Spring 4-27-2015

Robust and Real-Time Stereo Matching on Parallel Graphics Hardware using Gradient-Based Disparity Refinement

Jedrzej Kowalczuk
University of Nebraska-Lincoln, jkowalczuk@unl.edu

ROBUST AND REAL-TIME STEREO MATCHING ON PARALLEL GRAPHICS HARDWARE USING GRADIENT-BASED DISPARITY REFINEMENT

by

Jedrzej Kowalczuk

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Doctor of Philosophy

Major: Engineering

(Electrical Engineering)

Under the Supervision of Professor Lance C. Pérez

Lincoln, Nebraska

April, 2015

ROBUST AND REAL-TIME STEREO MATCHING ON PARALLEL GRAPHICS HARDWARE USING GRADIENT-BASED DISPARITY REFINEMENT

Jedrzej Kowalczuk, Ph.D.

University of Nebraska, 2015

Advisor: Lance C. Pérez

Computer vision attempts to provide camera-equipped machines with visual perception, i.e., the capability to comprehend their surroundings through the analysis and understanding of images. The ability to perceive depth is a vital component of visual perception that enables machines to interpret the three-dimensional structure of their surroundings and allows them to navigate through the environment. In computer vision, depth perception is achieved via stereo matching, a process that identifies correspondences between pixels in images acquired using a pair of horizontally offset cameras. It is possible to calculate depths from correspondences or, more specifically, the positional offsets (disparities) between pixels in correspondence.

A stereo matching method implemented on massively parallel graphics hardware is presented that allows for recovery of highly accurate disparities in real-time. This method combines a pixel dissimilarity metric computed using both the gradients and the census transforms of the input images, a non-iterative local disparity selection scheme based on an efficient approximation of the well-known edge-preserving bilateral image filter, and a refinement technique that iteratively improves the accuracy of disparities. The refinement technique, which also benefits from the use of the bilateral filter, eliminates mismatches by penalizing disparities that disagree with the the disparity estimates generated using local disparity values and/or gradients.

When evaluated using the Middlebury stereo performance benchmark (version 3), the proposed method ranks first and second to date using the training and test image sets, respectively, in terms of the overall accuracy of stereo matching measured as the average percentage of pixels with the absolute disparity error greater than 2 pixels at the nominal image resolution. Simultaneously, the method achieves the lowest error rates for 5 out of 15 image pairs in the training set, and 3 out of 15 image pairs in the test set. This method is also shown to enable robust matching in the presence of radiometric distortions caused by changes in illumination or camera exposure. The high accuracy of matching, that is largely maintained in the presence of radiometric distortions and the ability to operate in real time, make the proposed method well-suited for applications such as robotic navigation and structure reconstruction.

ACKNOWLEDGEMENT

First and foremost, I would like to sincerely thank my advisor Lance C. Pérez for his guidance and, most importantly, his friendship throughout the course of my doctoral program. Lance, without your advice and support this dissertation could not come to fruition and my emergence as a complete researcher would not be possible. It was a great honor and pleasure to be a member of the Perceptual Systems Research Group. I recognize and appreciate the effort that you have invested into its foundation and I am confident that this friendly work/learning environment will shape many successful researchers and future academic leaders. I direct my hopes and ambitions to achieve the level of academic and personal excellence that you possess.

It cannot be argued that Eric Psota is one of the most influential individuals that I have met in my graduate career. Eric's passion, infectious enthusiasm, and the highest work ethic make him an outstanding researcher. Eric, thank you for sparking my interest in image processing and computer vision. I was extremely fortunate to have you as a friend, a coworker, member of my doctoral committee and, in many ways, my co-advisor. Your help and advice were indispensable to my academic development and to the creation of this dissertation. My gratitude to you is beyond words.

I am also thankful to the members of my doctoral committee: Khalid Sayood, who fulfilled the role of the primary reader with incredible care and thoroughness, and Ashok Samal, who represented computer science and engineering, which was once my own field of study. Your invaluable comments have significantly improved both the presentation and the contents of this document.

Certainly, I could not complete this journey without the support from so many others. Words cannot express how grateful I am to my parents for their constant and immeasurable encouragement. I need to thank Bret Beermann and his family whom I shared so many memorable moments with; you are all very dear to me. Furthermore, I would like to thank the members of our volleyball club for their great company, both on and off the court. Finally, my most special thanks go to Luisa Rios-Avila, whose love, understanding, and patience were essential to the completion of this difficult and long process. You have all contributed irreversibly to the person I have become.

Contents

CHAPTER 1

Introduction

Computer vision strives to make computers understand images the way humans do by attempting to mimic some of the processes ongoing in the human visual system. One such process that has received a lot of attention in computer vision is the process of depth perception. While the exact mechanisms governing depth perception remain unknown, researchers agree that multiple depth cues are synthesized by the visual cortex of the brain to create the perception of depth. Depth cues can be classified as monocular, i.e., those arising from the retinal projections of the observed environment in each individual eye or each eye's responses to oculomotor stimuli, and binocular, those that integrate information from both eyes.

An important subclass of monocular cues is motion-related phenomena, which includes motion parallax and optical expansion. The former enables the evaluation of relative distances between objects based on their motion with respect to the background in the view of a moving observer. The latter is perceived as an increase in the size of objects as they move towards the observer. Depth cueing is also supported by the eye's

ability to adjust its focusing power, which is known as accommodation. The brain is able to determine whether an object is close to the observer or at a greater distance by analyzing the amount of blur as certain elements of the view go out of focus, while monitoring contractions of the ciliary muscles that control the shape of the lens. Note that, with the exception of accommodation at short range or motion parallax under known direction and velocity of the observer's movement, the monocular cues do not allow for recovery of absolute depth information; this is due to the dimensional compression associated with the 3D to 2D projection of the physical environment to the retina of each eye. This process is generally non-invertible.

Conversely, it is possible to recover absolute depths from binocular cues. Although depth at a short viewing range can be estimated from the angle of convergence between the eyeballs, human depth perception is primarily driven by differences in the appearance of objects in the images derived from both eyes. Specifically, the positional offsets between 3D to 2D projections in each eye are used by the brain to calculate the depth of objects in a process known as *stereopsis* or *stereo vision*.

In computer vision, stereopsis can be mimicked using a setup of two offset cameras that serve as a pair of artificial eyes. If the distance between the cameras is known, absolute depths can be obtained from the positional offsets relating corresponding points in a pair of images. The problem of identifying such correspondences or, equivalently, the retrieval of positional offsets from a pair of images, is called *stereo matching*. Stereo matching is the primary subject of this dissertation. In what follows, a brief history and an overview of previous work on stereo matching are given, and the contribution and structure of the dissertation are then explained.

1.1 Overview of Stereo Matching

In 1838, long before the era of computers and digital imaging, Wheatstone [1] demonstrated the differential nature of stereopsis. He did so by presenting two slightly different images of the same object to a viewer using a stereoscope, a device that enables a pair of images to be displayed to individual eyes through a system of mirrors. The viewer was able to fuse the images into a single image with perceivable depth.

Until 1954 when Aschenbrenner [2] popularized random-dot stereograms, pairs of images consisting of randomly generated dot patterns where depths of an imaginary object are encoded by displacements between the corresponding dots in the two images, it was believed that stereopsis was not possible in the absence of monocular cues or without any understanding of what is shown in the images. Using a stereoscope or by focusing the eyes behind a plane containing the pair of dot patterns, the viewers were able to establish correspondences between the dots, extract their displacements, and interpret their depths. This suggested that neither monocular cues nor recognition of the images are necessary for stereopsis as long the viewer is able to identify correspondences of features in the images.

By studying the capability of the human visual system to detect patterns in sequences of random numbers represented as images, Julesz [3, 4] later arrived at a samilar conclusion. In his book entitled "Foundations of Cyclopean Perception", Julesz stated that depth information can be inferred from positional offsets between projections of objects in the two views that arise form the lateral displacement of the

eyes, even if no identifiable objects are present in the views, or if no monocular cues are available. These offsets, called *binocular disparities* or *stereo disparities*, enable depths to be triangulated with high accuracy if the distance between the eyes is known. This distance, and how it relates disparities to depth, is learned by individuals in the early stage of their development.

The work of Julesz [5] also resulted in the first formulation of *stereo matching* as a problem of extracting the disparities relating visual features in a pair of views of the same scene. Two less commonly used names for the same problem are *stereo correspondence* and *binocular fusion*. A popular assumption at the time was that the human visual processor solves the stereo matching problem on random-dot stereograms by exchanging information between a large number of disparity-sensitive neurons, and that this exchange of information facilitates rejection of false targets which, in the case of random-dot stereograms, occur in profusion. This assumption led researchers to develop a class of cooperative algorithms [6–8] that enabled the recovery of disparities from random-dot stereograms, but failed when applied to real imagery.

In 1979, Marr and Poggio presented a theory of stereo vision [9] that integrated previous advancements in stereo matching. The elimination of false targets was central to the theory and was deemed to have difficulty proportional to the range and resolution of disparities present in the images. A coarse-to-fine stereo matching algorithm was proposed that matches zero crossings of the smoothed second directional derivatives of the images. These correspond to sharp changes in the image intensity functions, e.g., object boundaries. The amount of smoothing when computing the second directional derivatives is successively decreased, allowing the algorithm to progress from evaluating

large disparities at low resolution to evaluating small disparities (increments to the previously computed ones) at fine resolution. An implementation of this algorithm due to Grimson [10], which followed shortly after the original publication, was shown to enable matching of natural images.

Lucas and Kanade were the first to formulate stereo matching as an optimization problem. Their formulation from 1981 [11], which still prevails today, is as follows. Given two image intensity functions $I_0(\boldsymbol{x})$ and $I_1(\boldsymbol{x})$, the goal is to recover a disparity $\boldsymbol{d}$ at every pixel location $\boldsymbol{x}$, such that some measure of similarity between $I_0(\boldsymbol{x})$ and $I_1(\boldsymbol{x}+\boldsymbol{d})$ is maximized; this measure is said to quantify the photometric consistency of the solution. Oftentimes, the equivalent problem where some measure of dissimilarity between $I_0(\boldsymbol{x})$ and $I_1(\boldsymbol{x}+\boldsymbol{d})$ is to be minimized is solved. If the minimization is performed independently for every pixel the stereo matching method is said to be local. If the minimization is performed jointly for the entire image the stereo matching method is said to be global [12]. This remarkably simple formulation was also adopted in the problems of feature detection and matching, and optical flow estimation that stemmed from stereo matching. Together with stereo matching, these problems defined the field of visual correspondence.

A plethora of local stereo matching methods were developed during the mid-to-late 80's and 90's that attempted to maximize the similarity between windows of pixels centered at the pixels under consideration. Numerous window shapes, sizes, and weighting functions were proposed, along with various pixel similarity measures. These are surveyed in [13, 14]. With new methods being published every month, it became necessary to establish a set of performance metrics according to which these methods

could be evaluated and compared. Scharstein and Szeliski, recognizing this necessity, created the Middlebury stereo performance benchmark [12] that provides a common methodology for the evaluation of stereo matching methods.

Meanwhile, developments in projective geometry, i.e., the discipline that studies the process of image formation in cameras, helped researchers understand the geometric constraints relating projections of objects in a pair of views. It was shown that a setup of two identical cameras that are horizontally displaced and oriented in the same direction produces pairs of images with purely horizontal disparities [15]. In this case, matches can be found by searching within the corresponding scanlines of the images, making the problem one-dimensional. While this setup of cameras is hard to achieve in practice, it is possible to resample the images such that the matches become restricted to the corresponding scanlines. Nearly all modern stereo matching methods require the images to be rectified this way prior to matching.

From 2000 to 2005, global methods based on graph cuts, belief propagation, and tree-reweigthed message passing emerged that enabled the minimization of energy functions encoding both the global photometric consistency and smoothness of the solution. A recent survey of these methods may be found in [16]. The ability to explicitly model smoothness assumptions as part of the minimization problem allowed global methods to surpass the most accurate local methods with respect to performance benchmarks. The performance gap between local and global methods was eliminated when Yoon and Kweon introduced a window-based method that adapts the weights based on the color similarity and spatial distance between pixels within a window [17]. Their approach, which is essentially a reformulation of the bilateral image filter (a

well-known edge-preserving filter), recovers locally smooth solutions with well-defined object boundaries with accuracy that approaches that of global methods. Dozens of new methods extending the existing approaches, both local and global, have since been proposed that achieved further accuracy imrovements.

1.2 Motivation and Contribution

Stereo matching enables a myriad of applications. Perhaps the most promising of them is robotic navigation, where depth information is necessary for obstacle avoidance in driverless cars, autonomous drones, etc., or for visual servoing in industrial automation. Furthermore, the ability to recover depth information from images renders stereo matching particularly useful in structure reconstruction, a process that reconstructs detailed geometric models of the environment or individual objects. Such models are often imported into mapping software, computer games, or applications of virtual reality. Last, but not least, stereo matching has the potential to enhance video surveillance, object recognition, and image segmentation, a lot of which have been traditionally performed using flat images.

In addition to the accuracy of the matching, many of these applications require the stereo matching to be performed in real time. The extremely high computational complexity of stereo matching, however, prevents real-time implementation using modern general-purpose processors despite the availability of multiple processing cores. As an example, consider matching a pair of 640×480 images using a local method that evaluates disparities ranging from 0 to 60 pixels at every pixel of the reference

image, which results in a total of 18.7 million distinct disparity hypotheses being evaluated. Since a single disparity hypothesis in local methods is typically evaluated by aggregating intensity differences between corresponding pixels located within windows surrounding the pixels under consideration, and since each such window often contains hundreds of pixels, billions of arithmetic operations are necessary to match images of this modest size.

Recent advances in graphics hardware triggered the emergence of parallel programming frameworks that enable the computational power in the many-core, massively parallel graphics processing units to be used for purposes other than accelerated rendering of graphics. The graphics hardware and the complementary programming frameworks were immediately adopted by researchers as tools for implementation of high-performance stereo matching algorithms. Since 2007, a number of local methods have been proposed that achieve near real-time or real-time performance. They manage to do so by using simplified, computationally efficient matching techniques that come at great expense of the accuracy of matching. What makes local methods particularly feasible for parallel implementation on graphics hardware is their computational regularity and mostly non-iterative operation. Global methods, on the other hand, do not lend themselves well to parallel implementation as they often employ the inherently sequential graph-based message passing schemes, or require a large number of iterations to converge at an accurate solution.

To address the need for real-time and accurate stereo matching, a new method is proposed that achieves accuracy comparable to or exceeding the accuracy of global methods, while maintaining the computational efficiency of local methods. This method

incorporates a two-pass approximation of the adaptive support weights that makes it possible to establish an initial set of correspondences in an efficient, non-iterative way, and a novel disparity refinement technique based on probabilistic inference that is applied iteratively in order to improve the accuracy of matching. When implemented on graphics hardware, the method enables matching with interactive frame rates at image resolutions that are sufficient for tasks such as robotic navigation.

