order to quantitatively compare different plane assumptions. This cost function computes the dissimilarity of corresponding patches. Bleyer et al. used the following cost function, accounting for slanted support windows:

$$m(p, \pi_p) = \sum_{q \in W_p} w(p, q) \cdot \rho(q, q'_{\pi_p}). \quad (3.5)$$

The cost of pixel p for plane π_p is determined by aggregating over all pixels q of the support window W_p . The adaptive weight $w(p, q)$ is based on the color similarity of the neighboring pixel

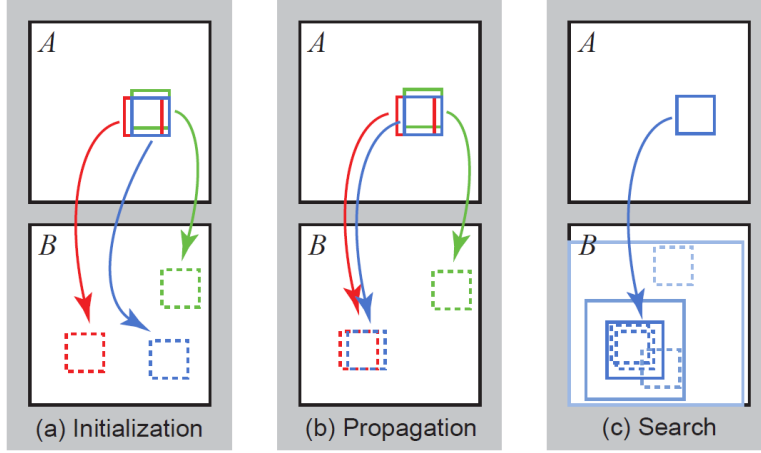


Figure 3.3: PatchMatch: Randomized nearest neighbor search using (a) random initialization, (b) propagation of good matches and (c) randomized search for better matches within neighborhood. Image taken from Barnes et al. [4].

q to the center pixel p (see Equation 3.2). The function $\rho(q, q'_{\pi_p})$ computes the dissimilarity of a patch around pixel q and its corresponding patch around pixel q'_{π_p} in the second image. It is based on color and gradient information and was originally introduced by Rhemann et al. [43]. The disparity at a certain pixel position for the current plane can be computed utilizing Equation 3.4. Based on this disparity value the pixel location in the second view can be computed:

$$q'_{\pi_p} = q \mp d(q, \pi_p). \quad (3.6)$$

A negative sign is used when mapping from left to right image view and a positive sign is used when mapping from right to left.

In order to estimate the 3D plane and consequently the disparity value at each pixel position the following steps are executed:

- **Random Initialization:** Each pixel is initialized with a random 3D plane. In order to evenly sample the space a random unit vector \mathbf{n} for the plane normal and a disparity value d within the defined disparity range are selected as plane parameters.
- **Iteration:** The following four steps are iteratively performed. According to the authors, three iterations are sufficient.
 - **Spatial Propagation:** The cost $m(p, \pi_p)$ of a pixel p for the current plane π_p is compared with the cost $m(p, \pi_q)$ for a neighboring plane π_q . If the neighboring plane improves the cost, the plane π_p is replaced by π_q . In even iterations the left and upper neighbors are considered and in odd iterations the right and lower neighbors are checked. An illustration of the spatial propagation step is shown in Figure 3.4.

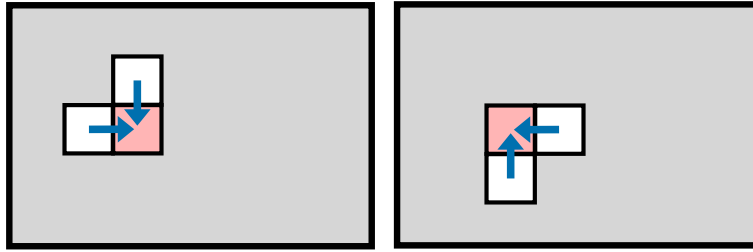


Figure 3.4: Spatially propagating planes from neighboring pixels to the current pixel (red): in even iterations propagating good guesses from left and top neighbors and in odd iterations propagating from right and bottom neighbors.

- **View Propagation:** If plane estimation is performed for both image views, good guesses from one view can be propagated to the other. Corresponding image points should have a similar plane in both image views. Hence, if the matching point of a pixel p' in the second view is the current pixel p , the cost $m(p, \pi_{p'})$ is compared with the current cost $m(p, \pi_p)$. Again the plane is replaced by the new plane $\pi_{p'}$ if it yields a lower cost. The view propagation step is depicted in Figure 3.5.

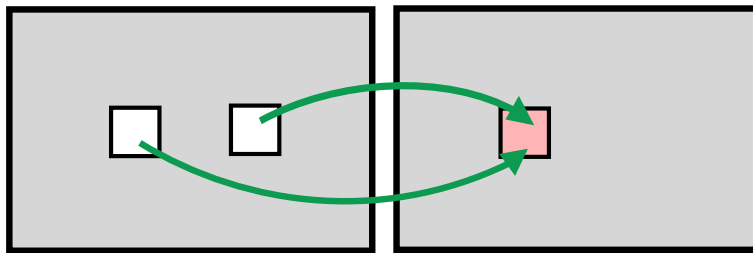


Figure 3.5: Propagating planes from the second image view for all pixels that have the current pixel (red) as matching point.

- **Temporal Propagation:** If stereo video sequences are available, planes can also be propagated over time. Again, the cost of the current plane is compared with the matching cost for the plane of the pixel in the previous video frame.
- **Plane Refinement:** This step is comparable to the search step in the original Patch-Match approach. The current best plane π_p for a pixel p is iteratively refined by randomly selecting a different plane within an iteratively decreasing search space, limiting the maximum change in disparity and normal direction.
- **Post Processing:** In order to detect inconsistent (potentially occluded) areas in the two estimated disparity maps (for the left and right view), left/right cross checking of disparity values is performed. Hence, for a pixel p and an estimated disparity value d in the left view the corresponding pixel location p' in the right view is computed using Equation 3.6. The disparity value in the second view at location p' is then compared with the disparity value of p in the first view. If their absolute difference is bigger than a given threshold the two pixel locations are labeled as inconsistent and their estimated disparity values are discarded. These inconsistent pixels are later filled by searching for the closest consistent

left and right neighboring planes and assigning the lower of the two disparity values to the current pixel. Assigning the lower disparity value is based on the assumption that occlusions occur in the background. Additionally a weighted median filter is applied on the filled-in disparity values in order to remove horizontal streak artifacts.

The approach performed well at the Middlebury Stereo Vision Benchmark [45], achieving rank 2 for the error threshold 0.5 (sub-pixel performance test) at its initial introduction in 2011. Additionally, the algorithm is suitable for high resolution images and high disparity ranges due to its low memory requirements (no need to store an entire cost volume in memory) and linear scale in computation time with respect to the number of pixels (in contrast to most global stereo matching approaches). However, the authors reported quantitative results only for low-resolution and highly textured indoor scenes, captured under laboratory conditions. No results were shown for outdoor scenes, where lighting conditions are usually less stable than indoors.

Adaptions

Various adaptations of the original PatchMatch Stereo approach have been published in recent years. Besse et al. [5] combined the method with a global particle belief propagation approach in order to model explicit smoothing. Another global stereo matching approach was proposed by Heise et al. [20]. In their approach the PatchMatch algorithm was combined with an explicit variational smoothing method. Both approaches achieved improved results in the Middlebury Stereovision Benchmark with the drawback of higher computational complexity. Heise et al. additionally formulated the plane estimation in scene space, using plane-induced homographies as described in Section 2.3. However, their implementation was based on rectified images, working in disparity space only. Hence, in the scope of their paper only results from rectified image pairs were shown.

A recent approach by Bao et al. [3] uses the PatchMatch idea of approximate nearest neighbor search for optical flow estimation. Instead of considering all pixels of a support window for matching, only the n most similar pixels to the window center pixel are used. This significantly reduces computation time for the cost computation. Additionally, the adaptive support weights of the used neighbors can be precomputed. PatchMatch is used as approximate search for the selection of the n most similar pixels as well as for the flow estimation. Another contribution of the paper is the presentation of a hierarchical matching strategy for further reduce of computation cost.

3.4 Panoramic and High-Resolution Images

Different acquisition methods, camera models and geometric relations for panoramic stereo images have been presented [27, 29, 37]. In general the assumption of straight epipolar lines no longer holds for panoramic images. E.g. for a single-center cylindrical panoramic camera the epipolar lines are sine curves [37]. Huang et al. [27] generalized single-center, multi-perspective and concentric panoramic images into a unified model of polycentric panoramas. They further derived epipolar curve equations for this generalized model. However, only little work has been done on the actual matching of panoramic stereo pairs (e.g. [34, 47, 52]).

Kang and Szeliski [34] presented an omni-directional stereo reconstruction approach from single-center cylindrical panoramic images. Different reconstruction approaches were compared. The first two approaches use only sparse feature points and do not limit the search space by epipolar constraints. In the third approach dense stereo matching was performed, using epipolar geometry to constraint the search space. A denser 3D mesh was recovered from the later method with the drawback of higher computational complexity. The main benefit of the use of panoramic images is the resulting wide field of view for depth estimation.

Concerning high resolution images, the traditional approach of evaluating a full cost volume reaches its limits w.r.t memory consumption, especially when considering slanted surfaces and continuous disparity values. Some recent approaches start from reliable, sparse keypoint matches and aim to propagate information to remaining areas [18, 48]. Another possible approach is to divide the image into smaller overlapping tiles and process them independently [22]. Tola et al. [56] presented a multi-view stereo reconstruction approach for high-resolution image sets. This is achieved by the use of DAISY descriptors [55] for efficient dense stereo matching. Furthermore, the PatchMatch Stereo approach is feasible for high resolution images, since the processing of a full cost volume is avoided.

3.5 Multi-View Stereo

Depth estimation from multiple views as extension to two-view stereo estimation can be divided into two steps. First, a single depth map can be generated from information obtained from multiple views [39]. As a subsequent step, multiple depth maps can be fused into a combined 3D representation of the scene [35, 53]. Szeliski reviewed different multi-view stereo matching approaches [54, Sec. 11.6].

In order to obtain a single depth map of a reference view, cost values from multiple corresponding stereo pairs of the reference view need to be fused. E.g. Okutomi and Kandade [39] accumulated the SSD costs of multiple views to a combined cost, the sum of sum of squared distances (SSSD). However, views where the support window is fully or partially occluded contribute to the accumulated cost and potentially impair results.

Kang et al. [33] presented various approaches for robust occlusion handling. Local techniques include temporal selection and adaptive window size. Instead of combining all available costs, temporal selection only considers a sub-selection of costs. The authors propose two different selection approaches. The first approach is to pick the best 50% of all available views. Alternatively, one can select either all predecessor or all successor frames to the reference view. Occlusion often occurs only in one moving direction, i.e. motion is smooth at high frame-rate. For example, an occluding foreground object moves from the right to the left when the images are ordered from left to right (see Figure 3.6). The adaptive window size approach aims to deal with untextured areas. The algorithm starts with a small window size and gradually increases the size in later iterations. If a depth value is already considered as reliable in earlier iterations this pixel position is not processed in later iterations. If the local variance of the cost function around a depth value is too low, the depth is considered as unreliable. Hence, these unreliable pixels will be processed again in later iterations. As an alternative solution, the authors propose

to use global techniques by introducing an additional smoothness term and optimizing a global energy function.

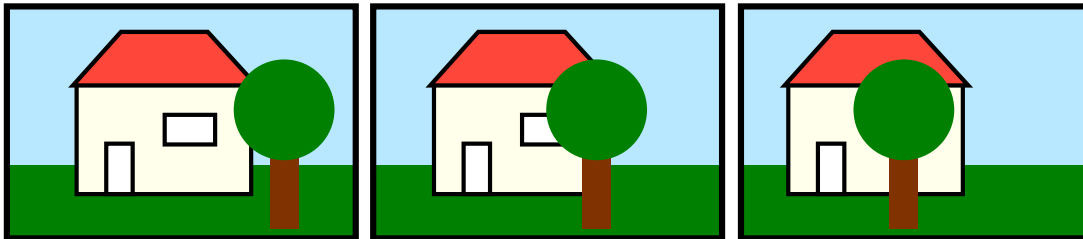


Figure 3.6: The occluding tree is moving from right to left when camera views are ordered from left to right. Hence, partial occlusions often occur only in one direction.

Shen [46] proposed a multiple view 3D reconstruction approach that internally uses PatchMatch Stereo. For each image a corresponding neighboring image is selected based on its viewing angle and baseline. PatchMatch Stereo in scene space is performed on this stereo pair. Hence, homographies are used to map 2D points between the unrectified image views. The method further differs from the original approach by using NCC for cost computation and skipping the view propagation step. In contrast to our approach, only one corresponding image is used for the initial depth map creation. In a subsequent step depth maps from multiple views are fused to a combined 3D representation of the scene, including occlusion handling for inconsistent matches.

Another multi-view approach that incorporates the PatchMatch idea was presented by Zheng et al. [66]. PatchMatch propagation is used for depth estimation from a subset of source image views. The selection of appropriate views for each pixel is performed jointly with the depth estimation by solving a probabilistic graphical model. Instead of performing normal estimation, only a single fixed plane is used for depth estimation. The dominant orientation of the scene is estimated in a pre-processing step. Without normal estimation, processing time can be reduced at the cost of reduced accuracy and biased surface reconstruction towards the estimated dominant orientation.

3.6 Summary

In this chapter we have given an overview of the state-of-the-art work related to our approach. We have briefly reviewed local stereo matching approaches and different cost computation functions. The PatchMatch Stereo approach by Bleyer et al. [8] was described in detail and some adaptations to this approach were stated. Furthermore, we reviewed methods for panoramic as well as high-resolution stereo image pairs. Another major topic of this thesis is multi-view stereo matching. Hence, different approaches in this field related to our work were reviewed in the last section of this chapter.

Multi-View Stereo Approach

In this chapter we present our implemented multi-view stereo matching algorithm. We start with the implementation of the two-view PatchMatch Stereo algorithm by Bleyer et al. [8]. Subsequently, we transform the approach to scene space and extend it to multi-view.

The PatchMatch Stereo algorithm estimates a disparity value and surface normal at each pixel position from a rectified stereo image pair as input data. Implementation details are given in Section 4.1. We furthermore extend the approach with three alternative cost functions, which are presented in Section 4.2. In Section 4.3 the transformation from disparity space to scene space is explained. This step allows us to directly process non-rectified images and estimate depth values instead of disparity values. Operating directly in 3D scene space allows the fusion of multiple input images at once. Hence, we can process multiple views in a single combined setup. This multi-view approach is presented in detail in Section 4.4.