The refinement technique, which is derived using a probabilistic framework, operates by penalizing disparities that deviate from expected values computed using the disparity information from nearby pixels. Combined with a confidence measure that assesses the reliability of individual matches, the penalties are used to identify and overwrite erroneous disparity assignments. A consensus on the disparity assignment using the refinement is achieved in just a few iterations, which is less than the number of iterations required by global methods by at least an order of magnitude. The computational complexity of disparity refinement is hence kept at a fraction of the computational complexity of the operations necessary for initial matching.

Using a collection of images with known true disparities and the evaluation criteria provided by the Middlebury stereo performance benchmark, the proposed method is shown to offer high accuracy of matching. Compared to all previously developed stereo matching methods, including those that cannot operate in real time, the proposed method ranks among the most accurate methods and, in many cases, is the most accurate method to date. In addition, when a pixel similarity measure is used that incorporates differences of gradients and census transforms evaluated at the pixels under consideration, the proposed method enables matching of images with radiometric

distortions, where the corresponding pixels differ in intensities due, for instance, to inconsistent camera settings. The high accuracy, invariance to radiometric distortions, and real-time operation make the proposed method a suitable option when reliable and efficient stereo matching is required.

The key contributions of this work are about expanding stereo matching to reflect a more sophisticated model of human vision. First, the local disparity selection framework is revised and extended to enable the disparity maps obtained in an initial stage of matching to be iteratively refined by penalizing the cost values at pixels whose disparities disagree with the disparity estimates calculated using the information from nearby pixels. This has two important implications. First, matching can be performed in a way that more closely resembles stereopsis in the human visual system. Specifically, humans are believed to identify correspondences by first determining regions where the correspondences exist, likely using additional depth cues, and then determining individual points that correspond. The initial matching and the disparity refinement within the proposed method are analogous to this model of the human visual system.

Second, the methods developed here allow for the the incorporation of additional disparity evidence into the process of matching. Stereo matching is commonly formulated as a problem of calculating a disparity map given a pair of rectified images, each in the form of a discrete-valued intensity function defined at integer pixel locations. Basing the disparity estimation solely on the similarities between the image intensity functions is an oversimplification of this much more complex visual correspondence problem, and arguably the fundamental limitation of many existing stereo matching

methods. Stereo matching can certainly benefit from the integration of external knowledge about the scene, e.g., its structure and objects within it, into the process of matching. However, the algorithmic structure of existing methods, particularly the local ones, often prohibits this. In contrast, the proposed method allows for additional disparity evidence to be included by augmenting the cost penalty function with terms that quantify the plausibility of individual disparity hypotheses with respect to what is known about the scene. Such disparity evidence can be obtained by performing contextual analysis of the scene using machine learning, e.g., object detection and recognition, or by using additional sensors, e.g., depth sensors or inertial measurement units, in combination with a conventional stereo camera rig.

Furthermore, the proposed method is the first one to use a cost metric that evaluates a weighted sum of distances between both the census vectors and the gradients at the pixels under consideration. While humans discriminate between matching and non-matching image regions in a census-like way, i.e., by assessing relative color differences between nearby points, in the case of sampled, discrete-valued image intensity functions the census transform becomes easily dominated by noise, and thus limits the method's ability to correctly perform such a discrimination. The addition of the gradient term overcomes this by capturing the local trend of intensities around the pixel of interest, similar to the way humans are able to capture gradual color transitions and use this information in depth inference. Note that both components of the cost metric are invariant to intensity shifts. As a result, changes in scene illumination or camera exposure do not affect the method's ability to discriminate image regions.

Finally, the proposed method defines a confidence measure that quantifies the uniqueness of matches with respect to their local pixel neighborhoods. The confidence measure is essential to the recovery of disparities in weakly textured image regions. Once again, the inspiration is found in the human visual system that is able to match weakly textured regions if reliable correspondences are known in nearby, sufficiently more distinct regions. To mimic this, the proposed method iteratively overwrites disparities in problematic regions with the estimates generated using a reformulation of the bilateral filter that operates on disparities, local disparity gradients, and confidence measures of nearby pixels. The confidence measures are recalculated in every iteration of refinement, which promotes the exchange of disparity information across the image and contributes to the accuracy of the method.

The remaining part of this dissertation is organized as follows. Essential concepts of projective geometry, including the camera model and constraints of the two-view geometry, are covered in Chapter 2. These concepts are then used to derive a relationship between disparities and depth in a setup of two offset cameras. A comprehensive review of visual correspondence problems follows in Chapter 3, where formal definitions of stereo matching and related problems are given and common solution methods are outlined. In Chapter 4, the proposed stereo matching method is derived and described, and details of its implementation on parallel graphics hardware are discussed. In Chapter 5, a widely accepted evaluation methodology is used to qualitatively and quantitatively assess the performance of the proposed method in terms of both the accuracy and the computational efficiency of matching. The proposed method is then compared to existing approaches. Further in Chapter 5, the

proposed method is integrated into a structure reconstruction pipeline, and examples of geometric models recovered from video sequences are given to illustrate the effects of this integration. A summary of the contents, discussion of strengths and weaknesses of the proposed method, and suggestions for future work conclude the dissertation in Chapter 6.

CHAPTER 2

Elements of Projective Geometry

As indicated in the introduction, stereo matching benefits from the assumption that the offset between the cameras is purely horizontal and that the cameras are oriented in the same direction. In order to understand the geometric constraints relating the locations of the matching points in a pair of image acquired using such a camera setup it is necessary to understand the fundamental concepts of *projective geometry*, i.e., the discipline that studies the projection of 3D world points to 2D image points in digital cameras. The fundamental concepts of projective geometry are covered in this chapter. In section 2.1, a common notation is established for geometric primitives and transformations of coordinate spaces and a camera model is then derived in section 2.2; estimation of camera models is discussed in section 2.3. Later in section 2.4, an arbitrary setup of two cameras is studied and constraints of the two-view geometry are derived. Finally, in sections 2.5 and 2.6 the setup of two parallel, horizontally offset cameras is considered to establish the relationship between disparities and depths.

2.1 Homogeneous Notation

One of the fundamental concepts of projective geometry assumes the use of vectors to represent lines and points in 2-dimensional (2D) Euclidean space. For instance, a line defined by the equation $ax + by + c = 0$ for some choice of a, b and c, can be represented by a vector $(a, b, c)^\mathsf{T}$. Note that vectors $(a, b, c)^\mathsf{T}$ and $k(a, b, c)^\mathsf{T}$ represent the same line, since $ax + bx + c = 0$ is equivalent to $k(ax + bx + c) = 0$, for any non-zero constant k. Any two vectors of the form $k(a, b, c)^\mathsf{T}$ are called *homogeneous vectors* and, since the scaling factor is not important, are considered equivalent. The application of homogeneous vectors in representing geometric primitives, such as lines and points, is known as *homogeneous notation* or *homogeneous coordinates*.

Using the homogeneous notation, a point $(x, y)^\mathsf{T}$ can be represented by any vector of the form (kx, ky, k). Typically, k is set to 1, thus the *inhomogeneous* point $(x, y)^\mathsf{T}$ becomes $(x, y, 1)^\mathsf{T}$ in homogeneous coordinates. In general, a homogeneous vector $\boldsymbol{x} = (x_1, x_2, x_3)^\mathsf{T}$ corresponds to a point $(x_1/x_3, x_2/x_3)^\mathsf{T}$ in 2-dimensional Euclidean space; only the quantities x_1/x_3 and x_2/x_3 are meaningful, since x_3 is treated as a scaling factor. An advantage associated with the homogeneous representation of lines and points in 2D Euclidean space is that algebraic operations involving these primitives can be performed using elementary vector operations. In particular, if $\boldsymbol{x}$ and $\boldsymbol{x}'$ are two points in 2D Euclidean space, the line $\boldsymbol{l}$ joining these points can be computed by evaluating the cross product of $\boldsymbol{x}$ and $\boldsymbol{x}'$, i.e.,

$$\boldsymbol{l} = \boldsymbol{x} \times \boldsymbol{x}' \,. \tag{2.1}$$

Likewise, the point of intersection of two lines $\boldsymbol{l}$ and $\boldsymbol{l}'$ is $\boldsymbol{x} = \boldsymbol{l} \times \boldsymbol{l}'$. One can also verify that for any point $\boldsymbol{x}$ belonging to a line $\boldsymbol{l}$, the inner product the two primitives $\boldsymbol{x}$ and $\boldsymbol{l}$ is zero, i.e.,

$$\boldsymbol{x}^{\mathsf{T}}\boldsymbol{l} = 0\,. \tag{2.2}$$

The concept of homogeneous coordinates extends to planes and points in higher-dimensional spaces. In order to obtain a homogeneous representation of a plane in n-dimensional space ($n \geq 3$), a vector is formed that contains all $n+1$ coefficients of the plane; n-dimensional points become vectors of length $n+1$, with 1 added as the last element.

Analogously, linear transformations (mappings between spaces) are represented with homogeneous matrices, i.e., matrices that are related by a non-zero scaling factor. In general, a $(n+1) \times (n+1)$ matrix of real elements can be used to represent any transformation in n-dimensional space, and commonly the scaling factor is chosen in a way that makes the last element of the transformation matrix equal to 1. Homogeneous representation of transformations, combined with homogeneous representation of geometric primitives, enables easy computation of transformed primitives by performing matrix-vector (or vector-matrix) multiplication. Specifically, if a point $\boldsymbol{x}$ is mapped to a point $\hat{\boldsymbol{x}}$ under some transformation represented by a matrix $\boldsymbol{H}$, the point $\hat{\boldsymbol{x}}$ is computed as

$$\hat{\boldsymbol{x}} = \boldsymbol{H}\boldsymbol{x}. \tag{2.3}$$

An inverse transformation can be found by computing $\boldsymbol{H}^{-1}$, which imposes a non-singularity constraint on $\boldsymbol{H}$. Furthermore, compound transformations can be computed

by multiplying transformation matrices of component transformations. This way for instance, rotation and translation can be combined into a single transformation matrix and applied to geometric primitives in a single step, which is not possible using inhomogeneous coordinates.

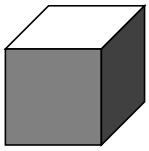
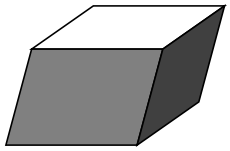
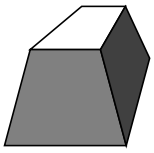
A class of transformations particularly important to the problems covered in this dissertation is the class of transformations of the 3-dimensional (3D) Euclidean space. A hierarchy of such transformations was established in [18] based on the number of parameters (degrees of freedom, dof), the type of geometric distortion introduced, and geometric properties invariant to these transformations. General transformation matrices and distortion examples associated with various groups of transformations of the 3-dimesional Euclidean space are listed in Table 2.1. The hierarchy of transformations of the 3D Euclidean space includes four subclasses, which are ordered from the most specialized (fewer degrees of freedom) to the most general (the most degrees of freedom):

- Euclidean transformations (isometries). The family of Euclidean transformations embodies two groups of volume-preserving transformations: translations and rotations, each introducing 3 degrees of freedom.

- Similarity transformations (similarities). Similarities extend Euclidean transformations with another degree of freedom in the form of a scale parameter. Since distances between points are not preserved by the scaling operation, neither are volumes; similarities preserve angles between lines and planes.

- Affine transformations (affinities). The 9-dof affine transformations are com-

Table 2.1: Hierarchy of transformations of 3D Euclidean space [18].

Group name	Matrix†	Distortion example
Euclidean (6 dof)	$\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^\mathsf{T} & 1 \end{bmatrix}$	
Similarities (7 dof)	$\begin{bmatrix} s\boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^\mathsf{T} & 1 \end{bmatrix}$	
Affinities (9 dof)	$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{0}^\mathsf{T} & 1 \end{bmatrix}$	
Perspectivities (15 dof)	$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{v}^\mathsf{T} & v \end{bmatrix}$	

† The matrix $\boldsymbol{R}$ is an orthogonal 3D rotation matrix, whose orthonormal rows define the orientation of the axes in the transformed coordinate system, $\boldsymbol{t} = (t_x, t_y, t_z)^\mathsf{T}$ is a translation vector, $\boldsymbol{A}$ is a 3×3 invertible matrix, $\boldsymbol{v}$ is a general 3-element vector, $\boldsymbol{0} = (0, 0, 0)^\mathsf{T}$ is an all-zero vector, and s and v are scalars.

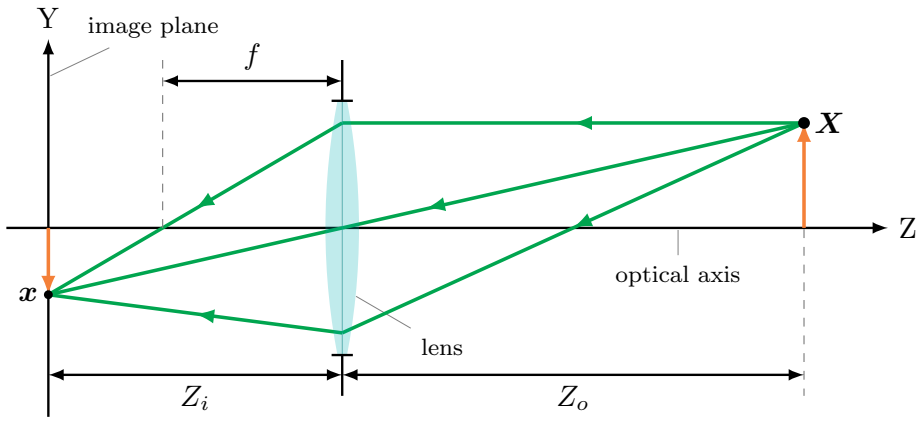
binations of shearing, scaling, rotation and translation. Parallelism of planes, volume ratios and centroids of point clouds are the main invariants of affine transformations.

- Projective transformations (projectivities). Transformations in this group are the general linear transformations of the 3D Euclidean space, where all but the last element of the transformation matrix can take arbitrary values, implying 15 degrees of freedom. Projectivities map lines to lines and preserve length ratios of line segments.

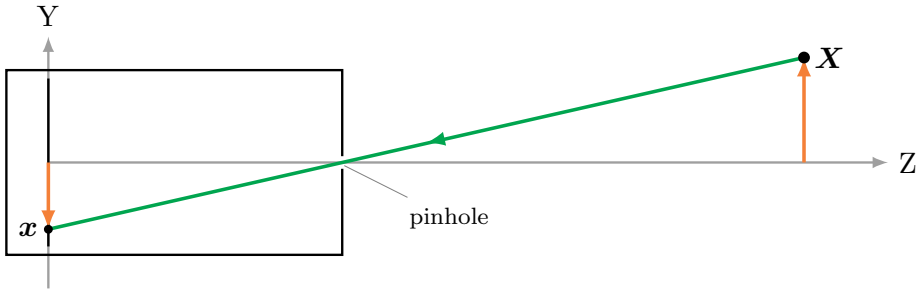
In the next section, homogeneous representation of both geometric primitives and transformations of the 3-dimensional Euclidean space are used to derive a pinhole camera model, i.e., a simplified mathematical model that describes the process of geometric image formation in digital cameras.

2.2 Geometric Image Formation

Modern digital cameras commonly use optical lenses to focus light rays on an image sensor that contains an array of photodetectors. Although cameras often come equipped with multi-lens optical systems, in order to understand the process of geometric image formation, it is best to consider a camera with a single biconcave lens, such as the camera shown in Figure 2.1a. In the figure, the horizontal axis runs through the optical centre of the lens and defines the *principal ray*, i.e., the look direction of the camera, otherwise known as the *optical axis* or the *principal axis*. The vertical axis represents the *image plane* (precisely, a side view of the image plane), that is, a plane



(a) Thin lens camera



(b) Pinhole camera

Figure 2.1: Image formation in thin lens (a) and pinhole (b) cameras.

perpendicular to the optical axis where the images are formed; the image plane is conceptually equivalent to an image sensor.

Light rays emitted or reflected from the observed objects travel along multiple paths through the lens experiencing refraction at air-glass interfaces (causing the rays to converge) and are captured on the image plane. The directions of light rays exiting the lens, and thus their point of convergence, are strictly determined by the refractive index of the glass and the curvature of the boundary surfaces of the lens. These two properties are conveniently characterized by the *focal length* of the lens, which is interpreted as a distance along the optical axis of the camera over which light rays

originating from an infinitely distant source converge. If the thickness of the lens is much smaller that its focal length, or equivalently, if the radii of curvature of the lens surfaces are much greater than the focal length, the lens is said to follow the *thin lens model* given by the *thin lens equation*

$$\frac{1}{z_i} + \frac{1}{z_o} = \frac{1}{f}\,, \tag{2.4}$$

where z_o is the distance between the lens and the observed object, z_i is the distance between the lens and the image plane, and f is the focal length of the lens.

An important implication following from the thin lens model is that in order for the light rays to converge at a single point on the image plane, the image plane must be located at a distance z_i from the lens that satisfies equation (2.4). Conversely, if the distance z_i does not satisfy the thin lens equation, the light rays will reach the image plane over a disc-shaped area, resulting in image blur.

2.2.1 The Pinhole Camera Model

To avoid modeling lenses and their associated phenomena, computer vision applications commonly adopt a simplified variant of the thin lens camera, which is known as a *pinhole camera*. A pinhole camera is illustrated in Figure 2.1b. In a pinhole camera, light rays are transferred onto the image plane through an infinitesimally small aperture, hereafter referred to as a *pinhole*, to form upside-down images of the observed objects. The result of substituting the usual lens with a pinhole are perfectly sharp, all-in-focus images, since a single world point always projects to a single point on the image plane. Note that, in the absence of a lens, the meaning of the focal length parameter changes.

In a pinhole camera, the focal length simply defines the distance between the image plane and the pinhole; this distance is often given in image units, i.e., pixels.

To derive a mathematical model describing image formation in pinhole cameras a case of *central projection* is considered that is shown in Figure 2.2. In the case of central projection, the *optical center* $\boldsymbol{C}$ of the camera (here, the pinhole), is located at the origin of the world coordinate system and the camera's optical axis is collinear with the world's Z-axis. The optical axis intersects with the image plane at the *principal point* $\boldsymbol{p} = (0, 0)^\mathsf{T}$ given in image coordinates. Additionally, to avoid having to account for the vertical inversion of images, it is convenient to assume that the image plane is located in front of the pinhole. The action of the pinhole camera describes a projection of points in the 3-dimensional Euclidean coordinate space $\mathbb{R}^3$ of the world to a 2-dimensional Euclidean coordinate space $\mathbb{R}^2$ of the image plane located at $\mathrm{Z} = f$.

Under central projection using a pinhole camera, a point $\boldsymbol{X} = \left(X, Y, Z\right)^\mathsf{T}$ in the world coordinates projects to a point $\boldsymbol{x} = \left(f\frac{X}{Z}, f\frac{Y}{Z}, f\right)^\mathsf{T}$ on the image plane. Thus, the mapping between the two coordinate spaces is given by

$$\left(X, Y, Z\right)^\mathsf{T} \mapsto \left(f\frac{X}{Z}, f\frac{Y}{Z}\right)^\mathsf{T}, \tag{2.5}$$

which corresponds to

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.6}$$

in homogeneous coordinates. The equation in (2.6) is known as the *projection equation*

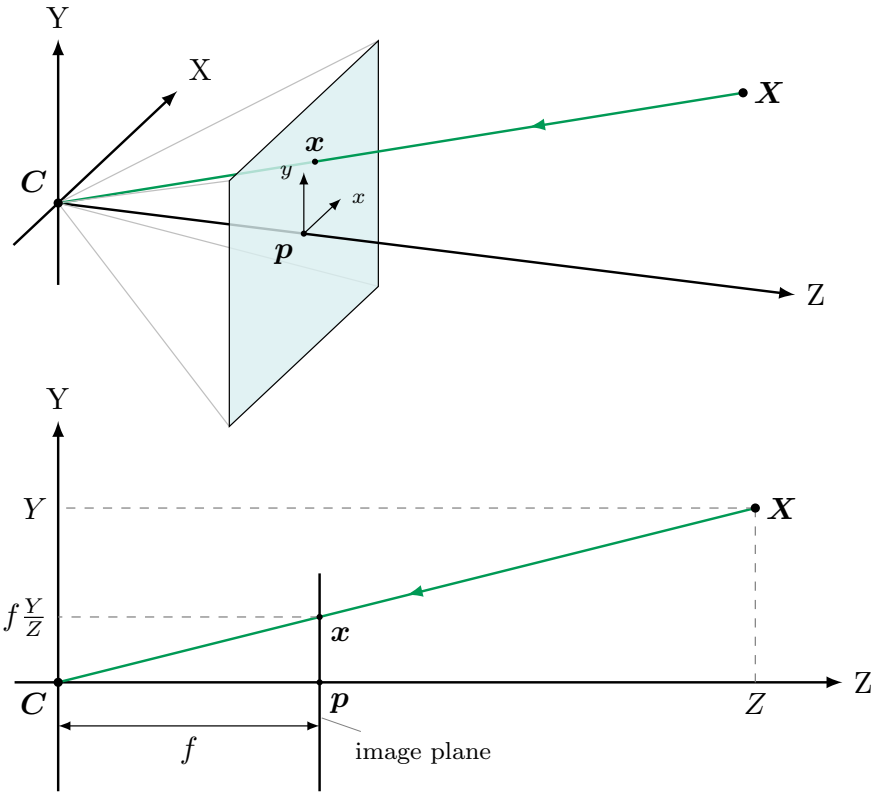


Figure 2.2: Central projection using a pinhole camera.

and is often written compactly as

$$\boldsymbol{x} = \boldsymbol{P}\boldsymbol{X}\,, \tag{2.7}$$

where the 3×4 matrix $\mathtt{P} = diag(f, f, 1)[\boldsymbol{I} \mid \boldsymbol{0}]$ is the *camera projection matrix*.

Since, based on a common convention, either the top left or the bottom left corner of the image is chosen to be the origin of the image coordinate system, the principal point is displaced from the origin by approximately half of the image size in both horizontal and vertical direction. Note that, however, when modeling real cameras as pinhole cameras, the principal point will generally not be at the exact center of the image, as the physical alignment of the lens with respect to the image sensor may not reflect the case shown in Figure 2.2. In practice, the optical center of the lens may not be positioned on the optical axis of the camera and/or the plane containing the lens may not be perfectly parallel to the plane containing the image sensor. In rare cases, physical imperfections of the lens may further affect the position of the principal point. To account for the offset of the principal point, now assumed to be arbitrarily placed at $\boldsymbol{p} = (p_x, p_y)^\mathsf{T}$, the mapping in equation (2.5) is rewritten as

$$\Big(X, Y, Z\Big)^\mathsf{T} \mapsto \Big(f\frac{X}{Z} + p_x, f\frac{Y}{Z} + p_y\Big)^\mathsf{T}\,, \tag{2.8}$$

and the corresponding equation in homogeneous coordinates now reads

$$\begin{bmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{bmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.9}$$

Introducing

$$\boldsymbol{K} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}, \tag{2.10}$$

equation (2.9) can be written as

$$\boldsymbol{x} = \boldsymbol{K}[\boldsymbol{I} \mid \boldsymbol{0}]\boldsymbol{X}\,, \tag{2.11}$$

where the matrix $\boldsymbol{K}$ is called the *(camera) calibration matrix*. A more general form of the calibration matrix

$$\boldsymbol{K} = \begin{bmatrix} m_x f & s & p_x \\ & m_y f & p_y \\ & & 1 \end{bmatrix} = \begin{bmatrix} f_x & s & p_x \\ & f_y & p_y \\ & & 1 \end{bmatrix} \tag{2.12}$$

is sometimes used, where m_x and m_y express the numbers of pixels per unit distance in the x- and y-direction of the image coordinate system, and s is a skew parameter. Incorporating the parameters m_x and m_y in the calibration matrix allows for modeling of cameras whose associated image coordinate systems have unequal scales in both directions, i.e, cameras having non-square pixels. In such cameras, two distinct focal length parameters $f_x = m_x f$ and $f_y = m_y f$ control the projection of X- and Y-coordinates of the observed world points, respectively. The skew parameter, on the other hand, allows for modeling of image coordinate systems defined by non-perpendicular axes; in this case, s takes a non-zero value. The non-zero elements of $\boldsymbol{K}$, that is, the focal lengths f_x and f_y, the skew parameter s, and the coordinates of the principal point p_x and p_y, are known as *intrinsic parameters*, *camera intrinsics* or *internal camera parameters*.

The model in equation (2.11) can further be generalized by allowing the coordinate systems of the world and the camera to be nonequivalent. Assume that the camera coordinate system is related to the world coordinate system by a rotation matrix $\boldsymbol{R}$ and a translation vector $\boldsymbol{t}$, such that $\boldsymbol{R}$ rotates the the camera coordinate system to match the orientation of the world coordinate system, and $\boldsymbol{t}$ shifts the origin of the camera coordinate system to the origin of the word coordinate system. The rotation matrix $\boldsymbol{R}$ and the translation vector $\boldsymbol{t}$ are jointly referred to as *extrinsic parameters* or *external camera parameters*. To enable arbitrary location and orientation of the camera, a rigid transformation composed of the translation $\boldsymbol{t}$ and the rotation $\boldsymbol{R}$ is applied to the world points prior to projecting the world points onto the image plane. The projection equation becomes

$$\boldsymbol{x} = \boldsymbol{K}[\boldsymbol{R} \mid \boldsymbol{t}]\boldsymbol{X} \,. \tag{2.13}$$

Observing that $\boldsymbol{t} = -\boldsymbol{R}\boldsymbol{C}$, the camera center can be made explicit in equation (2.13) resulting in

$$\boldsymbol{x} = \boldsymbol{K}\boldsymbol{R}[\boldsymbol{I} \mid -\boldsymbol{C}]\boldsymbol{X} \,. \tag{2.14}$$

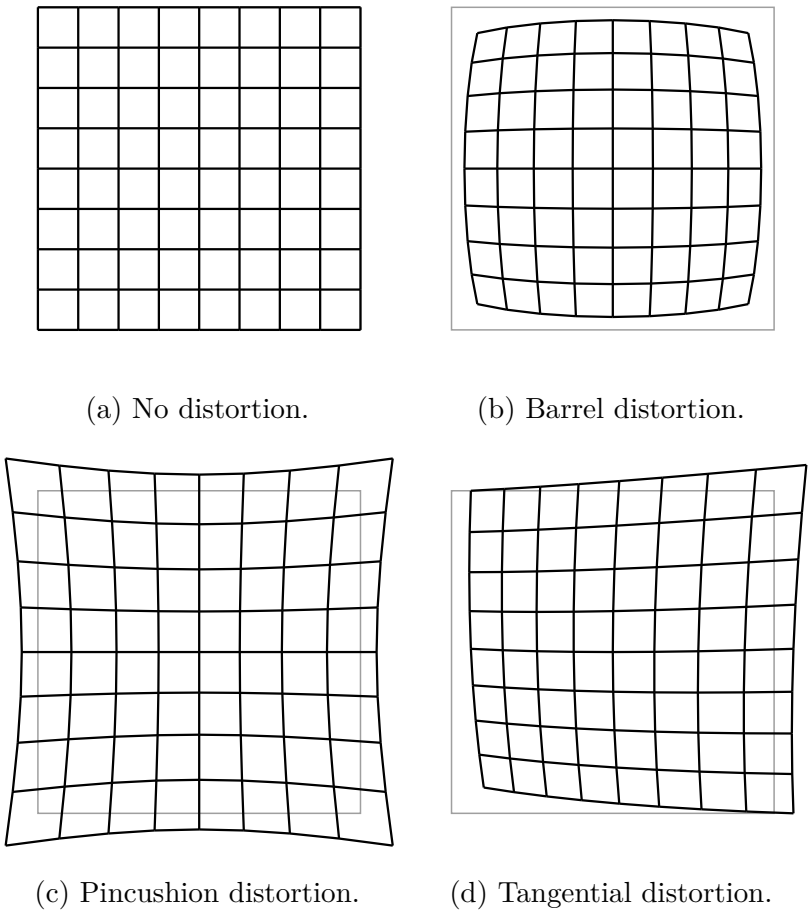
2.2.2 Image Distortion

The projection operation described by the pinhole camera model is rectilinear or line-preserving, i.e., straight lines present in the scene are projected as straight lines in the images. While the pinhole camera model idealizes and greatly simplifies the process of geometric image formation in digital cameras, used alone the model is often imprecise and thus insufficient in applications of computer vision, largely due to its

inherent inability to model image distortions introduced by the camera optics. Image distortion is understood as any deviation of image coordinates from the coordinates dictated by the rectilinear projection, as defined by equation (2.13), caused by the shape of the lens or defects in manufacturing or assembling of the optical components of the camera. Two types of image distortion that frequently occurr in cameras are the following:

- Radial distortion occurs in cameras equipped with spherical lenses and causes lines to curve toward or away from the image center, i.e., the principal point, as the distance from the optical center increases. In general, rectilinear projection is possible with parabolic lenses, which, due to difficulties in manufacturing, are often replaced with spherical lenses whose light-bending properties differ between the center and the edges of the lens. As a result, images appear magnified around the principal point (barrel-type radial distortion) or around the edges of the image (pincushion-type radial distortion), although a combination of both effects is not uncommon. Illustrations of barrel-type and pincushion-type radial distortion are given in Figures 2.3b and 2.3c. Radial distortion is symmetric around the principal point.

- Tangential or decentering distortion results from misalignment between the lens and the image sensor, particularly, from the lens not being perfectly parallel to the image sensor. In that event, magnification is observed in image areas corresponding to the greater physical distance between the image sensor and the lens, and the areas corresponding to the smaller lens-to-sensor distance



(a) No distortion. (b) Barrel distortion.