4.1 Two-View Stereo Matching

Our two-view stereo matching approach for rectified image pairs is a re-implementation of the PatchMatch Stereo algorithm by Bleyer et al. [8]. The basic concept of this approach is explained in Section 3.3. In this section we will focus on the technical details of our implementation.

The main idea of the approach is to utilize a randomized approximate correspondence search in order to estimate a 3D plane at each pixel position. The plane can be represented in the Hessian normal form by a normal vector \mathbf{n} and the distance d to the coordinate origin: $\pi = [\mathbf{n}^T d]^T$. For a point on the plane the following equation must hold: $\pi^T [x \ y \ z \ 1]^T = 0$. Hence, the disparity value z at a pixel position (x, y) can be computed from the plane parameters:

$$z = \frac{-n_x x - n_y y - d}{n_z} \quad (4.1)$$

Accordingly, the parameter d of the plane can be computed from a point (x, y, z) and a normal \mathbf{n} with the following formula: $d = -n_x x - n_y y - n_z z$.

The estimation of a whole 3D plane instead of only a disparity value at each pixel position allows the support of slanted surfaces. Respectively, pixels within a support window do not need to all have the same disparity. Only in case of a fronto-parallel plane ($\mathbf{n} = [0, 0, 1]^T$) the disparity value is the same for all pixels. A normal that is not orthogonal to the image view leads to a slanted support window in the second image view and, thus, allows the reconstruction of slanted surfaces.

The PatchMatch Stereo algorithm consists of three basic steps: random initialization, iterative propagation and refinement, and post-processing. Following, the implementation details of these steps are explained.

Random Initialization: In the initialization step a random disparity and unit normal vector is selected for each pixel. The disparity value is selected from a uniform distribution on the defined disparity range $[0, maxDisparity]$. For the random selection of the unit normal we followed the approach of Marsaglia [36] in order to pick a uniformly sampled random point from the surface of a unit sphere. Two independent points x_1, x_2 are picked from uniform distributions on $(-1, 1)$ until $S = x_1^2 + x_2^2 < 1$. The unit vector is then formulated in the following way:

$$\mathbf{n} = (2x_1\sqrt{1-S}, 2x_2\sqrt{1-S}, 1-2S)^T \quad (4.2)$$

In order to limit the normal vector to a hemisphere the dot product between the random unit vector and the view vector is computed. If the result is positive the vector is inverted. In case of rectified images we set the viewing vector to $[0, 0, 1]^T$. For the multi-view setup in Section 4.3 the viewing ray is defined by the location of the 2D point in the image plane and the camera center.

Iterative Propagation and Refinement: As stated in Section 3.3 the iteration step consists of spatial propagation, view propagation, temporal propagation and plane refinement. The temporal propagation step was omitted in our implementation since we are not considering temporal image sequences.

In the *spatial propagation* step a plane from a neighboring pixel is propagated to the current pixel if the cost value is improved. Bleyer et al. consider the left and upper neighbors in even iterations (top-left to bottom-right propagation) and the right and lower neighbors in odd iterations (bottom-right to top-left propagation). We altered this propagation scheme after the second iteration in order to also allow propagation from top-right to bottom-left and reverse.

In order to speed up computation on multiple cores we tested an alternative propagation scheme, which was proposed by Zheng et al. [66]. Instead of propagating from vertical and horizontal neighbors simultaneously, we use four alternating propagation directions: right, down, left, up. This allows us to process separate rows (respectively columns) in parallel on multiple cores. The propagation impact is limited to 1D in this setup. Hence, more iterations might be necessary. Quantitative evaluations indicated slightly worse results on our datasets, with the benefit of faster computation time and potential parallelization on GPU.

In the *view propagation* step we propagate good plane guesses between image views, based on the assumption that corresponding points in the two views have the same surface plane. Hence, for the currently processed view, we search on the epipolar line (= horizontal scanline for rectified images) of the second view for points that have the current pixel as matching point (within 1 pixel tolerance). For each matching point its plane is used for cost computation for

the current pixel. If the cost of the current pixel is improved by one of the planes, the plane information gets updated accordingly.

In the *plane refinement* step we aim to further decrease the matching cost by testing random plane candidates within an iteratively decreasing search window around the current best plane. Two parameters are used to define the search window: $\Delta_{z_0}^{max}$ for the disparity range and Δ_n^{max} for the range of the normal vector components. Hence, a random value within the interval $[-\Delta_{z_0}^{max}, \Delta_{z_0}^{max}]$ is added to the disparity value of the current plane and three random values between $[-\Delta_n^{max}, \Delta_n^{max}]$ are added to each of the three components of the normal vector respectively. The new normal vector is normalized and transformed to the hemisphere as done in the initialization step. Bleyer et al. [8] suggested to start with $\Delta_{z_0}^{max} = maxDisp/2$ and $\Delta_n^{max} = 1$ and divide the values by 2 after each iteration. The iterations stop when $\Delta_{z_0}^{max} < 0.1$. However, we prefer to divide the limit values by a factor of 4 after each iterations since it requires less iteration steps without loss of accuracy on our test set. Our test images cover a larger disparity range than the Middlebury Stereo Vision Benchmark [45], in the original PatchMatch Stereo paper.

Post-processing: A left/right consistency check is performed for each pixel in order to discard occluded or mismatched pixels. For a pixel p in the left view the matching point p' in the right view is computed. If the difference between the two disparities of p and p' is bigger than a given threshold (1 in our experiments), the pixel p is marked as inconsistent. We further added a normal check, suggested by Heise et al. [20]. Pixels are also marked as inconsistent if the deviation between the corresponding normals is more than 5° . If a dense disparity map is desired, the invalidated pixel positions can be filled in a post-processing step. As Bleyer et al. [8] suggested we search for the closest consistent pixel to the left and right and propagate the planes of the two pixels. The smaller of the two disparity values is then used for filling, since occlusions occur in the background. In order to get rid of horizontal streaks, that can occur due to this filling strategy, a weighted median filter is applied on the filled-in disparities [43].

The presented approach is capable of processing color images as well as gray-scale images. Processing of color images is computationally more expensive since three color channels need to be processed. It is unclear whether and how much color information actually improves the matching accuracy compared to single-channel gray-scale information [6, 7, 23, 38]. A reason for the moderate performance improvement with color images compared to gray-scale images is that the three color channels are strongly correlated to each other. Due to these observations, we performed tests mainly on gray-scale images.

Panoramic Images

Among others, we tested our two-view stereo implementation on high-resolution panoramic images obtained from a 360° panorama camera with an underlying cylindrical projection. For the underlying sensor model of the camera we refer to Amiri Parian and Gruen [40]. The non-linear epipolar geometry induced by the cylindrical projection results in generally non-straight epipolar curves. However, we directly work on provided rectified image pairs. Hence, the presented two-view stereo matching approach can be directly applied on the panoramic images. The low memory requirements of the PatchMatch Stereo algorithm allowed us to process the high-resolution images on our computer. Results are discussed in Section 5.1.

4.2 Cost Functions

In this Section we will shortly describe the implemented cost functions that were used in our evaluation. For a review of different cost computation methods see Section 3.2.

PM Cost: We implemented the cost function used by Bleyer et al. [8] as described in Section 3.2. The computation is based on truncated absolute color and gradient differences (see Equation 3.3). For color images we compute the L1 norm of a three dimensional RGB vector and divide the result by 3, while for gray-scale images the absolute difference of the gray-scale values is used. The horizontal and vertical gradient components are computed with a 3x3 Sobel operator [12,49] and the L1 norm divided by 2 is used for the difference computation. Additionally, the adaptive support weight idea is used in order to deal with depth discontinuities within the support window (Equation 3.2).

PM Self Similarity: Bao et al. [3] presented a self-similarity approach for faster cost computation. Instead of evaluating all pixels of a support window only the n most similar pixels to the center pixel are considered. We use the weight w of the adaptive support window (Equation 3.2) as similarity measure. Hence, this value is already precomputed, which speeds up the computation of the cost value. The PatchMatch idea for randomized correspondence search [4] is used for an efficient computation of the n most similar pixels in a window (the self-similarity vector). The assumption behind this idea is that neighboring pixels are similar in appearance and, hence, their self-similarity vectors share a large set of common points. Propagating those common points significantly reduces computation time compared to a brute-force selection of n similar points for each pixel individually. The authors suggest $n = 50$ as a good trade-off between speed and accuracy. A drawback of the method is that storing the similarity locations and weights increases memory consumption.

Census Transform: We implemented the extended Census Transform approach of Stein [50]. Stein introduced an additional parameter ϵ in order to categorize the intensity value of a support pixel into equal to (within ϵ threshold), lesser than or greater than the center pixel (see Equation 3.1). Our default value for the intensity threshold is $\epsilon = 2.5$ (for intensity values in the range of $[0, 255]$). A signature string is created for each support window, consisting of one of the three class labels per pixel position. The cost of a potential match is defined as the Hamming distance between the signature strings of the left and right support window.

Sparse Census Transform: Zinner et al. [67] introduced an adaption of the original Census Transform in order to speed up computation time. Instead of considering all pixels of the support window, only every second horizontal and vertical value is used. We combined this approach with our extended Census Transform implementation.

4.3 Scene Space PatchMatch Stereo

In this section we explain the necessary steps to move the PatchMatch Stereo approach from disparity space to scene space. Hence, instead of working with disparity values and horizontal epipolar scanlines, we operate directly in the 3D scene space, estimating 3D points and depth values. Furthermore, planes are defined in scene space instead of the non-uniform disparity space. This transition to scene space allows us to process non-rectified images. However, the

camera projection matrix \mathcal{P} of each camera is required. This projection matrix was introduced in Section 2.1. It combines intrinsic and extrinsic camera parameters into a single 3x4 matrix of the form $\mathcal{P} = \mathcal{K}[\mathcal{R}|\mathbf{t}]$, with \mathcal{K} representing the intrinsic camera calibration matrix and $(\mathcal{R}, \mathbf{t})$ representing the extrinsic rotation and translation of the camera in scene space.

The basic algorithm works similar to the PatchMatch Stereo approach in disparity space. Per pixel a plane $\pi = (\mathbf{n}^T, d)^T$ in scene space is estimated. In the initialization step a random depth value from the defined depth range and a random unit normal are picked. Special care has to be taken when selecting a random depth value. We do not want it to be uniformly distribution on the depth range, since this would lead to a non-uniform distribution along the epipolar line in the second image view (see Figure 2.5). This problem was described in Section 2.2 when explaining epipolar geometry and the relation between depth and disparity. To overcome this problem, we select a random disparity value instead and convert it to a depth value according to Equation 2.5. To ensure that the unit normal is not facing backwards, we invert the normal if the dot product between the normal and the viewing vector is positive. The viewing ray is passing through the camera center and the current pixel projected on the image plane.

In order to compute the parameter d of the plane π from the randomly assigned depth value ω , we compute the 3D point \mathbf{X} in scene space from the 2D image point x and the assigned depth value ω , utilizing Equation 2.3. Subsequently, the distance d of the plane to the coordinate origin (which is the camera center of the reference camera) can be computed from the definition of a plane in Hessian normal form:

$$d = -\mathbf{n} \cdot \mathbf{X}. \quad (4.3)$$

In Section 2.3 the concept of plane-induced homographies was introduced. A homography \mathcal{H}_π induced by a plane π in scene space defines a mapping from one image plane to another. Hence, we can map an image point \mathbf{x} to point \mathbf{x}' in another camera view with the formula $\mathbf{x}' = \mathcal{H}_\pi \mathbf{x}$. The homography \mathcal{H}_π is defined by the two camera projection matrices \mathcal{P} and \mathcal{P}' and the scene plane π (see Equation 2.6). By utilizing this mapping we are no longer restricted to rectified images. We transform the reference camera to the coordinate origin $([0, 0, 0])$ with zero rotation ($\mathcal{R} = \mathcal{I}$) in order to simplify the homography computation. The second camera is transformed accordingly (see Section 2.3 for details).

In the plane propagation step we propagate the plane parameters \mathbf{n} and d to neighboring pixels. In order to compute the depth value ω at the new position according to the propagated plane, we intersect the viewing ray (through camera center C and position \mathbf{x} on the image plane) with the 3D plane. Therefore we substitute the point \mathbf{X} in the Hessian normal form definition of the plane (Equation 4.3) by the viewing ray (Equation 2.3) and solve for ω :

$$\omega = \frac{\mathbf{n}(\mathcal{M}^{-1}\mathbf{p}_4)^T - d}{\mathbf{n}(\mathcal{M}^{-1}\mathbf{x})^T} \quad (4.4)$$

The major difference of processing in scene space compared to disparity space is the necessity of plane induced homographies in order to determine corresponding pixel locations in the second view. This requires the declaration of camera projection matrices for both views. However, no rectification is necessary and planes are estimated in 3D scene space instead of disparity space. Hence, a plane in the scene can be correctly represented. In disparity space,

an estimated plane would represent a curved surface in the 3D scene due to the non-uniform mapping of disparity values.

4.4 Multi-View PatchMatch Stereo

In the two-view stereo case we have assumed a left and a right camera view and estimated the depth value at each pixel of one view by finding the best match (lowest cost value) along the epipolar line in the second view. In scene space we can easily extend this idea to multiple views. We again start from one camera, the reference camera, and aim to estimate the depth value by finding the matching point with the lowest cost. This time, however, we have n other camera views and hence n matching points (one per view). Thus, for a plane candidate π we compute n cost values from the corresponding n matching points in the other camera views. These n cost values further need to be combined to yield a total cost, e.g. by summing over all partial costs. As a result we obtain a depth map and corresponding plane normals for the selected reference view.

By introducing multiple cameras we aim to be more robust against occlusions, mismatches and noise. E.g. an object in the background might be occluded by some foreground objects in certain views. If multiple views are considered, information from other views can be used to correctly reconstruct the partly occluded area. An example of partly occluded areas is depicted in Figure 4.1. When matching the center view with the left view, parts on the right side of the fountain (white circle) are occluded. If the center view is additionally matched with the right view, these occlusions can be resolved. The same accounts for partly occluded or non-visible areas on the wall and the floor. Furthermore, if we get consistent depth estimations over multiple views we can reduce noise by averaging over the depth and normal estimates.

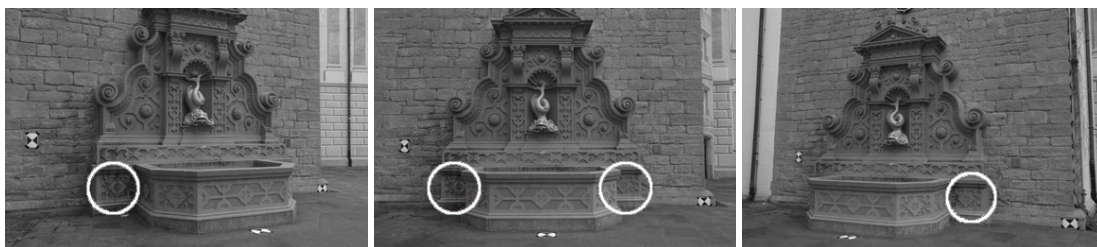


Figure 4.1: Areas circled in white are occluded in either the left or the right camera view. Images taken from the fountain-P11 dataset [51].

In order to transfer image coordinates from one camera view to another we use plane-induced homographies, as described in the previous section. An exemplary setup with one reference view and three other cameras is depicted in Figure 2.7.

View Selection

In order to limit the number of images for matching per reference view we pre-select a subset of image views. A view is excluded if the angle between its principal axis and the principal axis of

the reference view is smaller than 3° or larger than 45° . This is based on the assumptions that image pairs with a smaller viewing angle are too close to each other for reliable depth estimation and pairs with a bigger viewing angle are not matchable any more (see for example Tola et al. [56]). For larger datasets one could further take into consideration the distance between two cameras.

In general, we only want to consider image views with a large visual overlap to the current reference view. View selection allows for faster computation, since the number of images that need to be processed is reduced. Furthermore, wrong matches from far away view points do not contribute to the depth estimation.

Cost Combination

In the multi-view scenario we match a pixel in the reference view with pixel locations in n other image views, where n is the number of remaining views after view selection. Hence, for each pixel and 3D plane in the reference view n cost values are computed. For each image pair the cost value is computed as described in Section 4.2 for the two view case. The n cost values need to be combined in order to get a single cost estimate for a 3D plane π .

A possible approach is to accumulate over all n cost values, as proposed by Okutomi and Kandade [39]. However, if objects are occluded in some of the image views, these views will return a high cost value for the correct plane. In order to robustly handle these occlusions we follow the suggestions of Kang et al. [33]. They propose to consider only the best 50% of all n image matches for the combined cost computation. Instead of a fixed selection of 50% of the considered views, we introduce a parameter K in our implementation that specifies the number of single cost values considered (in ascending order) for the combined cost computation:

$$m_{srt} = \text{sort}_\uparrow(m_1 \dots m_N) \quad , \quad m_{best-K} = \sum_{i=1}^K m_{srt,i} . \quad (4.5)$$

We refer to this cost combination method as *best-K*. The choice of the parameter K depends on different factors. A higher value is desirable for more robust results. However, if a 3D point is only visible in a few views, high cost values from mismatches might negatively affect the result.

Therefore, we introduce a further alternative cost combination method which is independent of the number of non-occluded views. Again, a cost value is computed for each image view and the values are sorted in ascending order. The lowest cost values m_l of the n matches is used to compute a truncation threshold $t_l = km_l$ for the remaining views. In our experiments $k = 1.8$ performed best. We accumulate over all n truncated cost values with the following equation:

$$m_{trunc} = \sum_{i=1}^n \min(m_i, t_l) \quad (4.6)$$

The combined multi-view cost m_{trunc} is computed from the n single view cost values m_i , truncated by the threshold t_l . Hence, all good matches contribute to the results and mismatches do not negatively influence the result. We refer to this variant as *trunc(ated)*.

Depth Fusion

We first compute a depth map for each view by consecutively treating all n views as reference views. Then depth fusion is performed on these n depth maps in order to eliminate wrong depth estimations and reduce noise by averaging over consistent depth estimates. Furthermore, consistent depth values can be propagated to other views where depth values are missing.

Our fusion approach can be seen as an extension of the post-processing step in the original Patchmatch Stereo algorithm, consisting of left/right consistency checking and filling. Additionally, we introduce an update step, where consistent depth estimates from other views are propagated to the current view. Hence, we perform depth map fusion in three main steps: consistency checking, update and filling. Intermediate results before and after these steps are depicted in Figure 4.2 for image 7 of the *fountain-P11* dataset [51]. A similar multi-view depth map fusion approach was proposed by Pollefeys et al. [41]. However, we do not compute a confidence measure for a single depth estimate since that would require the full cost volume over different disparities.

- **Consistency Check:** Mismatches mainly occur in textureless regions or due to occlusions (including also non-overlapping regions due to different camera viewpoints). In order to eliminate those mismatches we only consider plane estimates that are consistent with other views. Hence, for each depth map we compute the corresponding 3D points in scene space per pixel location and map them to the $n - 1$ other views, resulting in a 2D pixel location p_i and a depth value z_i per view.

For a consistent match, the mapped depth value z_i needs to be equal to the depth value stored at the pixel location p_i up to a given threshold ϵ . The threshold ϵ for consistency is depending on the dataset, i.e. different datasets require different depth ranges and provide different depth accuracy. For the Strecha dataset we use $\epsilon = 5\text{cm}$. Additionally normal vectors are considered as an extra cue for consistency check, eliminating points with a normal difference of more than 30° .

If the computed depth value from the 3D point of the reference view is coinciding with at least one other view, we mark the pixel in the reference view as consistent. Furthermore, depth values and normals are averaged over all consistent views to suppress noise. Normals can be averaged directly in scene space. To average depths the 3D points of all consistent views are projected onto the viewing ray of the reference camera.

- **Update:** After consistency checking, we fill inconsistent points with consistent estimates from other views. Therefore, for each view an update map is created that stores consistent plane estimates from other views. If the plane estimate of one view is consistent with at least two other views, its plane parameters (depth and normal) are transferred to all other views and stored in their respective update maps. In order to handle occlusions, for each pixel location the closest depth value is stored in the update map. For example, a point in the background can be consistently visible in some views, while being occluded by a foreground object in the current view.
- **Filling:** Remaining inconsistent points after the consistency check and update step are filled in a similar vein as proposed by Bleyer et al. [8]. We search for closest consistent depth and normal values in (positive and negative) horizontal and vertical direction.

Plane parameters from these four resulting points are used to compute a depth value at the current pixel position. Bleyer et al. used the farther away depth value for filling. As an alternative approach we chose the plane of the consistent neighbor whose color value is most similar to the filled pixel. Thus, we want to propagate the plane of the most similar pixel, assuming that it belongs to the same object. To eliminate possible horizontal and vertical streaking effects we apply a weighted median filter on the depth map with a window size of 7×7 .

4.5 Summary

In this chapter we have presented our implementation of dense multi-view stereo matching in scene space. We have started with the two-view approach PatchMatch Stereo by Bleyer et al. [8]. It requires a rectified image pair as input data and estimates a disparity and normal at each pixel location. The algorithm uses an iterative propagation scheme in order to refine randomly initialized disparity and normal values. We have moved the original approach from disparity space to scene space by utilizing plane induced homographies. As a consequence, rectification is no longer necessary and, furthermore, the approach can be directly extended to multi-view.

This multi-view approach was introduced in the last section of this chapter. The cost function was extended to handle multiple views. In the fusion stage, consistency checking, update and filling are used in order to eliminate inconsistent depth values, propagate planes from neighboring pixels and views and denoise results.

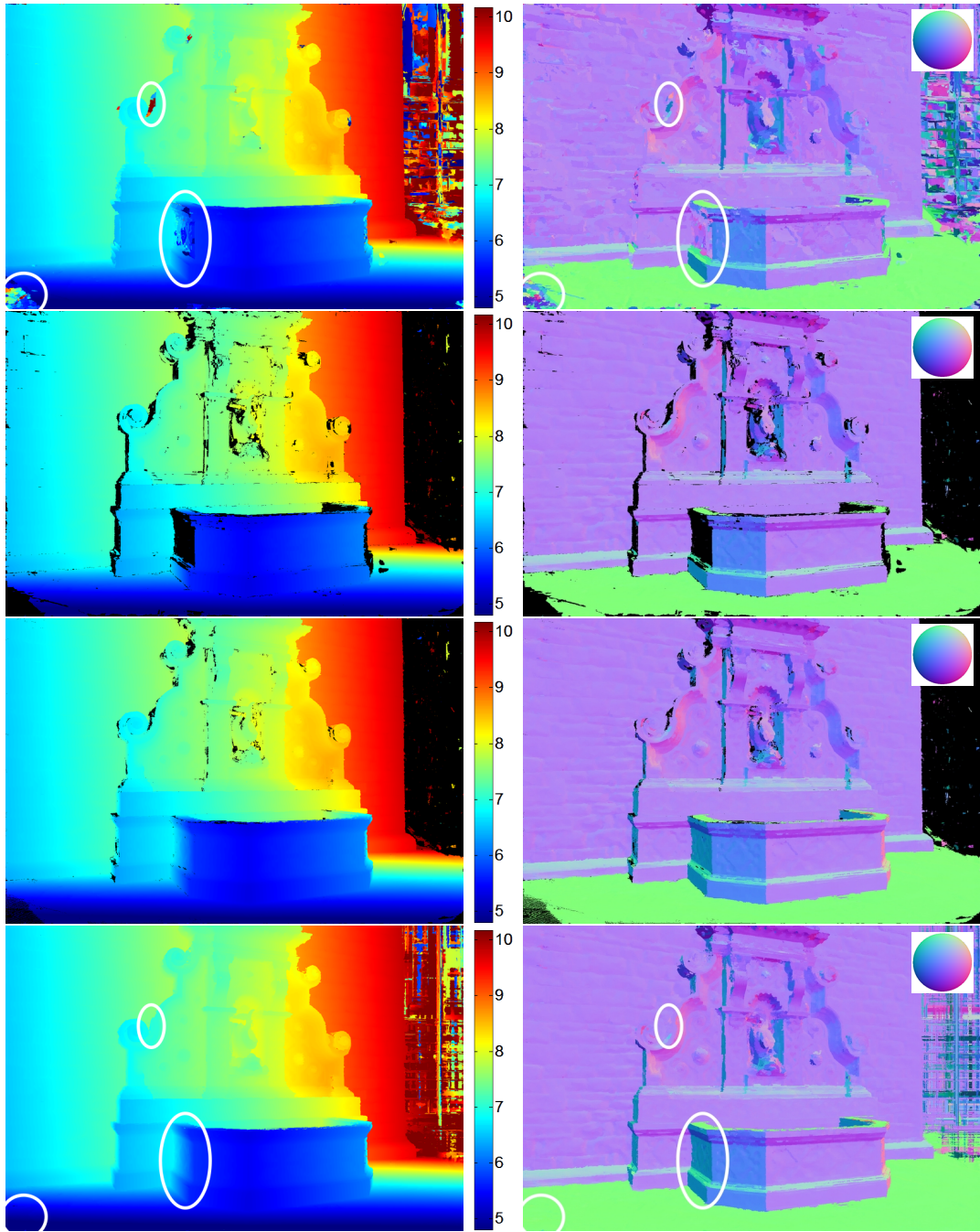


Figure 4.2: Depth map fusion for image 7 of the fountain-P11 dataset [51]. Left column: intermediate results of the depth maps before and after each stage. Right column: corresponding normal maps. Top to bottom: raw depth maps before consistency checking, results after consistency checking, update and filling. Note for example improvements on the left side part of the fountain and the ground floor (marked with white circles).

Results

We have conducted a detailed performance analysis of our approach on different two-view and multi-view datasets. Results on rectified image pairs with our two-view stereo matching implementation are presented in Section 5.1. We quantitatively evaluate our algorithm on the Kitti Vision Benchmark Suite provided by Geiger et al. [17]. The dataset includes a large set of stereo pairs of urban outdoor scenes. Subsequently, qualitative results of our algorithm on high-resolution panoramic image pairs are presented.

To demonstrate the performance of our algorithm on non-rectified multiple view data, we evaluated our multi-view stereo matcher on different outdoor datasets. Results are presented in Section 5.2. We state quantitative results on the multi-view stereo dataset of Strecha et al. [51] and on a multi-view setup of the Kitti dataset. Furthermore, we show qualitative results of our algorithm on a sequences of aerial image data.

Tests were performed for different cost functions and window sizes. Our testing machine is equipped with a 3.2 GHz processor and 24 GB of RAM. Runtime results should mainly be seen as relative indicators between different cost functions and window sizes, since the code is not optimized and executed on the CPU only.

5.1 Two-view Stereo

Results for stereo matching on rectified image pairs were obtained from our two-view stereo matching implementation in disparity space, as described in Section 4.1. For error measurements on the Kitti dataset we use dense disparity maps after occlusion filling and weighted median filtering. Additionally, we measure the error of the subset of consistent pixels and the density of this subset. Visual results are mainly shown after consistency checking, without filling. Hence, only consistent pixels of the two views remain in the disparity map.

Unless otherwise stated, tests were performed on gray-scale images only. We used the parameter values proposed by Bleyer et al. [8]: $\{\gamma, \alpha, \tau_{col}, \tau_{grad}\} = \{10, 0.9, 10, 2\}$. Three iterations of plane propagation and refinement were conducted per image pair. However, in our

tests results do not change significantly between iteration 2 and 3. Hence, in order to reduce computation time one could omit the last iteration.

Kitti Vision Benchmark Suite

The Kitti Vision Benchmark Suite [17] consists of 194 training and 195 test image pairs (rectified) with a resolution of about 1240x376 pixels (slightly varying due to different camera calibrations). For our evaluation we only consider the training set, since ground truth data for the test set is not public. The dataset consists of images of outdoor scenes captured from a moving car. Images include rural areas as well as highways around the city of Karlsruhe. Typical objects in the images include streets, houses, vegetation, sky, cars, pedestrians, street signs and street lamps. Hence, examples of challenging areas for the stereo matcher are cluttered vegetation, reflections on cars and other shiny objects, homogeneous areas such as house walls and thin structures such as poles. Furthermore, strongly varying lighting conditions and a large disparity range are challenging properties of the dataset for stereo matching.

For all training images, ground truth data from a Velodyne laser scanner is provided. The ground truth disparity maps have an average density of about 50%. Ground truth disparity maps are provided for non-occluded areas as well as for all ground truth pixels (including occluded areas). Occluded areas are regions in the image that are not visible in one of the two views due to occlusions (see Figure 1.1). These areas can not be resolved with matching and, hence, require additional filling approaches. An exemplary image pair and the corresponding non occluded ground truth disparity map for the left view are shown in Figure 5.1.