(c) Pincushion distortion. (d) Tangential distortion.

Figure 2.3: Common types of image distortion.

appear compressed. Tangential distortion is illustrated in Figure 2.3d. Note that tangential distortion is asymmetric.

The effects of both radial and tangential distortion are non-linear, and thus cannot be represented using homogeneous transformation matrices. Distortion models proposed in the literature include rational [19, 20] or polynomial functions [21–24] of the radial distance, i.e., a distance on the image plane from the principal point to a point of interest, that capture the relationship between the coordinates of the distorted image and the undistorted coordinates obeying the rectilinear projection.

The distortion models based on polynomial functions, which are more commonly used, are variants of the Brown-Conrady model [25, 26] given by

$$\hat{x} = x(1 + k_1 r^2 + k_2 r^4 + \ldots) + (2p_1 xy + p_2(r^2 + 2x^2))(1 + p_3 r^2 + p_4 r^4 + \ldots) \quad (2.15)$$

$$\hat{y} = y(1 + k_1 r^2 + k_2 r^4 + \ldots) + (2p_2 xy + p_1(r^2 + 2y^2))(1 + p_3 r^2 + p_4 r^4 + \ldots) \quad (2.16)$$

where $(\hat{x}, \hat{y})$ are undistorted pixel coordinates, $r = \sqrt{(x - p_x)^2 + (y - p_y)^2}$ is the radial distance from the principal point, $k_1, k_2, \ldots$ are the radial distortion coefficients, and $p_1, p_2, \ldots$ are the tangential distortion coefficients. In most cases, the four coefficients k_1, k_2, p_1, and p_2 are sufficient to accurately model the image distortion. However, enabling higher order radial distortion coefficients is recommended if the radial distortion is severe, e.g, when working with ultra wide-angle lenses.

Supplemented with the distortion coefficients, the pinhole camera model becomes adequate for modeling the process of geometric image formation in digital cameras. In particular, knowledge of the distortion model allows for the distortion to be removed through resampling of the distorted images at the coordinates obtained from the distortion model, making the pinhole camera model in equation (2.13) a valid representation of the camera's projective action. Methods of estimation of camera parameters and distortion coefficients are detailed in section 2.3.

2.3 Camera calibration.

The term camera calibration refers to the process of estimating camera parameters from a set of correspondences between geometric entities in the world coordinate space and their equivalents on the image plane. Camera calibration is a crucial process

in applications of computer vision that attempt to extract metric information from images. For instance, the accuracy of geometric models obtained using scene structure reconstruction methods strictly depends on the accuracy of the estimated camera parameters.

Camera calibration is typically performed by observing a calibration object, i.e., a reference object of known geometry. Methods of image processing are applied to the views of the calibration object in order to identify and extract the corresponding geometric entities. Such world-to-image correspondences deliver mathematical constraints necessary to estimate the camera parameters. While there exist techniques capable of determining camera parameters from correspondences between world and image lines [27], the majority of approaches rely on point correspondences [28–33]. Calibration techniques can be classified according to the dimensionality of the calibration objects used. The following classes of camera calibration techniques have been recognized in [34]:

- Calibration from a 3D reference object. This category of camera calibration methods require a calibration object of known 3D geometry, e.g., an object that consists of two or three orthogonal planes. These planes typically contain patterns that support automatic extraction of world-to-image correspondences from views of the calibration object. Alternatively a planar object experiencing precisely known motion can be used to acquire the necessary correspondences [28]. Given a sufficient number of world-to-image correspondences (typically in a single view of the calibration object), both the intrinsic properties of the camera

and its orientation can be inferred from the constraints on camera parameters resulting from the projection equation [18].

- Calibration from a 2D reference object. This type of calibration is performed using a planar calibration object that contains a grid pattern whose corner points can be automatically detected, e.g., a checkerboard pattern or a grid of circles. Different orientations of the pattern are presented to the camera, which is assumed to be at the origin of the world coordinate system. World-to-image correspondences extracted from multiple views are used to estimate internal camera parameters (globally, integrating the information from all views) along with the orientations of the calibration object in each view. Notable approaches in this category include [28–30].

- Calibration from a 1D reference object. This technique, also known as line-based calibration, exploits the constraints resulting from collinearity of points. It has been shown [27], that camera parameters can be recovered by observing the motion of 3 or more points belonging to a line that is rotated around one of the points, given that distances between the points are known.

- Self-calibration. Unlike the previously described techniques, self-calibration does not require any calibration object and relies purely on the displacements of image points arising from the motion of the camera in a static scene [31–33]. Combining image correspondences and rigidity constraints of the scene allows the camera intrinsic properties, motion parameters, and sparse scene structure to be recovered up to an unknown scale parameter, which, however, is a much more

complex problem than estimation of camera parameters alone. Self calibration is sometimes referred to as 0D calibration or auto-calibration.

Due to large number of parameters being estimated and computational complexity of self-calibration techniques, pre-calibration, i.e., calibration using a reference object, is recommended when possible. Among the calibration techniques involving the use of calibration objects, the 3D calibration is known to produce the most accurate results, since only one view is processed, and thus, the least number of parameters need to be estimated. The 3D calibration, however, requires a precise 3D calibration object (or a 2D one that can be precisely actuated), which is harder to realize than the objects used in 2D calibration. Furthermore, automatic extraction of correspondences between the points of the 3D calibration object and the image points requires sophisticated image processing techniques that are capable of recognizing individual planes of the calibration object. For the above reasons, 2D camera calibration techniques are frequently used in the applications of computer vision. In the following, Zhang's approach [30] is described that estimates camera parameters using multiple views of a planar calibration object. This approach employs mathematical methods that are fundamental to many estimation problems occurring in projective geometry.

Assume that n views of the calibration object have been acquired. The plane of the calibration object containing the grid pattern will be referred to as the model plane. In a similar way, the points located on the model plane will be referred to as model points. A single view of the calibration object delivers m model-to-image correspondences, each of the form $\boldsymbol{x}_i \leftrightarrow \boldsymbol{X}_i$, where $\boldsymbol{X}_i = (X_i, Y_i, 0, 1)^\mathsf{T}$ denotes a

model point and $\boldsymbol{x}_i = (x_i, y_i, 1)^\mathsf{T}$ denotes the corresponding image point. Under the pinhole camera model, the mapping between the model points and the image points is given by the projection equation, i.e.,

$$(x_i, y_i, 1)^\mathsf{T} = \begin{bmatrix} f_x & s & p_x \\ & f_y & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} (X_i, Y_i, 0, 1)^\mathsf{T} \tag{2.17}$$

where $r_{11}, \ldots, r_{33}$ are the entries of the rotation matrix $\boldsymbol{R}$ that holds the orientation of the camera, and the values t_1, t_2, and t_3 are the components of the corresponding translation vector $\boldsymbol{t}$, i.e., $\boldsymbol{t} = (t_1, t_2, t_3)^\mathsf{T}$. Noticing that the Z-coordinates of the model points are all zeros, the third column of the matrix containing extrinsic parameters in equation (2.17) can be eliminated, which yields

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & & p_x \\ & f_y & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix} . \tag{2.18}$$

It follows from equation (2.18), that the image point $\boldsymbol{x}_i = (x_i, y_i, 1)^\mathsf{T}$ and the point on the model plane $\boldsymbol{X}_i = (X_i, Y_i, 1)^\mathsf{T}$ (with the Z-coordinate removed) are related via a projective transformation $\boldsymbol{H}$, such that

$$\boldsymbol{H} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} = \boldsymbol{K} \left(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{t} \right) , \tag{2.19}$$

where $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are the first and the second column of $\boldsymbol{R}$. In particular, the transformation matrix $\boldsymbol{H}$ represents the mapping between coordinate spaces of the

image and the model plane. Any mapping between coordinate spaces that can be represented using a projective transformation will hereafter be called a *homography*.

If the homography $\boldsymbol{H}$ can be estimated for each view, so can be the intrinsic properties of the camera and the orientation parameters associated with each view of the calibration object. Let $\boldsymbol{h}_1$, $\boldsymbol{h}_2$, and $\boldsymbol{h}_3$ be the columns of $\boldsymbol{H}$, i.e., $\boldsymbol{H} = [\boldsymbol{h}_1\ \boldsymbol{h}_2\ \boldsymbol{h}_3]$. Since $\boldsymbol{R}$ is a rotation matrix, the columns of $\boldsymbol{R}$ form an orthonormal basis, thus

$$\boldsymbol{r}_1^{\mathsf{T}}\boldsymbol{r}_2 = 0 \tag{2.20}$$

and

$$\|\boldsymbol{r}_1\| = \|\boldsymbol{r}_2\| = 1\,. \tag{2.21}$$

From equation (2.19)

$$\boldsymbol{r}_1 = \boldsymbol{K}^{-1}\boldsymbol{h}_1 \tag{2.22}$$

$$\boldsymbol{r}_2 = \boldsymbol{K}^{-1}\boldsymbol{h}_2\,, \tag{2.23}$$

which substituted into equations (2.20) and (2.21) results in

$$\boldsymbol{h}_1^{\mathsf{T}}(\boldsymbol{K}^{-1})^{\mathsf{T}}\boldsymbol{K}^{-1}\boldsymbol{h}_2 = 0 \tag{2.24}$$

$$\boldsymbol{h}_1^{\mathsf{T}}(\boldsymbol{K}^{-1})^{\mathsf{T}}\boldsymbol{K}^{-1}\boldsymbol{h}_1 = \boldsymbol{h}_2^{\mathsf{T}}(\boldsymbol{K}^{-1})^{\mathsf{T}}\boldsymbol{K}^{-1}\boldsymbol{h}_2\,. \tag{2.25}$$

Equations (2.24) and (2.25) contain two constraints on camera parameters resulting from a single view of the calibration object. In Zhang's calibration technique, the constraints resulting from multiple views are combined in order to recover the camera parameters and the orientation and translation of the calibration object in each view with respect to the camera center. The approach involves two steps: estimation of

homographies relating points on the model plane and image points, and computation of camera intrinsics and orientation parameters based on the homography matrices estimated for each view. These steps are described in sections 2.3.1 and 2.3.2.

2.3.1 Estimation of Planar Homographies

Solving the camera calibration problem requires a set of homographies to be estimated that relate the coordinate systems of the model planes and the image plane. In the following, a method known as the Direct Linear Transformation algorithm (DLT) is described that can be used to estimate the homographies. The DLT algorithm provides a general approach for computing transformation matrices from a set of similarity relations between geometric entities. In case of camera calibration, a 3×3 homography matrix is computed for each view of the calibration object, given a set of correspondences between the points on the model plane and their corresponding image coordinates. The steps described in the remaining part of this section cover homography estimation for a single view; for clarity, subscripts indicating the view number are omitted.

Recall from equation (2.17), that the points on the model plane and their images are related via unknown homography $\boldsymbol{H}$, such that $\boldsymbol{x}_i = \boldsymbol{H}\boldsymbol{X}_i$ holds up to scale for each correspondence. Taking cross products of both sides with $\boldsymbol{x}_i$ gives

$$\boldsymbol{x}_i \times \begin{bmatrix} \bar{\boldsymbol{h}}_1 \boldsymbol{X}_i \\ \bar{\boldsymbol{h}}_2 \boldsymbol{X}_i \\ \bar{\boldsymbol{h}}_3 \boldsymbol{X}_i \end{bmatrix} = \boldsymbol{0}\,, \tag{2.26}$$

where $\boldsymbol{h}_k$ denotes the k-th row of $\boldsymbol{H}$. The cross product formula is applied to equation

(2.26), resulting in

$$\begin{bmatrix} y_i\bar{\boldsymbol{h}}_3\boldsymbol{X}_i - \bar{\boldsymbol{h}}_2\boldsymbol{X}_i \\ \bar{\boldsymbol{h}}_1\boldsymbol{X}_i - x_i\bar{\boldsymbol{h}}_3\boldsymbol{X}_i \\ x_i\bar{\boldsymbol{h}}_2\boldsymbol{X}_i - y_i\bar{\boldsymbol{h}}_1\boldsymbol{X}_i \end{bmatrix} = \boldsymbol{0}\,, \tag{2.27}$$

which, by rearranging of terms and grouping of unknowns, is reformulated as

$$\begin{bmatrix} \boldsymbol{0}^\mathsf{T} & -\boldsymbol{X}_i^\mathsf{T} & y_i\boldsymbol{X}_i^\mathsf{T} \\ \boldsymbol{X}_i^\mathsf{T} & \boldsymbol{0}^\mathsf{T} & -x_i\boldsymbol{X}_i^\mathsf{T} \\ -y_i\boldsymbol{X}_i^\mathsf{T} & x_i\boldsymbol{X}_i^\mathsf{T} & \boldsymbol{0}^\mathsf{T} \end{bmatrix} \boldsymbol{h} = \boldsymbol{0}\,, \tag{2.28}$$

where $\boldsymbol{h} = \left(\bar{\boldsymbol{h}}_1, \bar{\boldsymbol{h}}_2, \bar{\boldsymbol{h}}_3\right)^\mathsf{T}$ is a column vector that contains the elements of $\boldsymbol{H}$ in row-major order. Equation (2.28) describes a system of three homogeneous equations, among which only two equations are linearly independent (e.g., multiplying the first equation by x_i and the second one by y_i, adding them and then multiplying by -1 gives the third equation). Thus, each model-plane-to-image-plane correspondence provides two constraints on the elements of the homography matrix. Stacking the constraints from all such correspondences into a $2n \times 9$ matrix $\boldsymbol{A}$ leads to a larger system of homogeneous equations, given by

$$\boldsymbol{A}\boldsymbol{h} = \boldsymbol{0}\,, \tag{2.29}$$

which is solved for $\boldsymbol{h}$, and, once the vector $\boldsymbol{h}$ is known, its elements are used to form the homography matrix $\boldsymbol{H}$. The equations in (2.29) are linear in the elements of $\boldsymbol{h}$, while the matrix $\boldsymbol{A}$ contains coefficients that are quadratic combinations of coordinates on the model and image planes; a solution method for this system of equations is discussed in the remaining part of this section.