Figure 5.1: Left and right stereo image pair 110 of the Kitti Vision Benchmark dataset [17], together with the color-coded ground truth disparity map for the left image (yellow: close, blue: far away, black: no ground truth data available).

The algorithm was evaluated on the provided training dataset of the Kitti Vision Benchmark Suite [17]. Results are summarized in Table 5.1 for different cost functions on window sizes 35x35 and 25x25. As suggested by the Benchmark Suite an error threshold of 3 pixels (disparity) is used. Hence, the table lists the percentage of pixels whose disparity values differ from the ground truth by more than 3 pixels. Error rates are computed for non-occluded areas (noc) as well as for all areas with ground truth data (occ). We evaluate disparity maps after post-processing (consistency checking, filling and weighted median filtering). Additionally, we labeled all pixels that passed the consistency check as *consistent pixels* and evaluate them separately. The percentage of consistent pixels and their error rates are listed in Table 5.1 as well.

Cost	Window size	Error (noc)	Error (all)	Consistent pixel	Error (cons)	Runtime/image
(1) PM Cost	35x35	10.5	11.6	57.2	2.0	14.5 min
(2) PM SS (n=50)		11.4	12.4	46.2	1.6	3.4 min
(3) PM SS (n=100)		10.9	11.9	49.1	1.6	5.5 min
(4) Census Transform		9.3	10.9	48.9	2.5	5.9 min
(5) Sparse CT		9.8	11.5	46.1	2.6	3.2 min
(1) PM Cost	25x25	11.5	12.7	52.5	1.8	5.4 min
(2) PM SS (n=25)		14.1	15.2	37.7	1.6	2.9 min
(3) PM SS (n=50)		12.8	13.9	42.1	1.5	3.2 min
(4) Census Transform		9.9	11.6	44.9	2.3	4.3 min
(5) Sparse CT		10.7	12.4	41.8	2.5	3.1 min

Table 5.1: Performance (% of disparities with error > 3 pixel) of different cost functions for the training set of the Kitti Vision Benchmark Suite [17] with window sizes 35x35 and 25x25. Best results per window size are written in bold letters.

For visualization, image 110 of the Kitti training set was chosen. It includes some of the previously mentioned challenging areas, e.g. homogeneous house walls, thin poles and reflective cars. In Table 5.2 disparity maps after left/right consistency checking are listed for all evaluated cost functions with window size 35x35. Detailed visual results for the cost functions PM Cost and Census Transform are depicted in Table 5.3 and Table 5.4 respectively. Disparity maps after left/right consistency checking as well as after post-processing (occlusion filling and weighted median filtering) are depicted. Additionally, error maps for the post-processed disparity maps are shown.

When analyzing the results in Table 5.1 one can see that the original PatchMatch cost function gives the highest density of consistent pixels. This is for example visible in the dense reconstruction of the street areas in Table 5.3. However, the Census Transform cost functions detect valid matches even in challenging areas, such as the left and right house walls in Table 5.4. This leads to a lower overall error for Census Transform compared to PM cost after occlusion filling and weighted median filtering, since valid plane estimates in these areas are missing for the PM cost function. Sparse Census Transform and PM Self-similarity are the fastest of the tested approaches. As expected, the self-similarity trick significantly speeds up computation time at

the cost of higher errors and sparser disparity maps. Comparing the two window sizes, 35x35 obtains consistently better results for all cost functions at the drawback of a longer runtime.

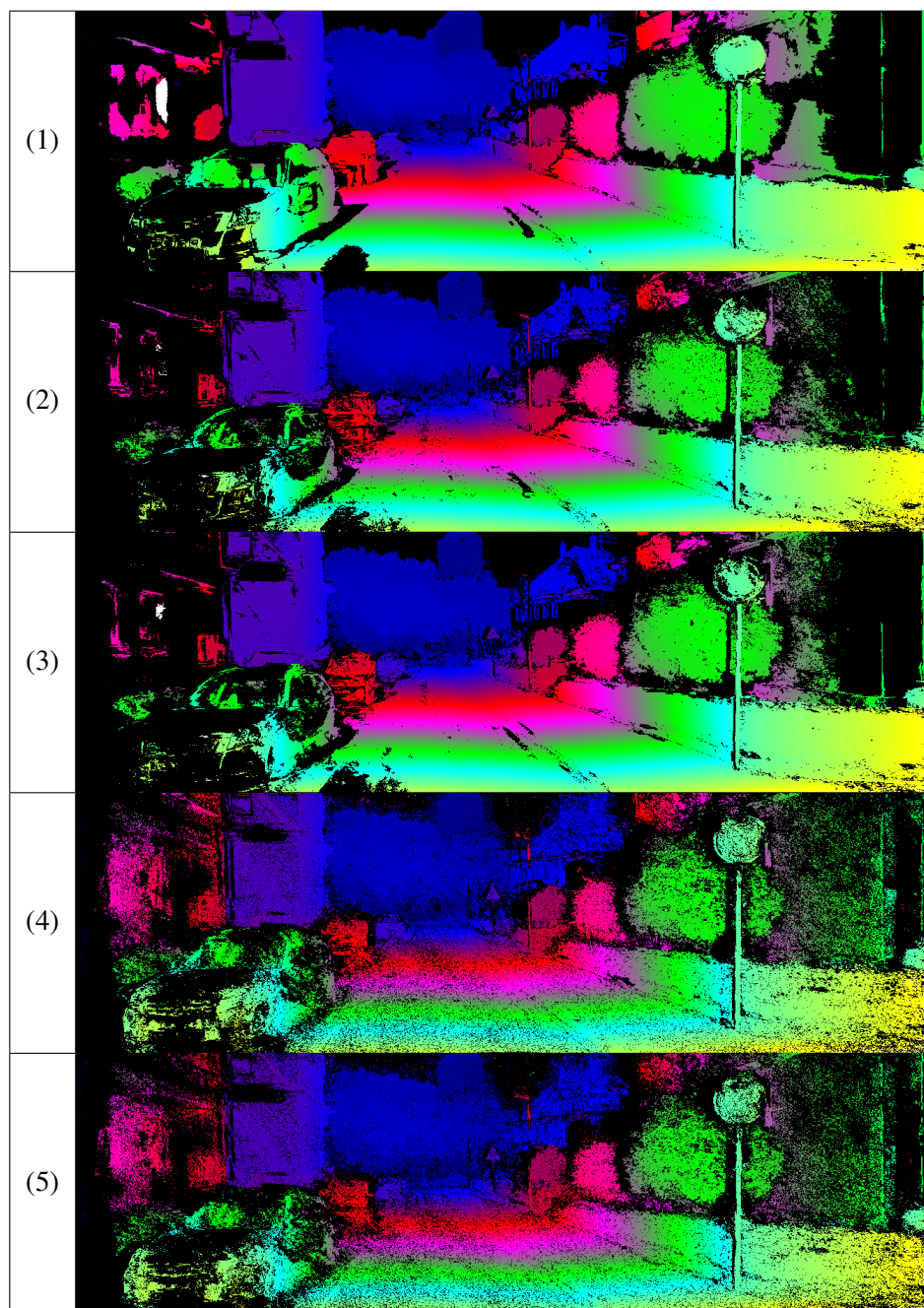


Table 5.2: Disparity maps for Kitti image 110 after consistency checking for different cost functions with window size 35x35: (1) PM Cost, (2) PM Cost with self-similarity (n=50), (3) PM Cost with self-similarity (n=100), (4) Census transform, (5) Sparse Census transform.


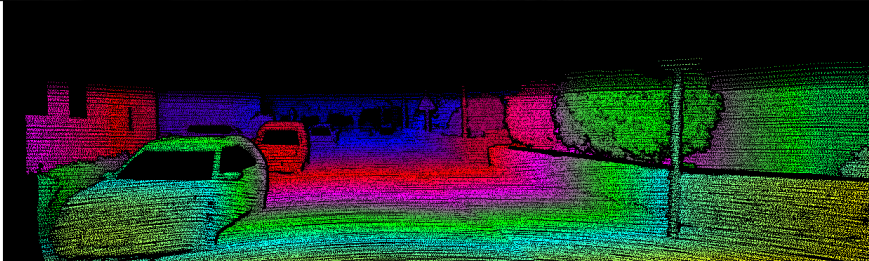
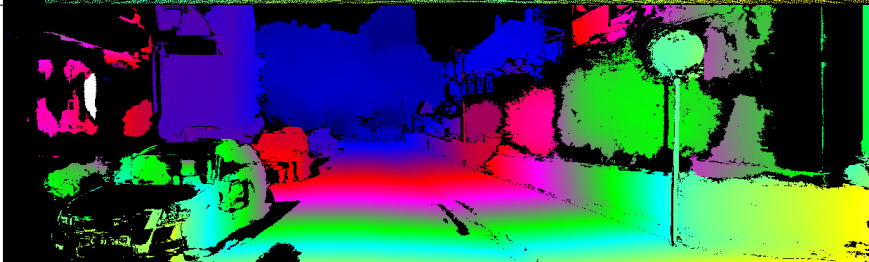
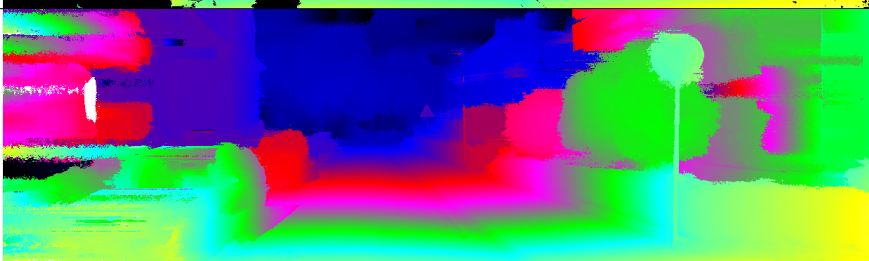
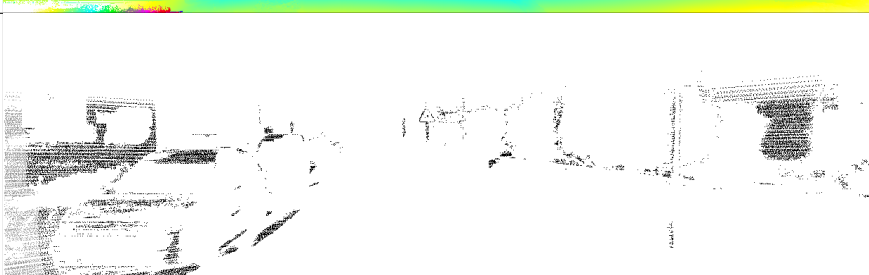
Left Image	
Ground Truth	
Consistent	
Post-Processing	
Error	

Table 5.3: Detailed results for Kitti image 110 with original PatchMatch cost and window-size 35x35: Left image, Ground-truth disparity map, disparity map after consistency check, disparity map after post-processing and error map (white: no error, black: noc error, gray: occ error).


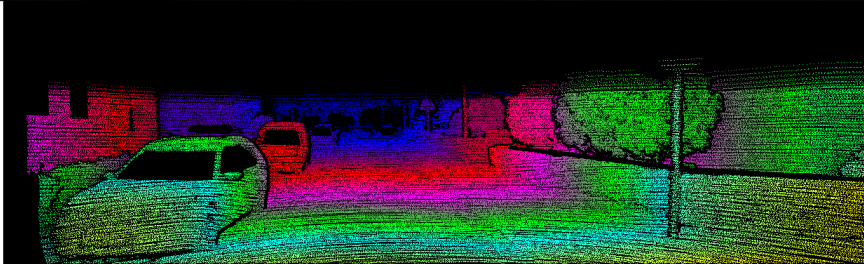
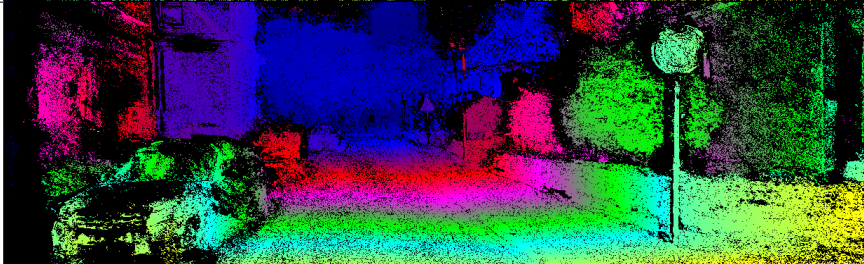
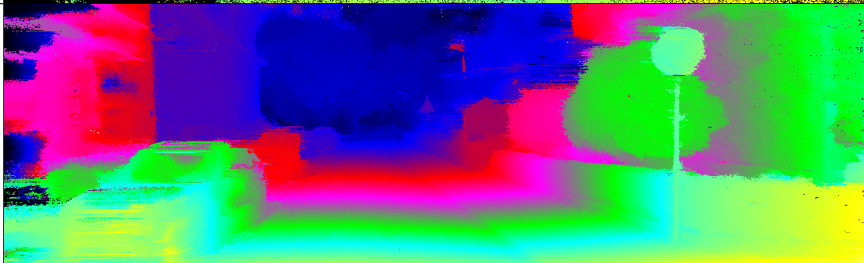
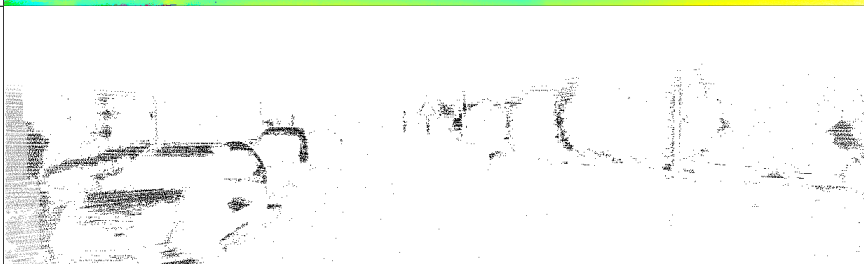
Left Image	
Ground Truth	
Consistent	
Post-Processing	
Error	

Table 5.4: Detailed results for Kitti image 110 with Census Transform cost and window-size 35x35: Left image, Ground-truth disparity map, disparity map after consistency check, disparity map after post-processing and error map (white: no error, black: noc error, gray: occ error).

Panoramic Images

The panoramic images for our evaluation were captured from a 360° panorama camera with an underlying cylindrical projection¹. The raw images have a resolution of 36.206x7.500 pixel (~270 megapixel). Additionally, rectified images are provided. For our evaluation we distinguish between horizontally and vertically displaced stereo pairs. For horizontally displaced panoramic image pairs (taken from roughly the same height) the epipoles are located within the images (along the baseline of the two cameras) [34]. After rectification, heavy distortions occur in the area of epipoles, impairing matching and hence disparity estimation in those areas. Vertically displacing the two cameras (on top of each other) avoids this problem. However, the baseline between two cameras is usually limited in a vertical setup. We show qualitative results for both setups.

Vertical Image Pairs

In this section we state results of our two view stereo matching approach on vertically displaced cameras. Hence, image pairs are referred to as top and bottom image instead of left and right. A rectified image pair of two vertically displaced cameras is depicted in Figure 5.2. Note that for processing images are actually rotated by 90° so that epipolar lines are horizontal.



(a) top image



(b) bottom image

Figure 5.2: Rectified panoramic image pair from two vertically displaced cameras: (a) top image captured from higher camera position and (b) bottom image captured from lower camera position.

¹<http://www.fovex.com/>

We ran our stereo matching algorithm on full resolution (9.103x36.206 for rectified images) as well as on downsampled versions of resolution 1.138x4.526 (12.5% of width and height). The downsampled versions require a smaller window size while giving visually similar results (see Figure 5.3). In order to reduce computation time, comparisons for different cost functions and window sizes were performed on these downsampled versions.

In Figure 5.4 we compare results for PM Cost, PM SelfSimilarity, Census Transform and Sparse Census Transform on the image pair of Figure 5.2. The disparity maps after consistency checking are displayed for the four cost functions with window size 35x35. The Census Transform cost functions estimate consistent disparity values at the floor where PM Cost functions fail. On the other hand, a higher amount of wrong estimates can be found in sky regions. In Figure 5.5 we compare results for the PM Cost function for different window sizes and for color vs. greyscale input images. Bigger window size and color information increase the density of consistent pixels at the cost of longer runtime. Runtime results for different resolutions, window sizes and cost functions are stated in Table 5.5. Considering also the initialization time of PM Self-Similarity, Sparse Census Transform is the fastest of the tested cost functions. Reasonable reconstruction results can be achieved on building facades and ground floor areas for all tested variants. Inconsistencies occur mainly in sky and vegetation areas.

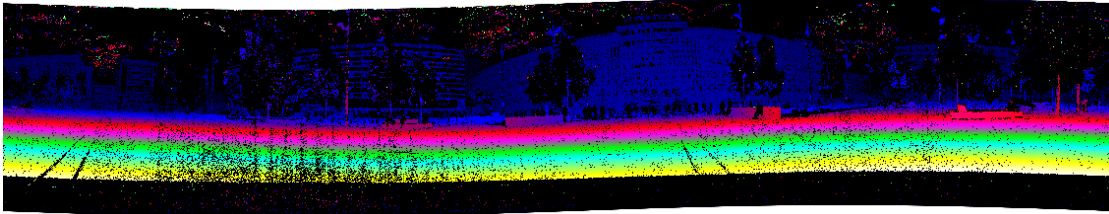
Results for two further vertical image pairs are shown in Figure 5.6 and 5.7. Disparity estimation on downsampled images was performed for PM Cost and Census Transform with a window size of 35x35.

Image size	Cost	Window size	Runtime per Iteration
full	Sparse CT	45x45	1700 min
12.5%	PM Cost	25x25	33 min
		35x35	49 min
		45x45	79 min
	PM SS (n=100)	35x35	16 (+8) min
	Census Transform	25x25	22 min
		35x35	32 min
		45x45	44 min
Sparse CT	35x35	19 min	

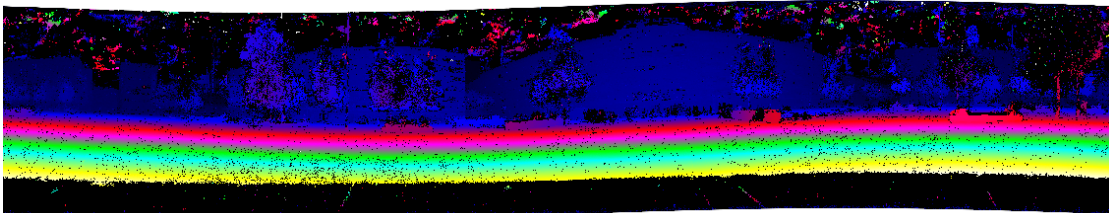
Table 5.5: Runtime results per iteration for panoramic images. Note that for the PM SS cost self-similarity propagation has to be performed during initialization. Runtime for this initialization step needs to be added proportionally to the runtime per iteration (in case of 3 iterations: divided by 3). This proportion is stated in parentheses.



(a) original image (bottom)



(b) full resolution

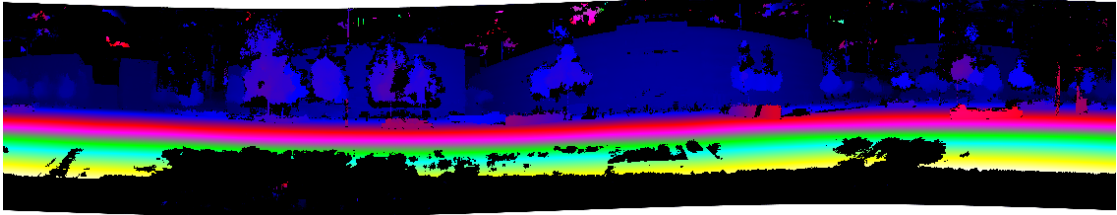


(c) 12.5% resolution

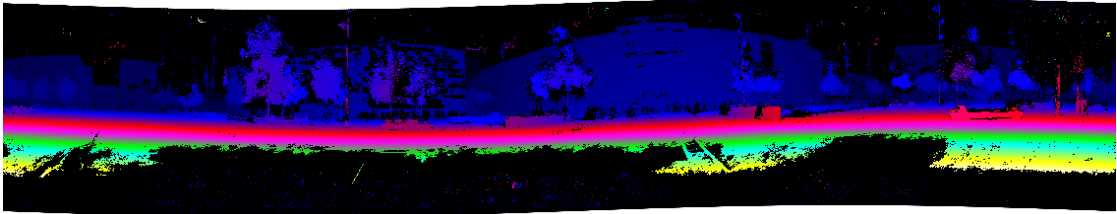
Figure 5.3: Sparse census transform on full resolution with window size 45x45 vs. downsampled version with window size 35x35.



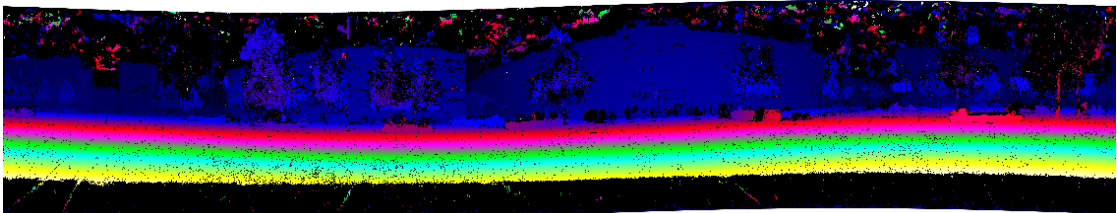
(a) original image (bottom)



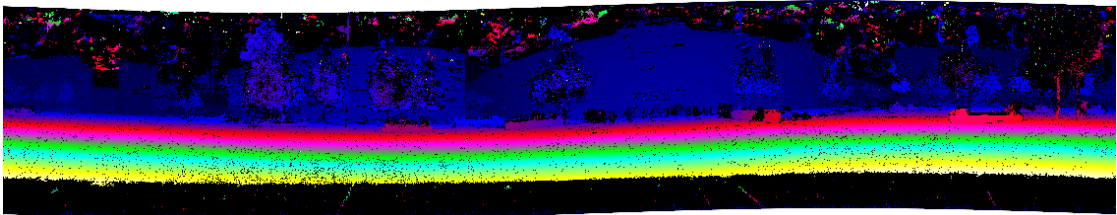
(b) PatchMatch Cost



(c) PatchMatch SelfSimilarity



(d) Census Transform

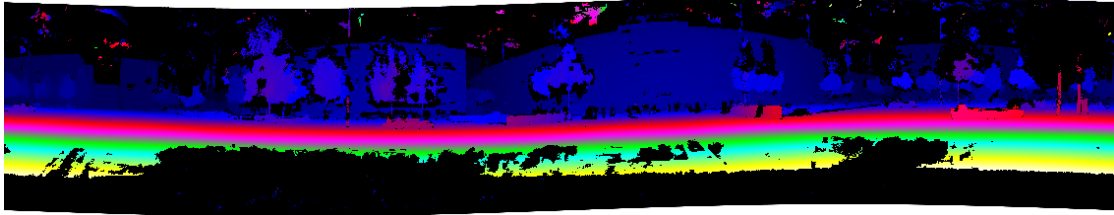


(e) Sparse Census Transform

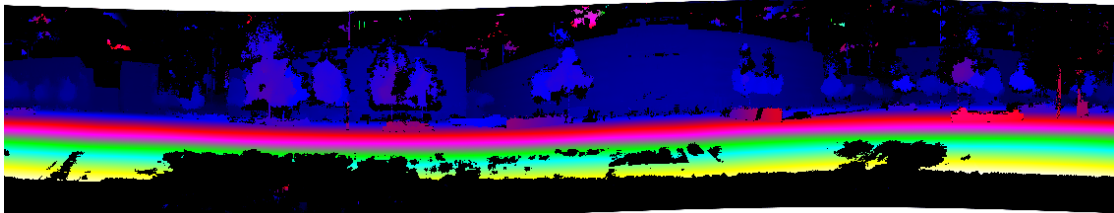
Figure 5.4: Comparison of different cost functions for window size 35×35 . Results are shown for the bottom image (a), hence areas at the lower part of the image cannot be matched.



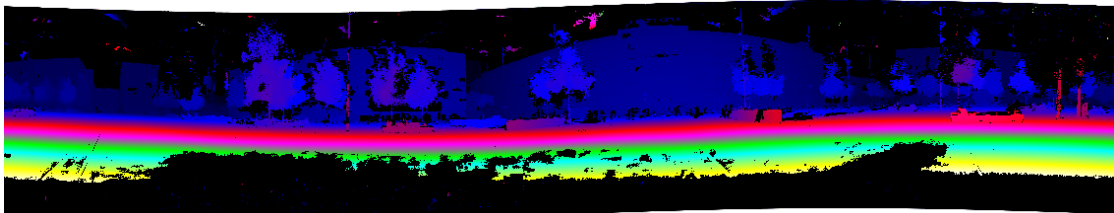
(a) original image (bottom)



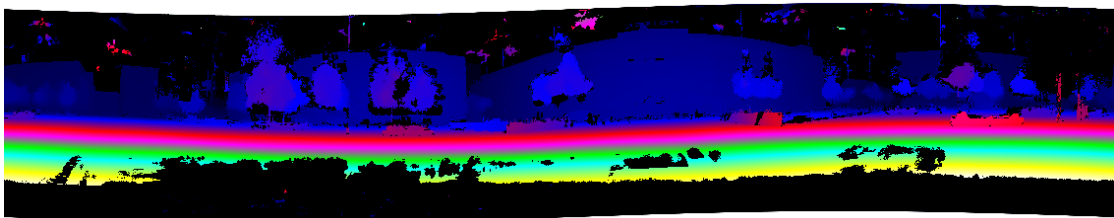
(b) 25x25



(c) 35x35



(d) 35x35 color



(e) 45x45

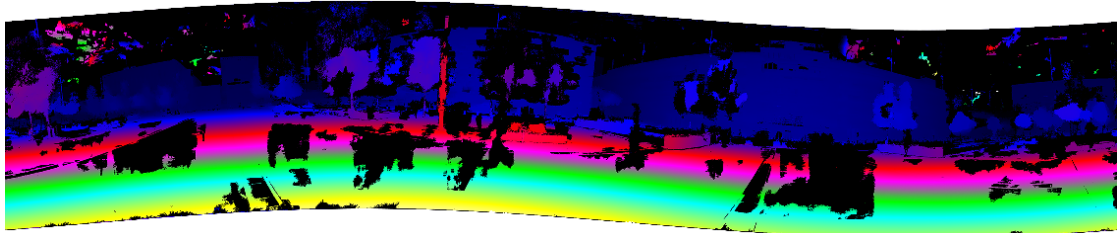
Figure 5.5: Comparison of different window sizes for PM Cost computation: (b) 25x25, (c) 35x35, (d) 35x35 color, (e) 45x45.



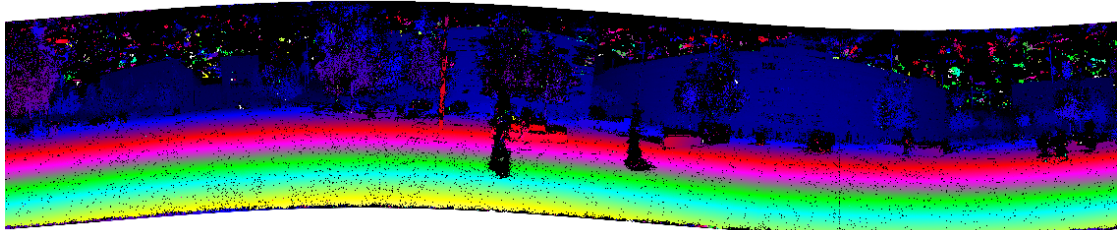
(a) top image



(b) bottom image



(c) PM Cost



(d) Census Transform

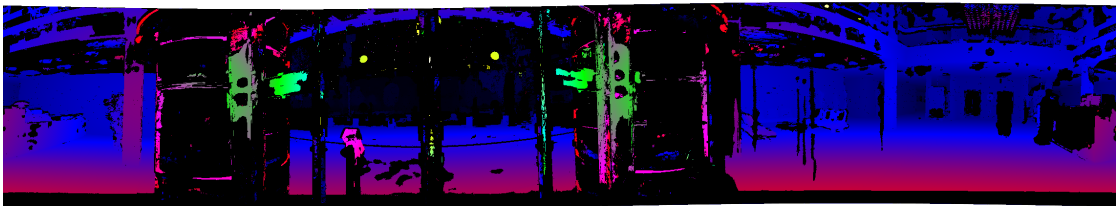
Figure 5.6: Another outdoor image pair (a-b) and its disparity map for the top image with (c) PM Cost and (d) Census Transform and a window size of 35x35 on the downscaled version.