The homography matrix $\boldsymbol{H}$ has 9 elements rearranged into a column vector $\boldsymbol{h}$, of which one can be assumed to be a scale factor (commonly, the last element of $\boldsymbol{h}$). Since each correspondence provides two equations, at least $n = 4$ unique correspondences are required to compute the 8 unknown elements of $\boldsymbol{H}$. In the minimal case, the solution is the null space of $\boldsymbol{A}$, given that correspondences are exact. In practice, upwards of 50 correspondences are extracted from a typical planar calibration pattern, causing the system in equation (2.29) to become overdetermined. Moreover, since the image measurements are corrupted with noise, there will not exist an exact non-trivial solution. Instead, a non-zero vector $\boldsymbol{h}$ is found that best fits $\boldsymbol{Ah} = \boldsymbol{0}$ in the least-squares sense, i.e.,

$$\boldsymbol{h} = \underset{h}{\operatorname{argmin}} \|\boldsymbol{Ah}\|, \tag{2.30}$$

subject to the constraint $\|\boldsymbol{h}\| = 1$. Such a vector $\boldsymbol{h}$ is known to be the right singular vector associated with the smallest singular value of $\boldsymbol{A}$, and can be computed from Singular Value Decomposition (SVD) of $\boldsymbol{A}$. Precisely, if $\boldsymbol{A} = \boldsymbol{UDV}^{\mathsf{T}}$ is the SVD of $\boldsymbol{A}$, where $\boldsymbol{D}$ is a diagonal matrix containing singular values of $\boldsymbol{A}$ (square roots of eigenvalues of $\boldsymbol{AA}^{\mathsf{T}}$ and $\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}$) in descending order, $\boldsymbol{U}$ is an orthogonal matrix with columns made up of right singular vectors of $\boldsymbol{A}$ (eigenvectors of $\boldsymbol{AA}^{\mathsf{T}}$), and $\boldsymbol{V}$ is an orthogonal matrix whose columns are made up of left singular vectors of $\boldsymbol{A}$ (eigenvectors of $\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}$), the least-squares solution to the system in equation (2.29) is the last column of $\boldsymbol{V}$.

The solution method, however, is not invariant to the choice of image and model plane's coordinate frames, i.e., the accuracy of estimation varies based on the choice of

image and model plane origins, and/or coordinate units. In certain cases, e.g., coordinates on the model plane expressed in meters (small numbers) and image coordinates expressed in pixels (significantly larger numbers), the coefficients in equation (2.29) will differ by orders of magnitude, making the problem poorly conditioned mathematically. To prevent this, simple normalization of point data prior to homography estimation is recommended. Respectively, a transformation matrix $\boldsymbol{T}$ is applied to the image points $\{\boldsymbol{x}_i\}$ that translates the centroid of the points to the origin and scales the points such that their average distance from the origin is $\sqrt{2}$; points on the model plane are normalized using a transformation matrix $\boldsymbol{T}'$ of analogous properties. An intermediate homography $\tilde{\boldsymbol{H}}$ is then found by setting up and solving the system in (2.29) for the normalized points. Finally, the sought homography $\boldsymbol{H}$ is computed as $\boldsymbol{H} = \boldsymbol{T}'^{-1}\tilde{\boldsymbol{H}}\boldsymbol{T}$, which reverses the effects of normalization.

The solution obtained using the DLT algorithm is based on minimization of an algebraic distance $\|\boldsymbol{A}\boldsymbol{h}\|$, whose geometric interpretation is not meaningful to the homography estimation problem. In particular, this distance does not assess the quality of the computed homography matrix. For this reason, it is recommended that the initial homography estimate be further refined by minimizing the total geometric distance between the model points transformed into the coordinate space of the image plane via the matrix $\boldsymbol{H}$, and the actual image measurements. The total geometric error is given by the functional

$$\sum_i \|\boldsymbol{x}_i - f(\boldsymbol{X}_i; \boldsymbol{H})\|^2 , \tag{2.31}$$

where $f(\boldsymbol{X}_i; \boldsymbol{H}) = \boldsymbol{H}\boldsymbol{X}_i$ represents the mapping applied to the coordinates of the

i-th model point, parametrized by the homography matrix $\boldsymbol{H}$. Minimization of (2.31) is typically achieved using the Levenberg-Marquardt algorithm.

2.3.2 Solving the Planar Calibration Problem

The methods used to extract internal camera parameters closely resemble those used for estimation of homography matrices. First, by introducing a matrix $\boldsymbol{B} = \boldsymbol{K}^{-\mathsf{T}}\boldsymbol{K}^{-1}$ equations (2.24) and (2.25) are rewritten as

$$\boldsymbol{h}_1^{\mathsf{T}}\boldsymbol{B}\boldsymbol{h}_2 = 0 \tag{2.32}$$

$$\boldsymbol{h}_1^{\mathsf{T}}\boldsymbol{B}\boldsymbol{h}_1 - \boldsymbol{h}_2^{\mathsf{T}}\boldsymbol{B}\boldsymbol{h}_2 = 0\,. \tag{2.33}$$

Note that the 3×3 matrix $\boldsymbol{B}$ is symmetric, thus the 6 elements located on and above the diagonal fully define $\boldsymbol{B}$. Let $\boldsymbol{b}$ be a column vector containing the 6 elements defining $\boldsymbol{B}$ scanned row-wise, i.e.,

$$\boldsymbol{b} = (b_{11}, b_{12}, b_{13}, b_{22}, b_{23}, b_{33})^{\mathsf{T}}\,. \tag{2.34}$$

Writing

$$\boldsymbol{h}_i^{\mathsf{T}}\boldsymbol{B}\boldsymbol{h}_j = \boldsymbol{m}_{ij}^{\mathsf{T}}\boldsymbol{b}\,, \tag{2.35}$$

where

$$\boldsymbol{m}_{ij} = (h_{i1}h_{j1},\, h_{i1}h_{j2} + h_{i2}h_{j1},\, h_{i2}h_{j2},\, h_{i3}h_{j1} + h_{i1}h_{j3},\, h_{i3}h_{j2} + h_{i2}h_{j2},\, h_{i3}h_{j3})^{\mathsf{T}}\,, \tag{2.36}$$

allows equations (2.32) and (2.33) to be written in matrix form

$$\begin{bmatrix} \boldsymbol{m}_{12}^{\mathsf{T}} \\ (\boldsymbol{m}_{11} - \boldsymbol{m}_{22})^{\mathsf{T}} \end{bmatrix} \boldsymbol{b} = \boldsymbol{0}\,. \tag{2.37}$$

Equation (2.37) contains a pair of per-view constraints on the camera intrinsics (implicitly, by constraining the elements of $\boldsymbol{b}$). Accumulating such pairs of constraints from all n views in a $2n \times 6$ matrix $\boldsymbol{M}$ results in a lest-squares problem given by a system of homogeneous equations

$$\boldsymbol{M}\boldsymbol{b} = \mathbf{0}\,. \tag{2.38}$$

The solution method involves the SVD of $\boldsymbol{M}$ and is analogous to the way homography matrices are found from (2.29) (refer to the previous section for details). In general, 3 or more views are needed to compute the elements of $\boldsymbol{b}$ up to a scale λ (having fixed the last element at 1). Assuming a skewless camera, i.e., $s = 0$, an additional equation $(0, 1, 0, 0, 0, 0)\boldsymbol{b} = 0$ is added to the system in (2.38). In case of square pixels, i.e., $f_x = f_y$, another constraint $(1, 0, -1, 0, 0, 0)\boldsymbol{b} = 0$ is included. An arbitrary aspect ratio f_x/f_y can be forced in a similar manner. If either the former or the latter is true, as little as 2 views of the calibration object are sufficient to recover the camera intrinsics. Once the solution vector $\boldsymbol{b}$ is known, the camera parameters are computed as follows:

$$p_y = (b_{12}b_{13} - b_{11}b_{23})/(b_{11}b_{22} - b_{12}^2) \tag{2.39}$$

$$\lambda = b_{33} - [b_{13}^2 + p_y(b_{12}b_{13} - b_{11}b_{23})]/b_{11} \tag{2.40}$$

$$f_x = \sqrt{\lambda/b_{11}} \tag{2.41}$$

$$f_y = \sqrt{\lambda b_{11}/(b_{11}b_{22} - b_{12}^2)} \tag{2.42}$$

$$s = -b_{12}f_x^2 f_y/\lambda \tag{2.43}$$

$$p_x = sp_y/f_x - b_{13}f_x^2/\lambda\,, \tag{2.44}$$

From (2.19), the corresponding orientation parameters are

$$\boldsymbol{r}_1 = \boldsymbol{K}^{-1}\boldsymbol{h}_1 \tag{2.45}$$

$$\boldsymbol{r}_2 = \boldsymbol{K}^{-1}\boldsymbol{h}_2 \tag{2.46}$$

$$\boldsymbol{t} = \boldsymbol{K}^{-1}\boldsymbol{h}_3\,, \tag{2.47}$$

and, since the columns of $\boldsymbol{R}$ are orthogonal,

$$\boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2\,. \tag{2.48}$$

Once the initial estimates of the intrinsic parameters $\boldsymbol{K}$ and orientation parameters associated with each view $\{\boldsymbol{R}_j, \boldsymbol{t}_j \mid 1 \leq j \leq n\}$ are computed using equations (2.39) - (2.48), the Levenberg-Marquardt algorithm is applied to refine the parameters by minimizing the total reprojection error, i.e., is the sum of Euclidean distances between the measured image coordinates and the coordinates obtained by projecting the corresponding model points into each view according to the currently known camera parameters. The total reprojection error is is given by

$$\sum_i \sum_j \|\boldsymbol{x}_{ij} - f(\boldsymbol{X}_i; \boldsymbol{K}, \boldsymbol{R}_i, \boldsymbol{t}_i)\|^2\,, \tag{2.49}$$

where the function $f(\boldsymbol{X}_i; \boldsymbol{K}, \boldsymbol{R}_j, \boldsymbol{t}_j) = \boldsymbol{K}\boldsymbol{R}_j[\boldsymbol{I} \mid \boldsymbol{t}_j]\boldsymbol{X}_j$ represents the projection of the i-th model point onto the image plane of the j-th view using the initial estimates of the camera parameters $\boldsymbol{K}$, $\boldsymbol{R}_j$, and $\boldsymbol{t}_j$. To enable easier evaluation of the derivatives of the projection function with respect to rotation parameters, each rotation matrix $\boldsymbol{R_j}$ is parametrized by a minimal rotatation vector $\boldsymbol{v}_j$, whose direction represents the axis of rotation, and whose magnitude encodes the angle of of rotation. The rotation

matrix and the minimal rotation vector are related by the Rodrigues' formula:

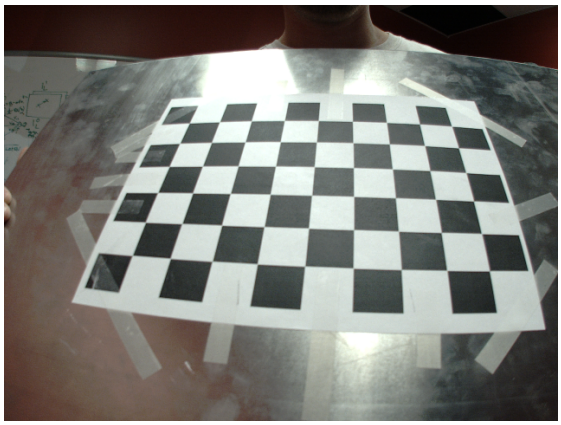
$$\boldsymbol{R} = \cos(\theta)\boldsymbol{I} + (1 - \cos(\theta))\boldsymbol{v}\boldsymbol{v}^{\mathsf{T}} + \sin(\theta)[\boldsymbol{v}]_{\times}\,, \tag{2.50}$$

where $\theta = \|\boldsymbol{v}\|$ and $[\boldsymbol{v}]_{\times}$ is a skew-symmetric matrix representing the cross product with $\boldsymbol{v}$, i.e.,

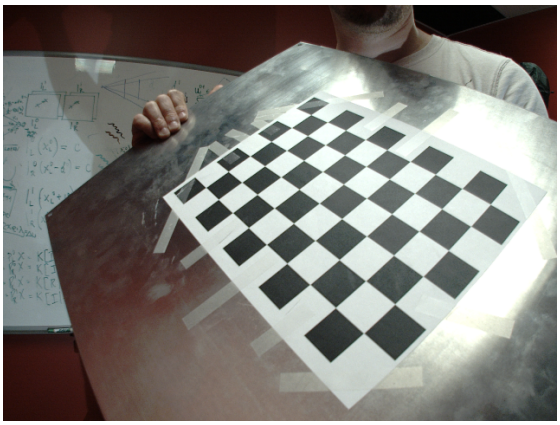
$$[\boldsymbol{v}]_{\times} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & a_1 & 0 \end{bmatrix}. \tag{2.51}$$

Note that the effects of image distortion were disregarded in the description of the camera calibration. In practice, estimation of distortion parameters is included in optimization of camera parameters, in that the distortion correction, as given by equations (2.15) and (2.16), is applied to the coordinates of projected points prior to computing the reprojection error. Starting with distortion coefficients set to zeros, the coefficients are updated together with the camera parameters in successive iterations of the optimization. Furthermore, note that since the solutions found in the homography estimation stage are not exact, the rotation matrices computed for every view will not, in general, satisfy the properties of a true rotation matrix. To address this, the author of [30] sugges performing the SVD of the rotation estimate $\boldsymbol{R}$, i.e., $\boldsymbol{R} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\mathsf{T}}$, and computing $\boldsymbol{R}' = \boldsymbol{U}\boldsymbol{V}^{\mathsf{T}}$, which is a true rotation matrix and also the closest approximation of $\boldsymbol{R}$ according to the Frobenius matrix norm.

Results of camera calibration using the planar technique are given in Figure 2.4 in the form of a 3D plot showing the orientations of the calibration object extracted from a set of 5 views; the views are included for reference. In the next section, a setup of

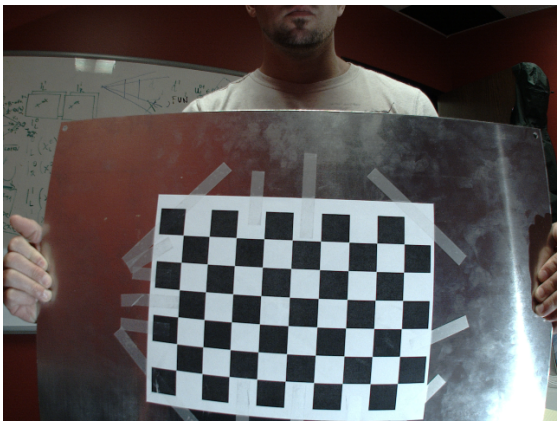


(a) View 1



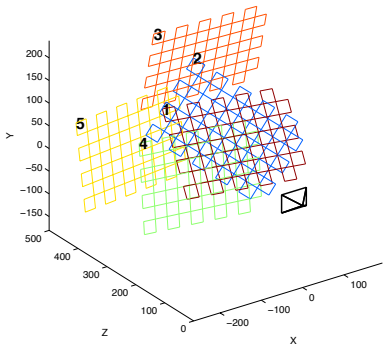
(b) View 2

(c) View 3



(d) View 4

(e) View 5



(f) Pattern orientations

Figure 2.4: Results of camera calibration using the planar technique: (a) - (e) views of the calibration object, (f) orientations of the calibration object recovered from each view (relative to the camera coordinate system).

two calibrated pinhole cameras is examined and geometric constraints are established that govern the geometry of the two views.

2.4 Epipolar Geometry

Epipolar geometry, the geometry of two views, or stereo geometry, describes intrinsic relations between geometric entities in the 3D world coordinate space and their corresponding 2D projections in a pair of cameras. These relations, which are induced solely by the internal parameters and relative orientation of the cameras, deliver constraints on the positions of corresponding image points.

To demonstrate the principles of epipolar geometry, a configuration of two cameras shown in Figure 2.5 is considered, where both cameras observe a point of interest $\boldsymbol{X}$ in the world coordinate space. The line joining the two camera centers $\boldsymbol{C}$ and $\boldsymbol{C}'$ is commonly referred to as a *baseline*. The points of intersection of the baseline and the image planes are known as *epipoles*, and are labeled as $\boldsymbol{e}$ and $\boldsymbol{e}'$ in Figure 2.5. An epipole can be though of as an image of one camera's center in the view of the other camera. Any plane that contains the baseline is called an *epipolar plane*. Individual epipolar planes are defined by the observed world points and the baseline. An epipolar plane intersects the image planes at *epipolar lines*; the families of epipolar lines in both view pass through the epipoles.

The epipolar lines are particularly important in two-view geometry, since they define domains of correspondence between image coordinates. Specifically, if $\boldsymbol{x}$ is the projection of the world point $\boldsymbol{X}$ in one view, the location of the projection $\boldsymbol{x}'$ of the

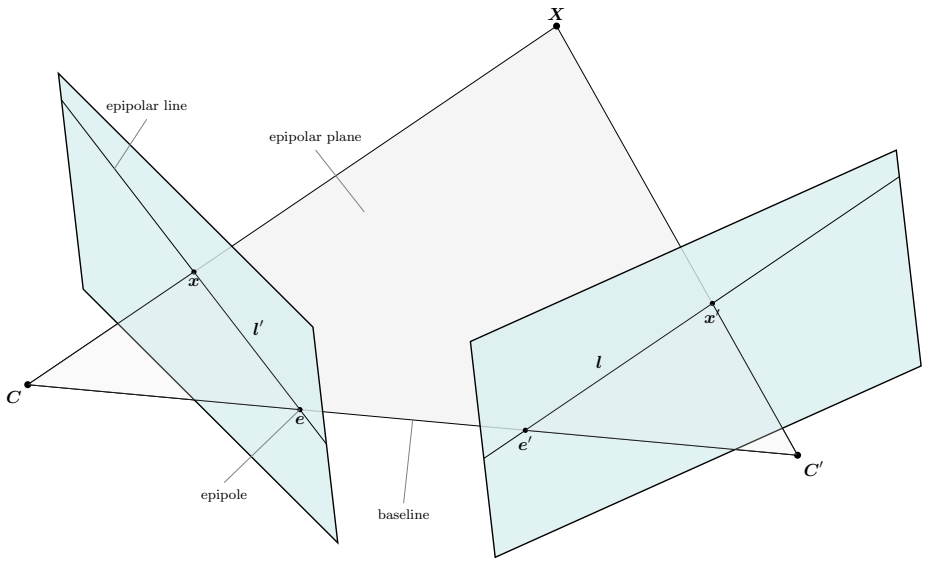


Figure 2.5: The geometry of two views.

same world point in the other view is restricted to the corresponding epipolar line $\boldsymbol{l}'$. In the following, two key entities of epipolar geometry are derived that encapsulate the relationships between image coordinates and epipolar lines. These entities are known as the *fundamental matrix* and the *epipolar matrix*.

2.4.1 The Fundamental Matrix

In Figure 2.5, the image point $\boldsymbol{x}$ in the first view is transferred into a point $\boldsymbol{x}'$ on the epipolar line $\boldsymbol{l}'$ in the other view via the epipolar plane associated with the world point $\boldsymbol{X}$. Equivalently, there exist a mapping between the image point $\boldsymbol{x}$ and the epipolar line $\boldsymbol{l}'$ that can be represented by the vector-matrix multiplication

$$\boldsymbol{l}' = \boldsymbol{F}\boldsymbol{x}\,, \tag{2.52}$$

where the 3×3 matrix $\boldsymbol{F}$ is called the *fundamental matrix*. The fundamental matrix can be algebraically derived by back-projecting a ray from the camera center $\boldsymbol{C}$

through the image point $\boldsymbol{x}$ [35]. The family of world points that lie on the ray is given by the parametric equation

$$\boldsymbol{X}(\lambda) = \boldsymbol{P}^{+}\boldsymbol{x} + \lambda \boldsymbol{C}\,, \tag{2.53}$$

where λ is a scalar parameter and $\boldsymbol{P}^{+}$ is the pseudo-inverse of the camera matrix $\boldsymbol{P}$. Two points that lie on the ray are considered: the camera center $\boldsymbol{C}$ (at $\lambda = \infty$) and the point $\boldsymbol{P}^{+}\boldsymbol{x}$ (at $\lambda = 0$). These points are imaged by the second camera as $\boldsymbol{P}'\boldsymbol{C}$ (the epipole $\boldsymbol{e}'$) and $\boldsymbol{P}'\boldsymbol{P}^{+}\boldsymbol{x}$, respectively. The epipolar line $\boldsymbol{l}'$ joins the two points $\boldsymbol{P}'\boldsymbol{C}$ and $\boldsymbol{P}'\boldsymbol{P}^{+}\boldsymbol{x}$, i.e.,

$$\boldsymbol{l}' = (\boldsymbol{P}'\boldsymbol{C}) \times (\boldsymbol{P}'\boldsymbol{P}^{+}\boldsymbol{x}) = \boldsymbol{e}' \times (\boldsymbol{P}'\boldsymbol{P}^{+}\boldsymbol{x}) = [\boldsymbol{e}']_{\times}\boldsymbol{P}'\boldsymbol{P}^{+}\boldsymbol{x}\,. \tag{2.54}$$

In equation (2.54), the fundamental matrix takes the form

$$\boldsymbol{F} = [\boldsymbol{e}']_{\times}\boldsymbol{P}'\boldsymbol{P}^{+}\,. \tag{2.55}$$

Since $\boldsymbol{F}$ represents a mapping from a 2-dimensional coordinate space of one image to a family of epipolar lines in the other image, it must have rank 2, i.e., $\boldsymbol{F}$ is singular.

The fundamental matrix is used to express the condition necessary for image points $\boldsymbol{x}$ and $\boldsymbol{x}'$ in two views to correspond. Particularly, if $\boldsymbol{x}'$ corresponds to $\boldsymbol{x}$, then $\boldsymbol{x}'$ must lie on the epipolar line $\boldsymbol{l}'$, and thus

$$\boldsymbol{x}'^{\mathsf{T}}\boldsymbol{l}' = \boldsymbol{x}'^{\mathsf{T}}\boldsymbol{F}\boldsymbol{x} = 0\,. \tag{2.56}$$

2.4.2 The Essential Matrix

The essential matrix is a specialization of the fundamental matrix that describes the intrinsic geometry of two normalized cameras, i.e., cameras whose calibration matrices

are identity matrices. Any camera with the projection matrix $\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R} \mid \boldsymbol{t}]$ can be transformed into a normalized camera by applying an inverse of the calibration matrix to the projection matrix, such that the normalized projection matrix becomes $\boldsymbol{K}^{-1}\boldsymbol{P} = [\boldsymbol{R} \mid \boldsymbol{t}]$. Similarly, an image point $\boldsymbol{x}$ can be expressed in the coordinate system of the normalized camera by applying $\boldsymbol{K}^{-1}$, such that the corresponding normalized image point is $\hat{\boldsymbol{x}} = \boldsymbol{K}^{-1}\boldsymbol{x}$.

Now consider a pair of normalized cameras $\boldsymbol{P} = [\boldsymbol{I} \mid \boldsymbol{0}]$ and $\boldsymbol{P}' = [\boldsymbol{R} \mid \boldsymbol{t}]$. Naturally, the coordinate systems of the normalized cameras are related via rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$. Thus, for any pair of corresponding normalized image points $\hat{\boldsymbol{x}} = \boldsymbol{K}^{-1}\boldsymbol{x}$ and $\hat{\boldsymbol{x}}' = \boldsymbol{K}'^{-1}\boldsymbol{x}'$ the following holds

$$\hat{\boldsymbol{x}}' = \boldsymbol{R}(\hat{\boldsymbol{x}} - \boldsymbol{t})\,. \tag{2.57}$$

Left multiplying both sides of equation (2.57) by $\hat{\boldsymbol{x}}'^{\mathsf{T}}[\mathbf{t}]_\times$ and reversing the sides results in what is known as the epipolar constraint

$$\hat{\boldsymbol{x}}'^{T}[\boldsymbol{t}]_\times \boldsymbol{R}\hat{\boldsymbol{x}} = 0\,, \tag{2.58}$$

where $\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R}$ is the essential matrix. The fundamental and the essential matrices are related by

$$\boldsymbol{F} = \boldsymbol{K}'^{-\mathsf{T}}\boldsymbol{E}\boldsymbol{K}^{-1}\,, \tag{2.59}$$

which can be verified by substituting the above into the correspondence condition of equation (2.56), and observing that this substitution leads directly to the epipolar

constraint, i.e.,

$$\boldsymbol{x}'^T \boldsymbol{F} \boldsymbol{x} = 0 \tag{2.60}$$

$$\boldsymbol{x}'^T \boldsymbol{K}'^{-\mathsf{T}} \boldsymbol{E} \boldsymbol{K}^{-1} \boldsymbol{x} = 0 \tag{2.61}$$

$$(\boldsymbol{K}'^{-1} \boldsymbol{x}')^{\mathsf{T}} \boldsymbol{E} (\boldsymbol{K}^{-1} \boldsymbol{x}) = 0 \tag{2.62}$$

$$\hat{\boldsymbol{x}}'^T \boldsymbol{E} \hat{\boldsymbol{x}} = 0\,. \tag{2.63}$$

2.4.3 Fundamental/Essential Matrix in Structure Recovery

The fundamental matrix (and the essential matrix, as one might be obtained from the other given the calibration matrices) are of high importance to the problems of visual correspondence and two-view scene structure reconstruction. In particular, the epipolar lines computed from the fundamental matrix can be used to guide matching of image features when searching for image correspondences. Furthermore, it is possible to extract camera projection matrices from the fundamental matrix. The projection matrices, combined with correspondence data, allow for scene structure reconstruction through recovery of the world coordinates of individual scene points.

The fundamental matrix, camera matrices and scene points are jointly estimated from image correspondences, each of the form $\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}'_i$, using Levenberg-Marquardt's iterative minimization scheme. The approach is outlined below:

1. An estimate $\hat{\boldsymbol{F}}$ of the fundamental matrix is computed from a set of at least 8 correspondences using the DLT algorithm. Precisely, a constraint on the elements of $\hat{\boldsymbol{F}}$ is obtained from equation (2.56) for each correspondence; all such constraints are combined to form a system of equations, and $\hat{\boldsymbol{F}}$ is found as the

least-squares solution to this system. Since the solution $\hat{\boldsymbol{F}}$ is non-singular, and thus does not satisfy the properties the fundamental matrix, a true fundamental matrix is determined by computing the closest singular approximation to $\hat{\boldsymbol{F}}$ in terms of the Frobenius norm. This approximation can obtained from the SVD of $\hat{\boldsymbol{F}}$ [18]. Note that normalization of point data is required for numerical stability of the DLT algorithm.

2. Camera matrices $\boldsymbol{P} = [\boldsymbol{I} \mid \boldsymbol{0}]$ and $\boldsymbol{P}' = [[\boldsymbol{e}']_\times \hat{\boldsymbol{F}} \mid \boldsymbol{e}']$ are formed, then for every pair $\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_i'$ the corresponding world point $\boldsymbol{X}_i$ is determined. The projection equation $\boldsymbol{x}_i = \boldsymbol{P}\boldsymbol{X}_i$ implies $\boldsymbol{x}_i \times (\boldsymbol{P}\boldsymbol{X}_i) = 0$, which gives rise to equations

$$x_i(\boldsymbol{p}_3^\mathsf{T}\boldsymbol{X}_i) - (\boldsymbol{p}_1^\mathsf{T}\boldsymbol{X}_i) = 0 \tag{2.64}$$

$$y_i(\boldsymbol{p}_3^\mathsf{T}\boldsymbol{X}_i) - (\boldsymbol{p}_2^\mathsf{T}\boldsymbol{X}_i) = 0 \tag{2.65}$$

$$x_i(\boldsymbol{p}_2^\mathsf{T}\boldsymbol{X}_i) - y_i(\boldsymbol{p}_1^\mathsf{T}\boldsymbol{X}_i) = 0\,, \tag{2.66}$$

where $\boldsymbol{p}_k^\mathsf{T}$ denotes the k-th row of $\boldsymbol{P}$. Discarding the third, linearly dependent equation, and adding analogous equations from the other view results in a homogeneous system of equations

$$\begin{bmatrix} x_i\boldsymbol{p}_3^\mathsf{T} - \boldsymbol{p}_1^\mathsf{T} \\ y_i\boldsymbol{p}_3^\mathsf{T} - \boldsymbol{p}_2^\mathsf{T} \\ x_i'\boldsymbol{p}_3^\mathsf{T} - \boldsymbol{p}_1^\mathsf{T} \\ y_i'\boldsymbol{p}_3^\mathsf{T} - \boldsymbol{p}_2^\mathsf{T} \end{bmatrix} \boldsymbol{X}_i = 0\,, \tag{2.67}$$

to which a least-squares solution is found. In the geometric sense, solving the system in (2.67) is equivalent to backprojecting rays from camera centers through the corresponding image points and finding their point of intersection

or, in the absence of thereof (due to inexact image measurements or projection matrices), a point with a minimal distance to both lines. This process is known as *triangulation* of world coordinates from image correspondences. Once the coordinates of $\boldsymbol{X}_i$ are known, auxiliary image points $\hat{\boldsymbol{x}}_i = \boldsymbol{P}\boldsymbol{X}_i$ and $\hat{\boldsymbol{x}}'_i = \boldsymbol{P}'\boldsymbol{X}_i$ are computed that are consistent with the fundamental matrix $\hat{\boldsymbol{F}}$.

3. The fundamental matrix $\hat{\boldsymbol{F}}$, camera matrices $\boldsymbol{P}$ and $\boldsymbol{P}'$, and the set of world point coordinates $\{\boldsymbol{X}_i\}$ are updated by minimizing the total reprojection error in both views, measured as the sum of geometric distances between each pair of points $(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i)$ and $(\boldsymbol{x}'_i, \hat{\boldsymbol{x}}'_i)$.

While the ability to compute epipolar lines allows for reduction of correspondence search domains in visual correspondence problems, computing general epipolar lines and searching for correspondences along such lines is still computationally expensive. Similarly, triangulation of world coordinates is a major factor of computational complexity in structure reconstruction, as the process is repeated for every image correspondence. In section 2.5, an image transformation method is discussed that can be applied in order to align the epipolar lines with the rows of the two images, effectively eliminating the need for computing general epipolar lines from the fundamental matrix, and confining the corresponding image points within the corresponding image scanlines. What is important is that this transformation also leads to a simple inverse relationship between the depth of world points and the positional offsets of their projections in both views, allowing for recovery of world points associated with individual image correspondences without the use of triangulation. This relationship is derived in

section 2.6.