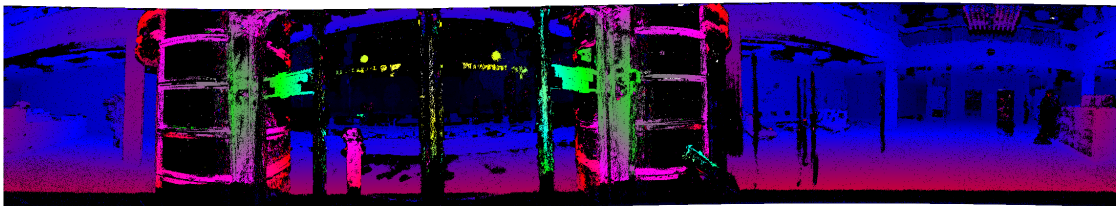
(a) top image



(b) bottom image



(c) PM Cost



(d) Census Transform

Figure 5.7: Indoor image pair (a-b) and disparity maps for top image with (c) PM Cost and (d) Census Transform and a window size of 35x35 on fourth size resolution.

Horizontal Image Pairs

A horizontally displaced, rectified image pair is displayed in Figure 5.8. The original (non-rectified) images are shown in Figure 1.2. Due to the mapping of epipolar lines to horizontal scanlines (rectification), heavy distortions occur in the area of epipoles (in the center and on the left/right sides of the original image or on the left/right sides of the rectified images) which impair a dense correspondence search.

We focus on the reconstruction of feasible areas, i.e. the center part of the images. Therefore, we manually created cutouts of the rectified images and used those cutouts for stereo matching. The upper and lower center part can be combined to a single cutout. Exemplary cutouts and their disparity maps after consistency checking for the Census Transform Cost function are shown in Figure 5.9. Distant objects can be reconstructed up to a certain degree of distortion. Less consistent matches could be detected in the far left and right areas of the images where distortion is strongest. For closer regions, such as the ground floor, distortions increase (see for example the benches in Figure 5.8). Thus, correspondence search is difficult in these areas.



Figure 5.8: Rectified images for input images of Figure 1.2.

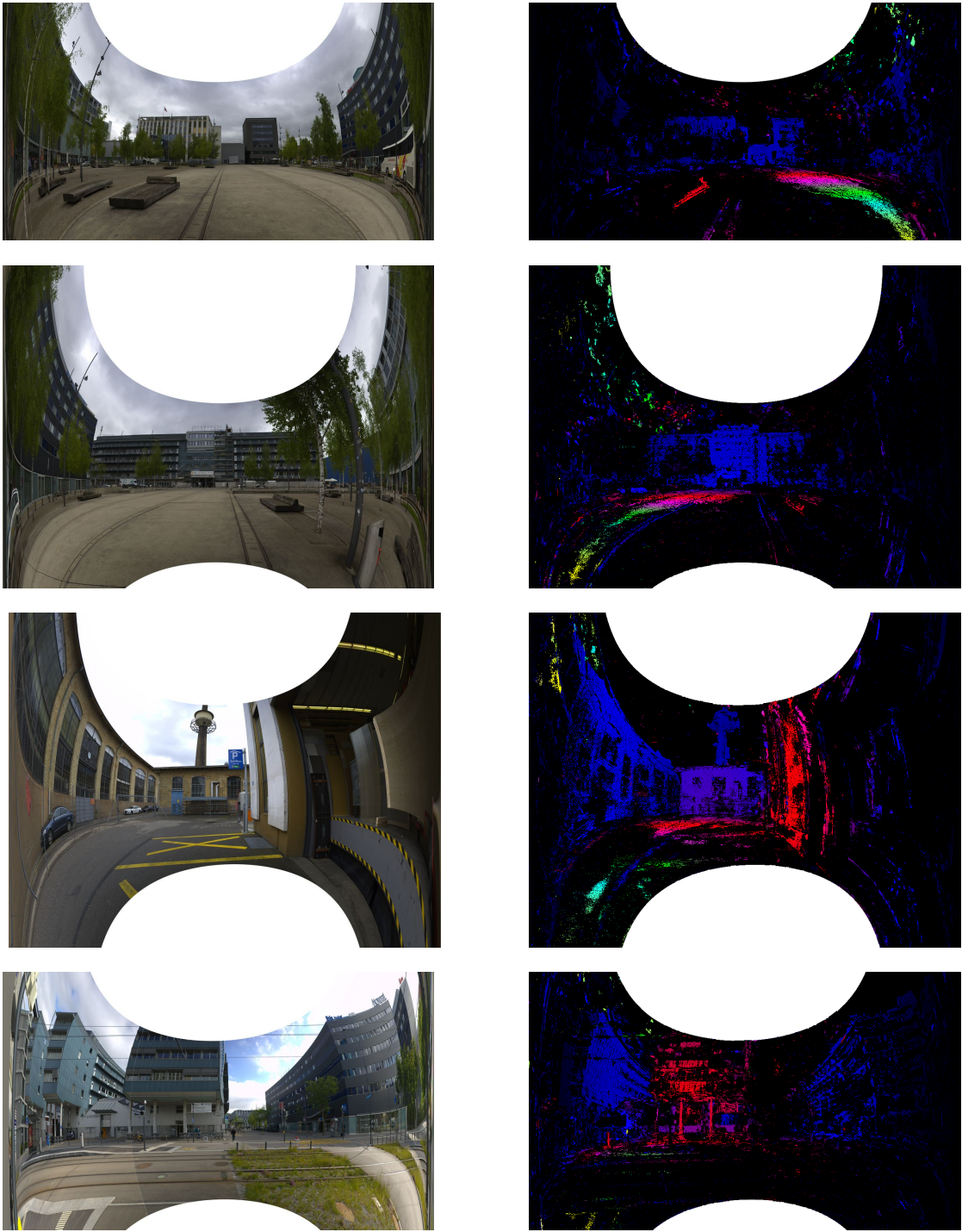


Figure 5.9: Left views of exemplary cutouts and their disparity maps.

5.2 Multi-view Stereo

For a quantitative evaluation of our multi-view stereo matcher we use the dataset of Strecha et al. [51]. We perform a detailed parameter analysis and compare our best results with those of several state-of-the-art multi-view methods. Furthermore, we test our algorithm on a multi-view setup of the Kitti Vision Benchmark Suite [17] and compare the performance of our method with results obtained from our two-view implementation (see Section 5.1). We also show qualitative results of our approach on aerial image data from the city of Enschede.

Strecha Dataset

The Strecha dataset consists of six different image sequences of outdoor scenes with provided camera parameters and bounding volumes of the scenes. Image sequences comprise 8 to 30 images, each having a resolution of 3072x2048 pixels. For two of the sequences (*fountain-P11* and *Herz-Jesu-P8*) ground truth data is provided in form of high-resolution triangle meshes. The meshes were generated from raw laser scanning data (LIDAR). For error measurements we use the ground truth depth maps provided by Tola et al. [55] which were generated from the triangle meshes.

We followed the evaluation concept of Hu and Mordohai [26] in order to compare our final results with their stated results. For each image sequence the percentage of correct depth values for the thresholds 10 cm and 2 cm is averaged over all depth maps except for the two extreme views (first and last image of the sequence).

Parameter Analysis

We show a detailed parameter analysis of our approach for the image sequence *fountain-P11* in Table 5.6. Different image resolutions, window sizes, cost functions and cost combinations were tested. For all variants we used the PM Cost functions with $\gamma = 13$ for the adaptive weight. The listed error results were computed from raw depth maps resulting from our multi-view stereo approach before fusion. On half resolution images (1536x1024) we compared results for three different values of K (1,2,3) for the cost combination variant *best-K*. Furthermore, we computed error results when considering all views with and without view selection. A visual comparison between these variants is shown in Figure 5.10. Additionally, results for cost combination variant *trunc* are shown. Results show that view selection does not only reduce computation time but also significantly improves quantitative results. This can be explained by the elimination of potential wrong matches from farther away views due to little visual overlap or changes in appearance (because of changing viewing angles). The two different cost aggregation variants *best-K* and *trunc* obtain similar results on the tested dataset. For *best-K* the value $K = 2$ obtains best results on an error threshold of 2 cm. Hence, we use this value for further evaluations.

The gray-scale variant of the PM cost function performs slightly worse than its color pendant with the strong benefit of computation time reduction. Hence, if runtime is relevant, one might consider to process only gray-scale images. However, for our quantitative comparison with state-of-the-art methods we use color images. Also the Census Transform cost function obtains inferior results compared to the PM cost function. Concerning window size, results with cost

combination *trunc* give similar results on the sizes 15x15, 19x19 and 25x25 for half-resolution images. We will consider window size 19x19 for our comparison with state-of-the-art methods.

Resolution	Cost	Color	Combination	Window	fountain-P11	
					10 cm	2 cm
half	PM	yes	all (no view selection)	19x19	0.859	0.716
			all	19x19	0.931	0.747
			best-K (1)	19x19	0.937	0.741
			best-K (2)	19x19	0.939	0.749
			best-K (3)	19x19	0.940	0.747
	no	best-K (2)	19x19	0.935	0.744	
	CT	no	best-K (2)	19x19	0.935	0.714
half	PM	yes	trunc	15x15	0.938	0.751
				19x19	0.939	0.751
				25x25	0.940	0.746
		no	trunc	19x19	0.936	0.745
full	PM	yes	best-K (2)	29x29	0.924	0.759
				35x35	0.923	0.757
			trunc	29x29	0.930	0.769

Table 5.6: Comparison on *fountain-P11* of different parameters before depth map fusion on half and full resolution images. The percentage of correct pixels for error thresholds 10 cm and 2 cm is stated for all tested variants.

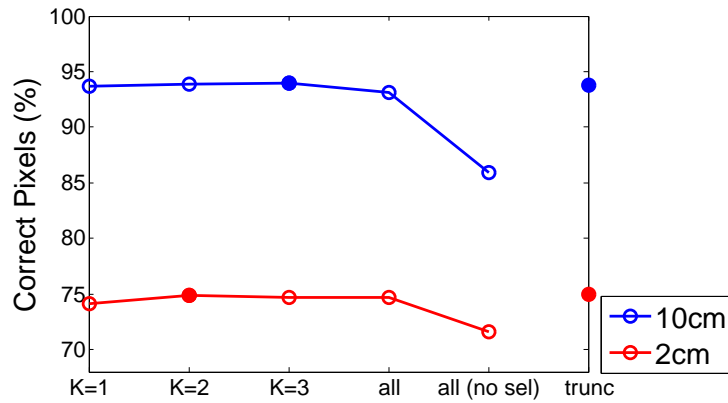


Figure 5.10: Percentage of correct pixels for cost combination approach *best-K* on half resolution image sequence of *fountain-P11* for different values of K : 1, 2, 3, all, all (without view selection) and results for cost combination approach *trunc*.

Since tests were performed on different machines (utilizing a computing cluster) no consistent runtime results can be stated. However, we state runtime results for our parallel implemen-

tation (alternating spatial propagation in four different directions: left, down, right, up). On our 12-core machine one pass with 6 views on half resolution with window size 19x19 takes about 3 minutes for grayscale images and 12 minutes for color images. For our parameter analysis we used the original propagation scheme with diagonal directions. This does not allow for parallel execution of separate rows and columns but gives slightly better quantitative results on our testset.

In Figure 5.11 we show the convergence of the stereo algorithm after each iteration for the parallel implementation variant for error thresholds of 2 and 10 cm, for one specific view. Iteration 0 means correct pixels in the random initialization. Each iteration corresponds to a horizontal or vertical propagation pass plus plane refinement. Furthermore, final error results after depth map fusion are shown. As can be seen, depth estimates are already reasonable after one single propagation step in horizontal directions. Performance improves only marginally after the second iteration. For most practical applications, four axis aligned passes or two diagonal passes will obtain sufficient results. For our quantitative evaluation, we used three diagonal passes, as proposed by Bleyer et al. [8], since quantitative results are slightly improved in the last iteration step.

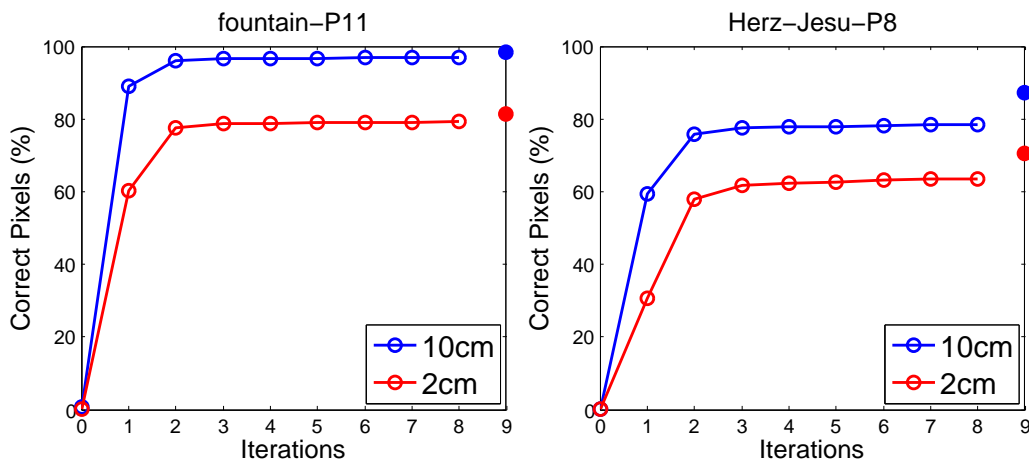


Figure 5.11: Convergence of the Patchmatch procedure. The graphs show the percentage of correct pixels (errors < 2 cm and < 10 cm) for two exemplary views. Each iteration on the horizontal axis corresponds to one pass of axis-aligned propagation in either horizontal or vertical direction. Solid dots denote the final result after fusing multiple depth maps.

Comparison with State-of-the-Art Methods

We compare results obtained from our multi-view approach with other multi-view depth map estimation methods on the Strecha dataset. Reported values from other approaches are taken from Hu and Mordohai [26] as well as Zheng et al. [66]. For evaluation, a depth map for each view was generated. The results in Table 5.7 show the percentage of correct pixels for the thresholds 10 cm and 2 cm, averaged over all views except the two extreme views (first and last) of each dataset.