2.5 Stereo Image Rectification

Consider a stereo camera configuration shown in Figure 2.6. Since the two cameras are placed alongside each other, they will be correspondingly referred to as the left camera and the right camera. The left camera is assumed to be located at the origin of the world coordinate system and looking in the direction of the Z-axis. The right camera, whose internal parameters are equivalent to those of the left camera, and whose optical axis is parallel to the optical axis of the left camera, is displaced by b units along the X-axis. A setup of two horizontally displaced, parallel cameras is known as an ideal stereo configuration or an ideal stereo camera setup.

It can be shown that, in the ideal stereo configuration, the epipolar lines in both images are collinear and run in parallel with the horizontal axis of the images. In such case, the translation vector that relates the right and the left camera is $\boldsymbol{t} = (-b, 0, 0)^{\mathsf{T}}$, and the rotation matrix is an identity matrix, i.e., $\boldsymbol{R} = \boldsymbol{I}$. Thus, the corresponding epipolar matrix is

$$\boldsymbol{E} = [\boldsymbol{t}]_{\times}\boldsymbol{R} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & b \\ 0 & -b & 0 \end{bmatrix}. \tag{2.68}$$

Let $\boldsymbol{x}_L = (x, y, f)^{\mathsf{T}}$ be the projection of some world point $\boldsymbol{X}$ on the image plane of the left camera, and let $\boldsymbol{x}_R = (x', y', f)$ be the projection of $\boldsymbol{X}$ on the image plane of

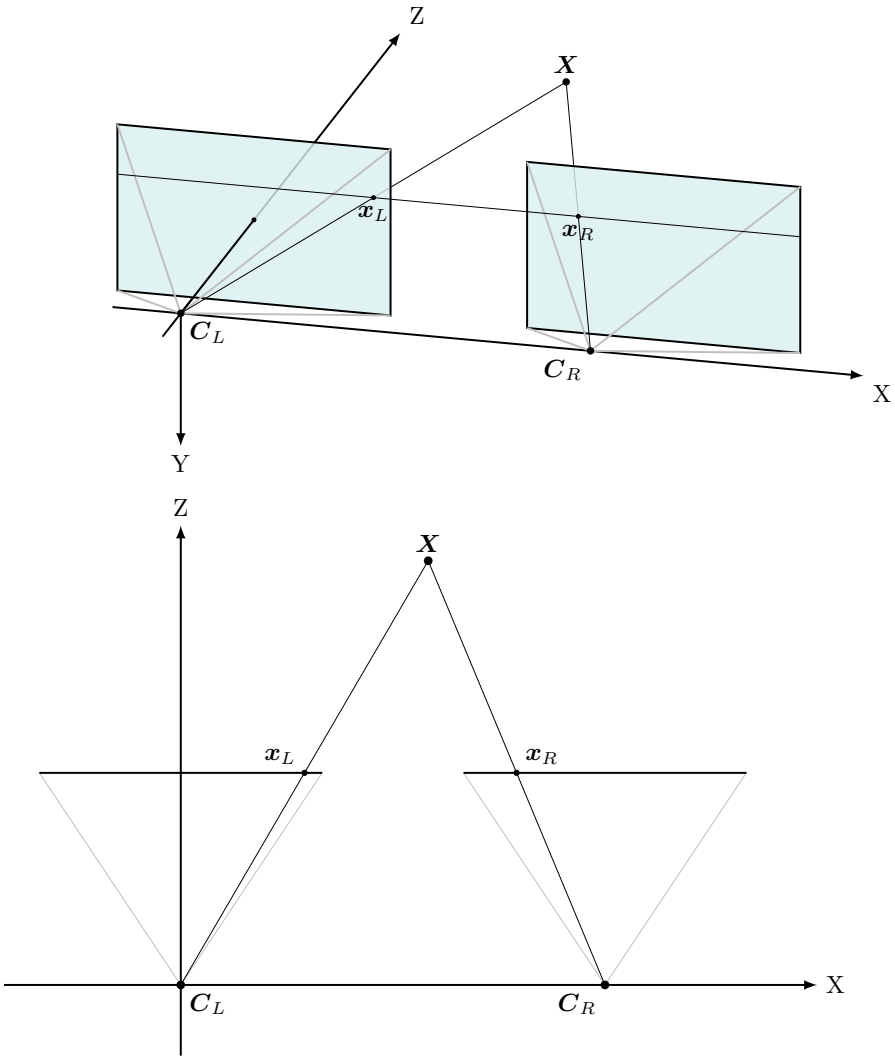


Figure 2.6: An ideal stereo camera configuration.

the right camera. From the epipolar constraint $\boldsymbol{x}_R^{\mathsf{T}}\boldsymbol{E}\boldsymbol{x}_L = 0$, we have

$$\begin{bmatrix} x' & y' & f \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & b \\ 0 & -b & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ f \end{bmatrix} = 0\,, \tag{2.69}$$

hence

$$\begin{bmatrix} x' & y' & f \end{bmatrix} \begin{bmatrix} 0 \\ bf \\ -by \end{bmatrix} = 0\,, \tag{2.70}$$

which implies $y = y'$. Since the above holds for any pair of horizontal coordinates x and x', given that $\boldsymbol{x}_L$ and $\boldsymbol{x}_R$ are indeed the projections of the same world point, the epipolar lines must coincide with the rows of both images.

Placing a pair of cameras side-by-side will generally result in physical misalignment, i.e., the optical axes of the cameras will not be parallel. This misalignment, however, can be corrected for in a process known as *stereo image rectification* that transforms a setup of two cameras into an ideal stereo configuration. Numerous methods for stereo image rectification have been proposed [15, 36–38]. In the following, the method of [38] is outlined that operates by orienting the cameras such that the the line through both camera centers becomes the horizontal axis, and then reprojecting the images onto a common image plane which is parallel to the baseline. Camera projection matrices of the corresponding ideal stereo configuration are also determined by this method.

Assume that the projection matrices along with the associated internal and external parameters $\boldsymbol{P}_L = \boldsymbol{K}_L\boldsymbol{R}_L[\boldsymbol{I} \mid \boldsymbol{C}_L]$ and $\boldsymbol{P}_R = \boldsymbol{K}_R\boldsymbol{R}_R[\boldsymbol{I} \mid \boldsymbol{C}_R]$ are known prior to image

rectification. It is recommended that a rigid transformation composed of rotation and translation is applied to both cameras that places the optical center of the left camera at the origin, and aligns its principal axis with the Z-axis of the world coordinate system; this way $\boldsymbol{P}_L = \boldsymbol{K}_L[\boldsymbol{I} \mid \boldsymbol{0}]$. The rectification procedure involves three steps:

1. First, a rotation matrix $\boldsymbol{R}^*$ is found that aligns the horizontal axis of the left camera with the baseline. Let vectors $\boldsymbol{r}_1^*$, $\boldsymbol{r}_2^*$, and $\boldsymbol{r}_3^*$ be the rows of $\boldsymbol{R}^*$. A unit vector obtained by normalizing the translation vector that relates both cameras becomes the first row of $\boldsymbol{R}^*$, i.e.,

$$\boldsymbol{r}_1^* = \frac{\boldsymbol{t}}{\|\boldsymbol{t}\|} \,. \tag{2.71}$$

   The vector $\boldsymbol{r}_2^*$ is computed as

$$\boldsymbol{r}_2^* = \boldsymbol{k} \times \boldsymbol{r}_1 \,, \tag{2.72}$$

   where $\boldsymbol{k}$ is an arbitrary unit vector that places the new Y-axis on a plane orthogonal to $\boldsymbol{r}_1$. The vector $\boldsymbol{k}$ can be chosen as a unit vector in the direction of the original Z-axis of the left camera, i.e., $\boldsymbol{k} = (0, 0, 1)^\mathsf{T}$. In this case, the orientation of the new Y-axis is

$$\boldsymbol{r}_2^* = \frac{1}{\sqrt{t_x^2 + t_y^2}}(-t_y, t_x, 0) \,. \tag{2.73}$$

   Seeing that the last row of $\boldsymbol{R}^*$ must be orthogonal to the other two, it follows that

$$\boldsymbol{r}_3^* = \boldsymbol{r}_1 \times \boldsymbol{r}_2 \,. \tag{2.74}$$

2. A common camera matrix $\boldsymbol{K}$ is computed by averaging the two camera matrices $\boldsymbol{K}_L$ and $\boldsymbol{K}_R$, i.e., $\boldsymbol{K} = (\boldsymbol{K}_L + \boldsymbol{K}_R)/2$, and the projection matrices of the

rectified stereo setup are computed as

$$\boldsymbol{P}_L^* = \boldsymbol{K}[\boldsymbol{R}^* \mid -\boldsymbol{R}^*\boldsymbol{C}_L] \tag{2.75}$$

$$\boldsymbol{P}_R^* = \boldsymbol{K}[\boldsymbol{R}^* \mid -\boldsymbol{R}^*\boldsymbol{C}_R]\,. \tag{2.76}$$

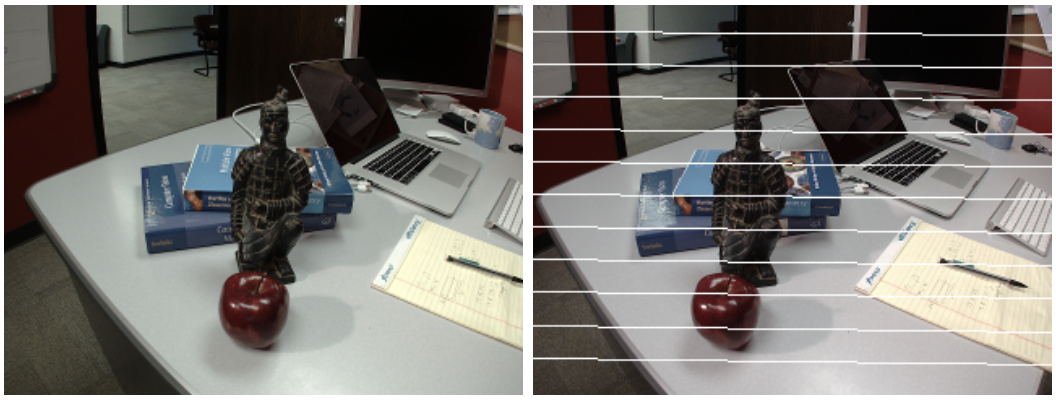
3. In order for the matrices in (2.75) and (2.76) to accurately model the projection, the two images need to be reprojected onto a plane that contains the image planes of the corresponding ideal stereo configuration. This is accomplished using a pair homography matrices $\boldsymbol{H}_L$ and $\boldsymbol{H}_R$ given by

$$\boldsymbol{H}_L = \boldsymbol{M}_L^*\boldsymbol{M}_L^{-1} \tag{2.77}$$

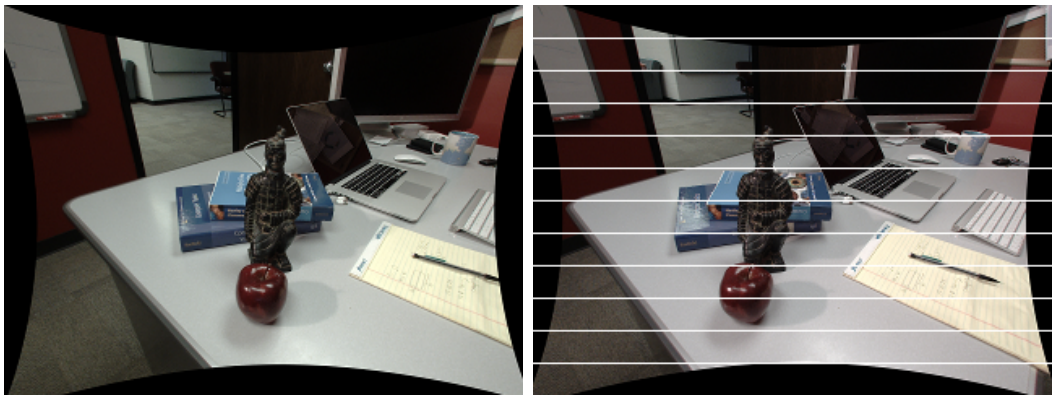
$$\boldsymbol{H}_R = \boldsymbol{M}_R^*\boldsymbol{M}_R^{-1}\,, \tag{2.78}$$

where $\boldsymbol{M}_L$ and $\boldsymbol{M}_R$ are the 3×3 left submatrices of the original projection matrices $\boldsymbol{P}_L$ and $\boldsymbol{P}_R$, and similarly, $\boldsymbol{M}_L^*$ and $\boldsymbol{M}_R^*$ are the 3×3 left submatrices of the rectified projection matrices $\boldsymbol{P}_L^*$ and $\boldsymbol{P}_R^*$.

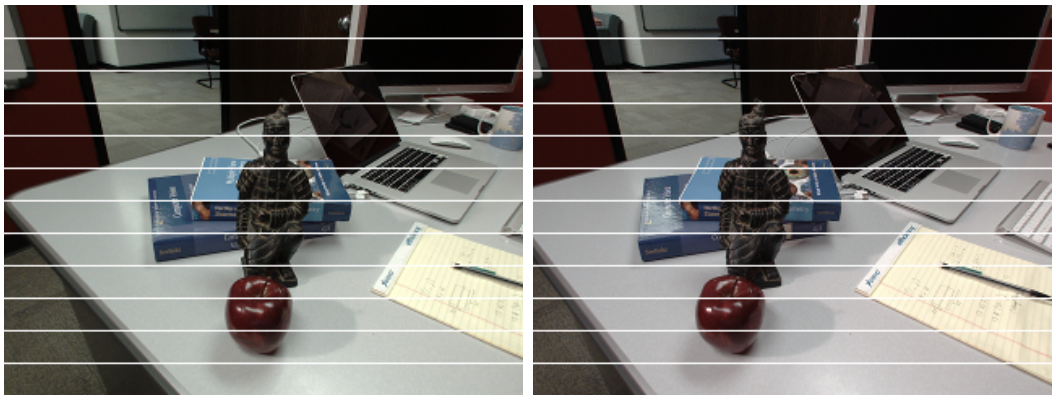
The transformations $\boldsymbol{H}_L$ and $\boldsymbol{H}_R$ achieve rectification by warping the images in a way that enforces parallelism of the epipolar lines and image rows, and aligning the images such that the corresponding epipolar lines become coincident. Note that it is necessary to perform distortion correction on the original images in order for $\boldsymbol{H}_L$ and $\boldsymbol{H}_R$ to be a valid pair of rectifying transforms. The combination of distortion correction and rectification will impact the size of the images, resulting in under- or overfilling of the rectangles bounding the original images. Consequently, it will not be possible to match some extents of rows located next to the boundaries of the rectified images. In practice, additional scaling and cropping operations are applied to the



(a) Original images



(b) Rectified/undistorted images



(c) Rectified/undistorted images (scaled and cropped)

Figure 2.7: The effects of image rectification. Epipolar lines are drawn in white.

rectified images in order to eliminate such rows and to preserve the original image sizes. The effects of image rectification are shown in Figure 2.7.