We show results of our approach before and after depth fusion on half and full resolution (window size 19x19 and 29x29 respectively). Cost combination variant *best-K* performed significantly better than *trunc* on the *Herz-Jesu-P8* image sequence. Hence, for full resolution only results for *best-K* are stated. Furthermore, results of six different approaches are listed for comparison: Zheng et al. (ZHE) [66], Hu and Mordohai (HU) [26], Furukawa and Ponce (FUR) [15], Zaharescu et al. (ZAH) [64], Tylecek and Sara (TYL) [58] and Jancosek and Pajdla (JAN) [30]. The results of HU were generated from downsampled images of a resolution of 1536 x 1024, while results for ZHE were generated from full-resolution images (3072 x 2048). For HU and ZHE we state intermediate results as well as final results after post-processing (PP). Results for FUR, ZAH, TYL and JAN were extracted from 3D models provided by the authors, i.e. filling might be based on more sophisticated surface fitting methods that could also be used in conjunction with our depth maps. E.g. JAN generated 3D point clouds from estimated depth maps that were created from a plane-sweeping approach [31]. This point cloud was then used as input for a surface reconstruction approach utilizing visual hull constraints.

Results show that our method is comparable to state-of-the-art methods, outperforming most competitors on both half and full resolution (only JAN performs better on both sequences). 3D point clouds generated by merging consistent depth estimates of different views are visualized in Figure 5.12. We show colored as well as shaded versions of four different image sequences of the Strecha dataset (*fountain-P11*, *Herz-Jesu-P8*, *entry-P10* and *castle-P19*). Note that for shading our estimated normals are used.

Method		fountain-P11		Herz-Jesu-P8	
		10 cm	2 cm	10 cm	2 cm
half, trunc	Raw	0.939	0.751	0.822	0.645
	Fusion	0.956	0.770	0.871	0.698
half, best-K	Raw	0.939	0.749	0.863	0.655
	Fusion	0.955	0.769	0.918	0.705
full, best-K	Raw	0.924	0.759	0.813	0.605
	Fusion	0.956	0.786	0.913	0.683
ZHE [66]	Raw	0.911	0.732	0.833	0.619
	PP	0.929	0.769	0.844	0.650
HU [26]	Raw	0.874	0.695	0.781	0.584
	Fusion	0.930	0.754	0.848	0.649
FUR [15]		0.838	0.731	0.836	0.646
ZAH [64]		0.832	0.712	0.501	0.220
TYL [58]		0.822	0.732	0.852	0.658
JAN [30]		0.973	0.824	0.923	0.739

Table 5.7: Results of our multi-view stereo approach on the Strecha dataset compared to results reported from Hu and Mordohai [26] as well as Zheng et al. [66].

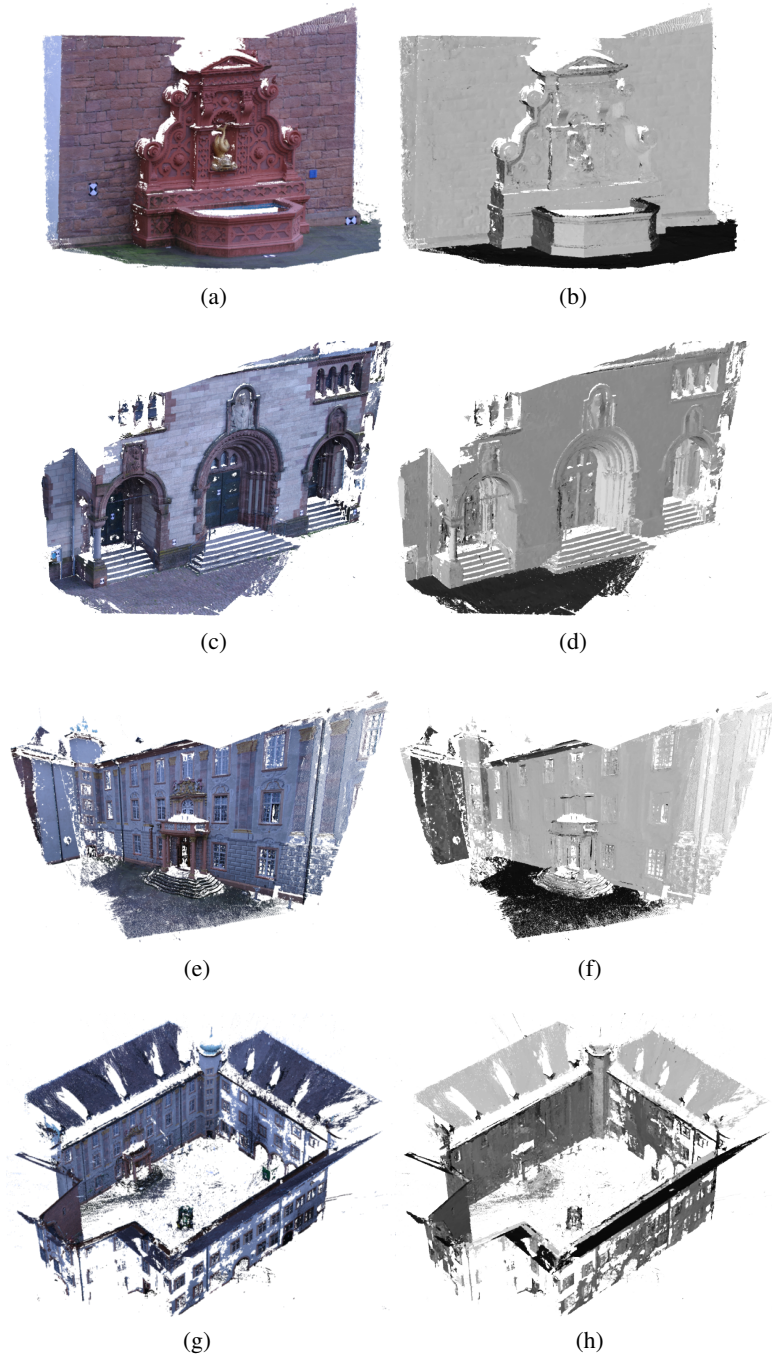


Figure 5.12: Colored (left column) and shaded (right column) 3D meshes generated from fused depth maps of the Strecha dataset on half resolution: (a-b) *fountain-P11*, (c-d) *Herz-Jesu-P8*, (e-f) *entry-P10*, (g-h) *castle-P19*.

Multi-view Reconstruction on Kitti

In our two-view evaluation we covered the Kitti Vision Benchmark Suite (see Section 5.1). Additional to the two-view stereo pairs captured at the same time, stereo pairs from previous and successive frames are provided for the dataset. Since the Kitti scenes are largely static, we can use these additional images for a multi-view setup. Intrinsic and relative extrinsic camera parameters for each stereo pair are provided in the benchmark suite. The relative camera rotation and translation between consecutive stereo pairs were kindly provided by the authors of [59]. They computed the ego-motion of the stereo camera system by minimizing a robust error measure over pre-computed correspondences. A similar ego-motion estimation was introduced by Badino and Kanade [1].

We compare results of our two-view implementation, considering only the current stereo pair, with results of our multi-view algorithm, considering three consecutive image pairs (six images in total). Results for the two-view approach for both cost functions were listed in Table 5.1 in Section 5.1. Since the Census Transform cost function performed best on the Kitti image sequence we use this cost function also for our multi-view algorithm. Results for window sizes 21, 25 and 35 with cost aggregation *best-K* ($K = 2$) are provided in Table 5.8. The use of additional image pairs significantly improved the accuracy of the algorithm. E.g. for window size 35x35 an error rate of 6.27% over all ground truth points is achieved compared to an error rate of 10.94% in the two-view variant. We show visual results for image 110 with Census Transform cost function and window size 25 in Table 5.9. Main differences are a higher number of consistent pixels and detected matches in areas that were partly occluded in the two-view image pair.

Cost	Window size	Error (noc)	Error (all)	Consistent pixel	Error (cons)
Census Transform	35x35	5.89	6.27	85.4	3.48
	25x25	5.85	6.26	82.3	3.05
	21x21	5.87	6.30	80.2	2.87

Table 5.8: Performance (% of disparities with error > 3 pixel) of our multi-view stereo algorithm on the Kitti training set, using six views.


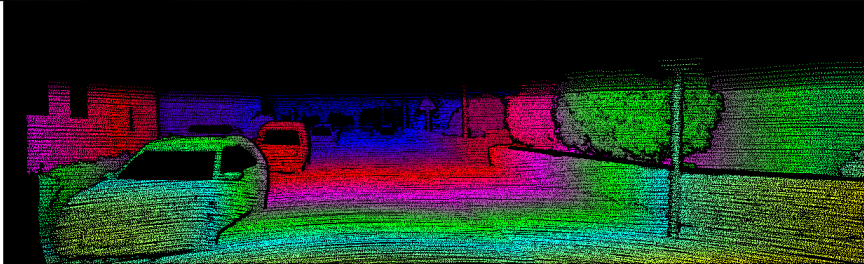
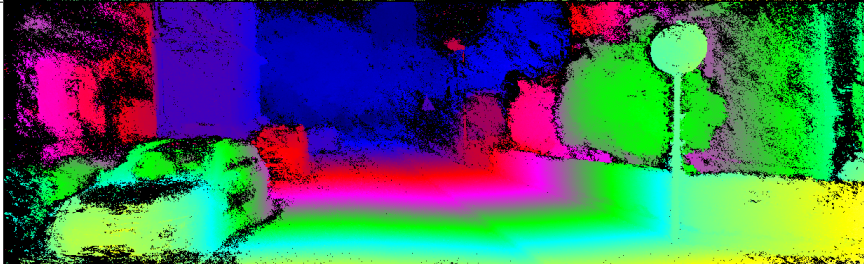
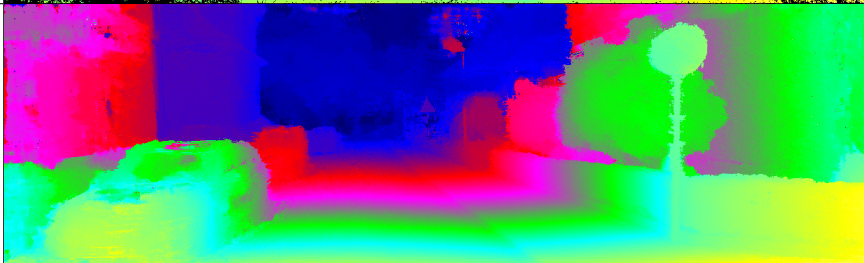
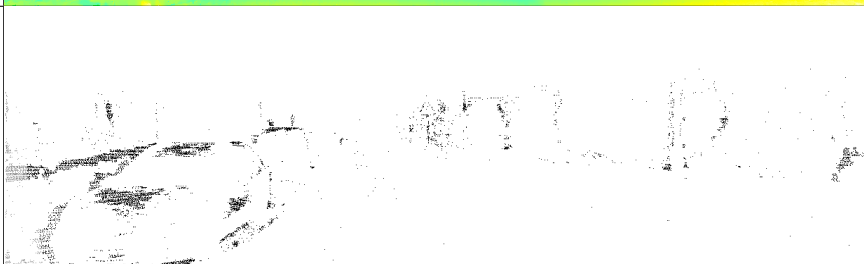
Left Image	
Ground Truth	
Consistent	
Post-Processing	
Error	

Table 5.9: Detailed results for Kitti image 110 with Census Transform cost and window-size 25x25 for the multi-view setup (6 view): Left center image, Ground-truth disparity map, consistent disparity map after fusion, disparity map after post-processing and error map (white: no error, black: noc error, gray: occ error).

Aerial images

We have further qualitatively evaluated our multi-view stereo approach on a set of aerial images. The two tested sequences contain four views of oblique aerial image data from the city of Enschede². We show results for an image resolution of 1404x936 pixels for the two sequences in Figure 5.13 and 5.14 respectively. The PatchMatch cost function was used for both sequences with a window size of 25 and cost combination variant *best-K*. We were able to reconstruct objects with piecewise constant normals, such as streets, roofs and building walls. Erroneous regions that could not be reconstructed correctly (e.g. trees) were removed during the consistency check.

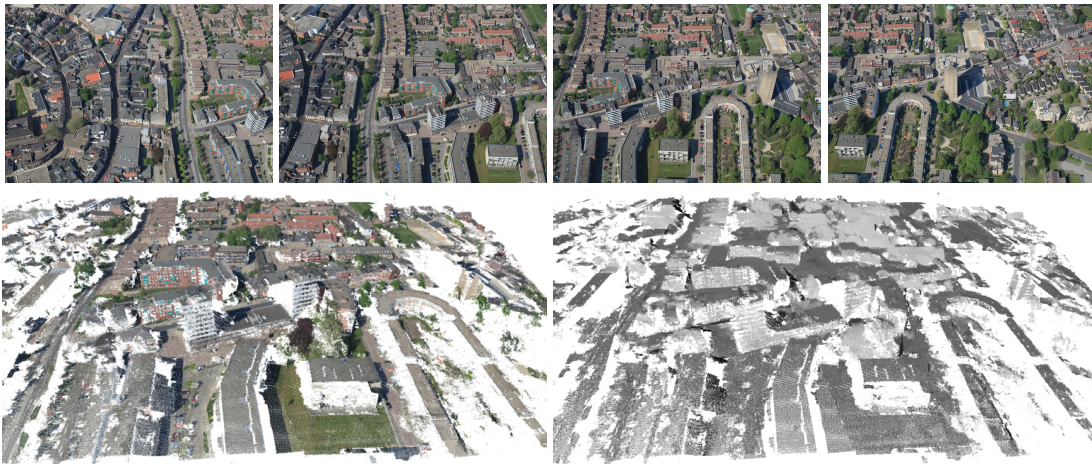


Figure 5.13: Aerial image sequence 1 (top row) with resulting colored and shaded point clouds (bottom row left and right).

5.3 Summary

We have evaluated our approach on various outdoor datasets for both two-view and multi-view image sequences. Quantitative evaluation on different benchmarks show that our method obtains competitive results compared to state-of-the-art methods. Furthermore, qualitative results were shown on high resolution panoramic images and aerial image data. A comparison between two-view and multi-view image sequences on the Kitti dataset revealed that the additional information from more than two views significantly improves reconstruction results.

We have further tested different cost functions and cost combination variants for the aggregation of cost over multiple views. Census Transform based cost functions obtain more robust results for input data with strongly varying lighting conditions, such as the tested Kitti dataset. For scenes with mainly constant lighting conditions the original cost function used by Bleyer et al. [8] generally obtained a higher number of consistent correspondences and more homogeneous normal estimations. Concerning cost aggregation methods, we achieved best results

²<http://www.slagboomenpeeters.com/>

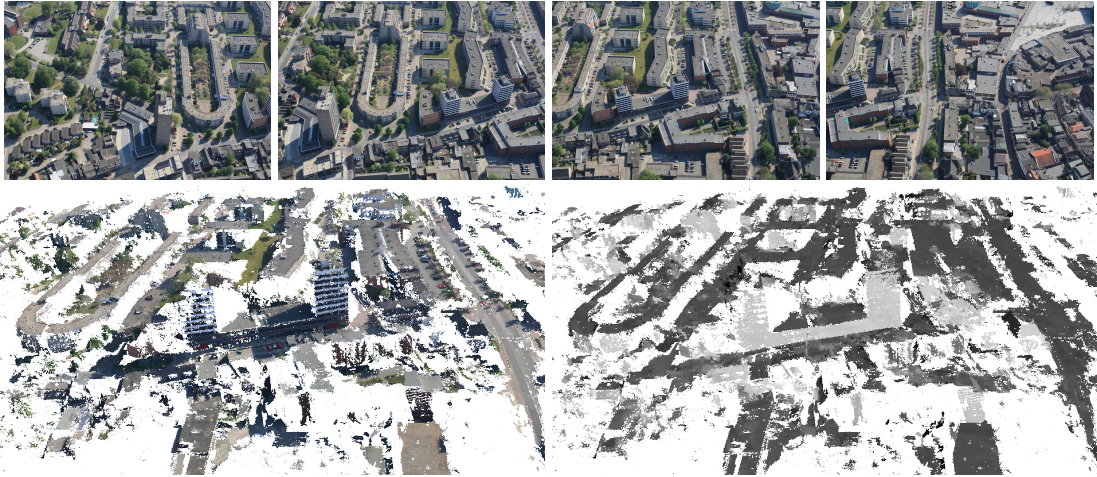


Figure 5.14: Aerial image sequence 2 (top row) with resulting colored and shaded point clouds (bottom row left and right).

when pre-selecting views (limiting the allowed viewing angle difference between cameras) and aggregating only over a subset of selected views. The later was accomplished by two different aggregation variants: *best-K* and *trunc*, both allowing robustness to partial occlusions. Variant *best-K* aggregates over the best K cost values per pixel. In our tests, a value of $K = 2$ obtained best results. The second variant, *trunc*, is a data-driven approach, circumventing the need of selecting a fixed parameter for the number of considered views. We accumulate over all pre-selected views while truncating high costs with a threshold based on the lowest cost value of all views. On our tested datasets both approaches delivered similar results with the variant *best-K* obtaining slightly better quantitative results.

Conclusion

In this thesis we have presented a multi-view stereo matching approach for the estimation of a depth value and surface normal at each pixel position. The approach is based on the randomized correspondence search PatchMatch. This allows efficient memory handling and thus facilitates 3D reconstruction from high-resolution input images.

Our implementation starts from the two-view stereo matching approach PatchMatch Stereo where continuous disparity values and normal directions are estimated from rectified image pairs. We extend the original method to scene space, which allows us to handle multiple non-rectified images in a true multi-view setting, i.e. by measuring low-level image similarity over multiple images. Estimated depth and normal maps for each view are further refined in a fusion step by removing inconsistent estimates and filling missing areas with plane information from other views and extrapolation from nearby planes. Moreover, we have tested different cost (similarity) functions for per-pixel matching as well as different schemes to combine the costs across multiple images. The most suitable cost function depends on the dataset (respectively, its illumination conditions and surface properties) and on whether speed or accuracy is more important. Correspondence search is more robust when only accumulating over a selected subset of views and thus implicitly accounting for potential partial occlusions. Concerning speed, working with gray-scale images as well as limiting the number of propagation iterations significantly decreases runtime while still yielding similar results.

We have evaluated our approach on both two-view and multi-view outdoor datasets. Tested images cover a wide range of different settings: street-level images of road scenes under strong lighting variations, oblique aerial image data, high-resolution panoramic images of urban scenes and high-quality images of historic buildings captured explicitly for 3D architectural modeling. Our method was capable of dealing with these different sets of input data and delivered state of the art results on existing benchmarks.

Future Work

Even though the computational cost of our approach scales linearly with the number of pixels in an image (allowing the processing of multiple high-resolution images) runtime remains an issue in our current implementation. This is mainly due to the relatively large windows empirically required for normal estimation: per point, a relatively large support window must be warped to each of the other views with a homography, including bilinear resampling. Thus, future work will focus on computation time reduction.

First steps in this direction were taken by employing alternative cost functions, such as PM Self-Similarity and Sparse Census Transform, and the parallelization of the propagation scheme. The independent spatial propagation along individual rows, respectively columns, allows for a parallel depth estimation on different cores. Implementing this parallel variant on the GPU could further reduce runtime. Alternative directions for runtime improvements are a multi-scale extension of the approach or the use of adaptive window sizes. Starting on lower resolution images or with smaller window sizes and refining estimates in subsequent steps could help to reduce computation time.

A different promising direction for future work is the application of surface reconstruction techniques, such as the one of Jancosek and Pajdla [30], starting from our depth estimates. This could help to reconstruct complicated regions more robustly, and fill in missing values in a more qualified manner, especially in areas with few valid matches.

Bibliography

- [1] Hernan Badino and Takeo Kanade. A head-wearable short-baseline stereo system for the simultaneous estimation of structure and motion. In *Conference on Machine Vision Application (MVA)*, 2011.
- [2] Jasmine Banks and Peter I. Corke. Quantitative evaluation of matching methods and validity measures for stereo vision. *International Journal of Robotics Research*, 20(7):512–532, 2001.
- [3] Linchao Bao, Qingxiong Yang, and Hailin Jin. Fast edge-preserving patchmatch for large displacement optical flow. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [4] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In *Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2009.
- [5] Frederic Besse, Carsten Rother, Andrew Fitzgibbon, and Jan Kautz. Pmbp: Patchmatch belief propagation for correspondence field estimation. In *British Machine Vision Conference (BMVC)*, 2012.
- [6] Michael Bleyer and Sylvie Chambon. Does color really help in dense stereo matching? In *International Symposium 3D Data Processing, Visualization and Transmission (3DPVT)*, 2010.
- [7] Michael Bleyer, Sylvie Chambon, Uta Poppe, and Margrit Gelautz. Evaluation of different methods for using colour information in global stereo matching approaches. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVII, Part B3a*, 2008.
- [8] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo - stereo matching with slanted support windows. In *British Machine Vision Conference (BMVC)*, 2011.
- [9] T. Brox, A. Bruhn, N. Papenbergh, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision (ECCV)*, 2004.

- [10] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):500–513, March 2011.
- [11] Frédéric Devernay and Olivier Faugeras. Straight lines have to be straight: Automatic calibration and removal of distortion from scenes of structured environments. *Mach. Vision Appl.*, 13(1):14–24, August 2001.
- [12] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. A Wiley-Interscience publication. Wiley, 1973.
- [13] W.S. Fife and J.K. Archibald. Improved census transforms for resource-optimized stereo vision. *Circuits and Systems for Video Technology, IEEE Transactions on*, 23(1):60–73, Jan 2013.
- [14] Pascal Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 6(1):35–49, 1993.
- [15] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.
- [16] D. Gallup, J.-M. Frahm, P. Mordohai, Qingxiong Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.
- [17] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [18] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient large-scale stereo matching. In *Asian Conference on Computer Vision (ACCV)*, 2010.
- [19] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
- [20] P. Heise, S. Klose, B. Jensen, and A Knoll. Pm-huber: Patchmatch with huber regularization for stereo matching. In *International Conference on Computer Vision (ICCV)*, 2013.
- [21] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.
- [22] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2):328–341, Feb 2008.
- [23] H. Hirschmuller and D. Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(9):1582–1599, Sept 2009.

- [24] Asmaa Hosni, Michael Bleyer, and Margrit Gelautz. Secrets of adaptive support weight techniques for local stereo matching. *Computer Vision and Image Understanding*, 117(6):620 – 632, 2013.
- [25] I.P. Howard and B.J. Rogers. *Binocular Vision and Stereopsis*. Oxford psychology series. Oxford University Press, 1995.
- [26] Xiaoyan Hu and Philippos Mordohai. Least commitment, viewpoint-based, multi-view stereo. In *International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission (3DIMPVT)*, 2012.
- [27] Fay Huang, Shou Kang Wei, and Reinhard Klette. Stereo reconstruction from polycentric panoramas. In Reinhard Klette, Shmuel Peleg, and Gerald Sommer, editors, *Robot Vision*, volume 1998 of *Lecture Notes in Computer Science*, pages 209–218. Springer Berlin Heidelberg, 2001.
- [28] Stephen S. Intille and Aaron F. Bobick. Disparity-space images and large occlusion stereo. In *European Conference on Computer Vision (ECCV)*, 1994.
- [29] H. Ishiguro, M. Yamamoto, and S. Tsuji. Omni-directional stereo for making global map. In *International Conference on Computer Vision (ICCV)*, 1990.
- [30] M. Jancosek and T. Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.
- [31] Michal Jancosek and Tomas Pajdla. Hallucination-free multi-view stereo. In KiriakosN. Kutulakos, editor, *Trends and Topics in Computer Vision*, volume 6554 of *Lecture Notes in Computer Science*, pages 184–196. Springer Berlin Heidelberg, 2012.
- [32] B. Julesz. *Foundations of Cyclopean Perception*. University of Chicago Press, 1971.
- [33] Sing Bing Kang, R. Szeliski, and Jinxiang Chai. Handling occlusions in dense multi-view stereo. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.
- [34] Sing Bing Kang and Richard Szeliski. 3-d scene data recovery using omnidirectional multibaseline stereo. *International Journal of Computer Vision*, 25(2):167–183, November 1997.
- [35] Sing Bing Kang and Richard Szeliski. Extracting view-dependent depth maps from a collection of images. *International Journal of Computer Vision*, 58(2):139–163, July 2004.
- [36] George Marsaglia. Choosing a point from the surface of a sphere. *The Annals of Mathematical Statistics*, 43(2):645–646, 04 1972.
- [37] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1995.

- [38] Wided Miled and Beatrice Pesquet-Popescu. The use of color information in stereo vision processing. In Marta Mrak, Mislav Grgic, and Murat Kunt, editors, *High-Quality Visual Experience*, Signals and Communication Technology, pages 311–330. Springer Berlin Heidelberg, 2010.
- [39] M. Okutomi and T. Kanade. A multiple-baseline stereo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(4):353–363, Apr 1993.
- [40] Jafar Amiri Parian and Armin Gruen. Sensor modeling, self-calibration and accuracy testing of panoramic cameras and laser scanners. *{ISPRS} Journal of Photogrammetry and Remote Sensing*, 65(1):60 – 76, 2010.
- [41] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénus, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2-3):143–167, July 2008.
- [42] R. Ranftl, S. Gehrig, T. Pock, and H. Bischof. Pushing the limits of stereo using variational stereo estimation. In *Intelligent Vehicles Symposium (IV)*, 2012.
- [43] Christoph Rhemann, Asmaa Hosni, Michael Bleyer, Carsten Rother, and Margrit Gelautz. Fast cost-volume filtering for visual correspondence and beyond. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.
- [44] T. W. Ryan, R. T. Gray, and B. R. Hunt. Prediction of correlation errors in stereo-pair images. *Optical Engineering*, 19(3):193312–193312–, 1980.
- [45] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, April 2002.
- [46] Shuhan Shen. Accurate multiple view 3d reconstruction using patch-based stereo for large-scale scenes. *Image Processing, IEEE Transactions on*, 22(5):1901–1914, May 2013.
- [47] Heung-Yeung Shum and Richard Szeliski. Stereo reconstruction from multiperspective panoramas. In *International Conference on Computer Vision (ICCV)*, 1999.
- [48] Sudipta N. Sinha, Daniel Scharstein, and Richard Szeliski. Efficient high-resolution stereo matching using local plane sweeps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [49] I. Sobel and G. Feldman. A 3x3 Isotropic Gradient Operator for Image Processing. Never published but presented at a talk at the Stanford Artificial Project, 1968.
- [50] Fridtjof Stein. Efficient computation of optical flow using the census transform. In *Pattern Recognition*, volume 3175 of *Lecture Notes in Computer Science*, pages 79–86. Springer Berlin Heidelberg, 2004.

- [51] Christoph Strecha, Wolfgang von Hansen, Luc J. Van Gool, Pascal Fua, and Ulrich Thoenessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.
- [52] Changming Sun and S. Peleg. Fast panoramic stereo matching using cylindrical maximum surfaces. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(1):760–765, Feb 2004.
- [53] R. Szeliski. A multi-view approach to motion and stereo. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 1999.
- [54] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.
- [55] Engin Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(5):815–830, May 2010.
- [56] Engin Tola, Christoph Strecha, and Pascal Fua. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Machine Vision and Applications*, 23(5):903–920, 2012.
- [57] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *International Conference on Computer Vision (ICCV)*, 1998.
- [58] Radim Tylecek and Radim Sara. Refinement of surface mesh for accurate multi-view reconstruction. *International Journal of Virtual Reality*, 9:45–54, 2010.
- [59] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *International Conference on Computer Vision (ICCV)*, 2013.
- [60] Qingxiong Yang, Ruigang Yang, J. Davis, and D. Nister. Spatial-depth super resolution for range images. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.
- [61] Y. Yang, A Yuille, and J. Lu. Local, global, and multilevel stereo matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 1993.
- [62] Kuk-Jin Yoon and In-So Kweon. Locally adaptive support-weight approach for visual correspondence search. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.
- [63] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *European Conference on Computer Vision (ECCV)*, 1994.
- [64] A Zaharescu, E. Boyer, and R. Horaud. Topology-adaptive mesh deformation for surface evolution, morphing, and multiview reconstruction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(4):823–837, April 2011.

- [65] Yilei Zhang, Minglun Gong, and Yee-Hong Yang. Local stereo matching with 3d adaptive cost aggregation for slanted surface modeling and sub-pixel accuracy. In *International Conference on Pattern Recognition (ICPR)*, 2008.
- [66] Enliang Zheng, Enrique Dunn, Vladimir Jovic, and Jan-Michael Frahm. Patchmatch based joint view selection and depthmap estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [67] Christian Zinner, Martin Humenberger, Kristian Ambrosch, and Wilfried Kubinger. An optimized software-based implementation of a census-based stereo matching algorithm. In *Advances in Visual Computing*, volume 5358 of *Lecture Notes in Computer Science*, pages 216–227. Springer Berlin Heidelberg, 2008.