2.6 Depth as an Inverse of Stereo Disparity.

The rectification process transforms an arbitrary stereo camera configuration into an ideal stereo configuration, where the intrinsic parameters of the cameras are consistent, the optical axes of the cameras are parallel, and the horizontal axes of both camera coordinate systems contain the baseline. As described in section 2.5, rectification is acheived by reprojecting the two images onto a common image plane parallel to the baseline, such that epipolar lines of the resampled images are coincident and run in parallel to the x-axis, and consequently the y-coordinates of the corresponding points must be identical.

In the following, a relationship is derived between the depth (Z-coordinate) of scene points and the *stereo disparity*, i.e., the positional offset relating the corresponding image points in two views. The ideal stereo configuration of Figure 2.6 is considered, where the optical center of the left camera oriented along the Z-axis is at the origin of the world coordinate system, and a parallel, internally equivalent right camera is displaced by b units along the horizontal axis.

The action of the left camera on a world point $\boldsymbol{X} = (X, Y, Z)^{\mathsf{T}}$ given in the left

camera's coordinates is modeled by the projection equation

$$\boldsymbol{x}_L = \boldsymbol{K}\boldsymbol{X} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} fX + p_x Z \\ fY + p_y Z \\ Z \end{bmatrix} = \begin{bmatrix} \frac{fX}{Z} + p_x \\ \frac{fY}{Z} + p_y \\ 1 \end{bmatrix} . \tag{2.79}$$

Similarly, the very same world point $\boldsymbol{X} = (X - b, Y, Z)^\mathsf{T}$, this time expressed in the coordinate system of the right camera, is mapped to a pixel location $\boldsymbol{x}_R$ given by

$$\boldsymbol{x}_R = \boldsymbol{K}\boldsymbol{X} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X - b \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \frac{f(X-b)}{Z} + p_x \\ \frac{fY}{Z} + p_y \\ 1 \end{bmatrix} . \tag{2.80}$$

The two image points $\boldsymbol{x}_L = (x_L, y)$ and $\boldsymbol{x}_R = (x_R, y)$ are related via the stereo disparity d given by

$$d = x_L - x_R \, . \tag{2.81}$$

Substituting $x_L = \frac{fX}{Z} + p_x$ and $x_R = \frac{f(X-b)}{Z} + p_x$ into equation (2.81) yields

$$d = \frac{fX}{Z} + p_x - \frac{f(X-b)}{Z} - p_x \tag{2.82}$$

$$Z \cdot d = fX - f(X-b) \tag{2.83}$$

$$Z = \frac{f \cdot b}{d} \, , \tag{2.84}$$

which suggest that the depth Z and the disparity d are inversely proportional; the corresponding x- and y-coordinates are

$$X = \frac{b(x_L - p_x)}{d} \tag{2.85}$$

$$Y = \frac{b(y_L - p_y)}{d} \, . \tag{2.86}$$

Thus, given a triplet (x_L, y, d), i.e., a pair of image coordinates (x_L, y) in the view of the left camera, and the associated disparity d, the corresponding world point $\boldsymbol{X}$ can be computed as

$$\boldsymbol{X} = \frac{b}{d} \begin{bmatrix} x - p_x \\ y - p_y \\ f \end{bmatrix} . \tag{2.87}$$

Equation (2.87) is sometimes written in matrix form

$$\boldsymbol{X} = \begin{bmatrix} \frac{b(x-p_x)}{d} \\ \frac{b(y-p_y)}{d} \\ \frac{fb}{d} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -p_x \\ 0 & 1 & 0 & -p_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{1}{b} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = \boldsymbol{Q} \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} , \tag{2.88}$$

where the matrix $\boldsymbol{Q}$ represents the disparity-to-depth mapping.

CHAPTER 3

Visual Correspondence

Since computer vision came to existence in the mid-1960's, the term stereo matching functioned as a common name for a broad class of problems aimed at identifying correspondences between points in pairs of images of the same scene. In the 1980's, the definition of stereo matching was narrowed down to the specific problem of finding the displacements relating pixels in images obtained using a setup of two parallel, horizontally offset digital cameras. At the same time, two related problems emerged known under the names of feature detection and matching, and optical flow estimation that, together with stereo matching, constitute the modern-day research field of visual correspondence. In this chapter, a taxonomy of visual correspondence problems is established and their formulations and solution methods are reviewed.

Modern formulations of the visual correspondence problems are based on the work of Lucas and Kanade [11] originally focused on finding image features that can be reliably tracked between frames in a video sequence. In their paper, by many considered to the be fundamental publication in visual correspondence, the problem

of identifying correspondences was for the first time formulated as an optimization task. This formulation is now summarized. Let I_0 and I_1 be two intensity functions representing the input images that, for simplicity, are assumed to be grayscale. The image given by I_0 is often referred to as the reference or the template image and I_1 is called the matched or the target image. The notation $I_0(\boldsymbol{x})$ will be used to denote the intensity value associated with a specific pixel location $\boldsymbol{x} = (x, y)$, where the coordinates x and y are discrete.

Lukas and Kanade defined a measure of visual dissimilarity that evaluates a sum of squared pairwise intensity differences between a set of pixels $\{\boldsymbol{x}_i\}$ within an image patch centered around some pixel of interest $\boldsymbol{x}$ in the reference image, and the pixels within an equally shaped patch whose centroid is displaced by a disparity vector $\boldsymbol{d} = (d_x, d_y)^\mathsf{T}$ in the matched image, i.e.,

$$E(\boldsymbol{x}, \boldsymbol{d}) = \sum_i w(\boldsymbol{x}_i) \Big[I_1(\boldsymbol{x}_i + \boldsymbol{d}) - I_0(\boldsymbol{x}_i) \Big]^2, \tag{3.1}$$

where $w(\boldsymbol{x}_i)$ is an arbitrary weighing function that assigns a weight value to every pixel $\boldsymbol{x}_i$ within the patch. Assuming that image intensities are preserved in matching features, the problem of identifying correspondences becomes the problem of finding a displacement vector that minimizes (3.1).

What differentiates feature detection and matching, stereo matching, and optical flow estimation problems from each other are the assumptions made regarding the images, disparities, rigidity of the scene, the motion of the cameras or objects in the scene, and whether correspondence search is performed for all pixels or for a subset of pixels of the reference image. These visual correspondence problems, their applications,

the type of assumptions made, and how the dissimilarity metric in (3.1) evolved in their individual formulations, are discussed in sections 3.1 – 3.3.

3.1 Feature Detection, Description, and Matching

Feature detection focuses on identifying salient visual features that repeatably appear in multiple views of a scene. This inherently sparse process is frequently combined with feature description, an operation tasked with constructing mathematical feature representations that support matching of features. An example of feature detection and matching is given in Figure 3.1. The ability to identify and match visual features is vital in applications such as registration of images, alignment of 3D models, and camera pose estimation in structure reconstruction. No assumptions are typically made regarding the images. More specifically, images acquired using different cameras are allowed and the viewpoints may differ significantly. The scene, on the other hand, is assumed to be rigid in many applications. In what follows, the processes of feature detection, description, and matching are discussed in detail.

3.1.1 Feature Detection

Remarkably, the area-based dissimilarity metric in equation (3.1) can be used to assess the uniqueness of an image patch with respect to local image displacements, i.e., small displacements within the same image. To do so, the *autocorrelation function* given by

$$E_{AC}(\boldsymbol{x}, \boldsymbol{d}) = \sum_i w(\boldsymbol{x}_i) \Big[I_0(\boldsymbol{x}_i + \boldsymbol{d}) - I_0(\boldsymbol{x}_i) \Big]^2 . \tag{3.2}$$

(a) I_0 (b) I_1 (c) Feature displacements

Figure 3.1: An example of feature detection and matching using the SURF algorithm [39]; images are provided by [12].

is evaluated and analyzed across the image for a range of local displacements. This auto-correlation function is known to have strong, dominant minima at pixel locations corresponding to corner-like structures. Strong minima of the auto-correlation function indicate high uniqueness of such structures, suggesting that corners are a feasible choice of features for detection and tracking.

The computational cost associated with evaluating the autocorrelation function using equation (3.2) is prohibitive in practical scenarios. To obtain a more tractable form, the image intensity function $I_0(\boldsymbol{x_i} + \boldsymbol{d})$ is approximated using the Taylor series as

$$I_0(\boldsymbol{x}_i + \boldsymbol{d}) \approx I_0(\boldsymbol{x}_i) + \nabla I_0(\boldsymbol{x}_i) \cdot \boldsymbol{d}\,, \tag{3.3}$$

where $\nabla I_0(\boldsymbol{x_i}) = (I_x(\boldsymbol{x_i}), I_y(\boldsymbol{x_i}))$ is the image gradient composed of partial derivatives $I_x(\boldsymbol{x_i}) = \frac{\partial I_0}{\partial x}(\boldsymbol{x}_i)$ and $I_y(\boldsymbol{x_i}) = \frac{\partial I_0}{\partial y}(\boldsymbol{x}_i)$, evaluated at pixel $\boldsymbol{x}_i$. Substituting the

approximation in (3.3) into equation (3.2) results in

$$E_{AC}(\boldsymbol{x}, \boldsymbol{d}) = \sum_i w(\boldsymbol{x}_i)\Big[I_0(\boldsymbol{x}_i + \boldsymbol{d}) - I_0(\boldsymbol{x}_i)\Big]^2 \tag{3.4}$$

$$\approx \sum_i w(\boldsymbol{x}_i)\Big[I_0(\boldsymbol{x}_i) + \nabla I_0(\boldsymbol{x}_i) \cdot \boldsymbol{d} - I_0(\boldsymbol{x}_i)\Big]^2 \tag{3.5}$$

$$= \sum_i w(\boldsymbol{x}_i)\Big[\nabla I_0(\boldsymbol{x}_i) \cdot \boldsymbol{d}\Big]^2 \tag{3.6}$$

$$= \boldsymbol{d}^\mathsf{T} \boldsymbol{A} \boldsymbol{d}\,. \tag{3.7}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \sum_i w(\boldsymbol{x}_i) I_x^2(\boldsymbol{x}_i) & \sum_i w(\boldsymbol{x}_i) I_x(\boldsymbol{x}_i) I_y(\boldsymbol{x}_i) \\ \sum_i w(\boldsymbol{x}_i) I_x(\boldsymbol{x}_i) I_y(\boldsymbol{x}_i) & \sum_i w(\boldsymbol{x}_i) I_y^2(\boldsymbol{x}_i) \end{bmatrix} \tag{3.8}$$

is known as the *autocorrelation matrix*.

In [11] and later in [40], the autocorrelation matrix was shown to deliver useful information regarding the local shape of the auto-correlation function. In particular, eigenvalue analysis of the autocorrelation matrix can be performed to identify dominant gradient directions within the image patch surrounding the pixel of interest. More specifically, the eigenvector associated with the smallest eigenvalue of $\boldsymbol{A}$ corresponds to the direction of the slowest intensity change, and similarly, the eigenvector associated with the larger of the eigenvalues corresponds to the direction of the fastest intensity change.

Shi and Tomasi [41] concluded that image patches where the intensities vary rapidly along both dominant directions, i.e., corners or other readily distinguishable patterns, produce two large eigenvalues λ_0 and λ_1. In the same work, a detector was proposed that accepts a pixel as a feature if the quantity $\min(\lambda_0, \lambda_1)$ exceeds a given threshold. The corner detector of Harris and Stephens [42] computes a corner response function

given by

$$\det(\boldsymbol{A}) - \text{trace}(\boldsymbol{A})^2 = \lambda_0\lambda_1 - \alpha(\lambda_0 - \lambda_1)^2\,, \tag{3.9}$$

which avoids direct eigenvalue decomposition of the auto-correlation matrix. Alternative corner response functions based on the auto-correlation matrix have been proposed in [43, 44].

Another category of corner detectors attempt to identify corners through direct examination of image patches surrounding a pixel of interest [45–49]. A recent approach in this category, the Features from Accelerated Segment Test (FAST) algorithm [48], considers 16 pixels forming a circle around the pixel of interest, and accepts the center pixel as a corner if the circle contains a segment of at least 9 connected pixels that are brighter or darker than the pixel in the center by a given threshold. In [49] corner detection is viewed as a classification problem and solved using decision trees that are built according to the decision rules of FAST. The local nature of patch examination, however, prevents such detectors form capturing larger-scale features.

Besides the ability to identify larger-scale features, feature detectors are expected to reliably detect the same points of interest in a set of scaled views of the scene. To address this, Mikolajczyk and Schmid [50] performed feature detection on a scale-space representation of the image [51], i.e., a pyramid of images where fine-scale information is successively suppressed through smoothing. Once the pyramid is constructed by repetitively convolving the image with the Laplacian of Gaussian (LoG) operator (a combination of smoothing and edge enhancement), the Harris corner response function is evaluated at multiple image scales and features are found as its extrema along

the scale dimension. The integration of corner detection into the multi-scale image processing framework enables larger-scale features to be captured and also delivers a way to assess the scale of individual features.

In a related work, Lowe [52] applied incremental Gaussian smoothing and subtracted intermediate results to produce a set of difference-of-Gaussian (DoG) images covering a range of image scales. The associated feature detector, coined Scale-Invariant Feature Transform (SIFT), searches for extrema in a volume of DoG images by comparing every pixel's value to its 26 immediate neighbors within a $3 \times 3 \times 3$ neighborhood (8 in the same DoG, 9 in both finer and coarser scales). In contrast to the method of Mikolajczyk and Schmid [50], extremal points identified by the SIFT detector correspond to blobs and edges, not corners; additional filtering is applied to discard keypoints located on an edge as these do not constitute distinctive features.

More recently, Bay *et al.* [39] introduced the Speeded Up Robust Features (SURF) algorithm, which employs the Hessian matrix [53] - a popular blob detector. At some scale σ obtained through repetitive Gaussian smoothing and subsampling of the input image, the Hessian matrix at pixel $\boldsymbol{x}$ is given by

$$\boldsymbol{H}(\boldsymbol{x}, \sigma) = \begin{bmatrix} L_{xx}(\boldsymbol{x}, \sigma) & L_{xy}(\boldsymbol{x}, \sigma) \\ L_{xy}(\boldsymbol{x}, \sigma) & L_{yy}(\boldsymbol{x}, \sigma) \end{bmatrix}, \tag{3.10}$$

where $L_{xx}(\boldsymbol{x}, \sigma)$, $L_{xy}(\boldsymbol{x}, \sigma)$, and $L_{yy}(\boldsymbol{x}, \sigma)$ denote the convolutions of the image with the second order derivatives of a Gaussian function with standard deviation σ; subscripts indicate the directions along which differentiation is performed. The authors proposed discrete approximations of the second order Gaussian derivatives that can be efficiently convolved with the input image by applying a series of box filters. The evaluation of

box filters was further accelerated using integral images [54]. To identify points of interest, the determinant of the Hessian matrix is computed for every pixel in every image within the pyramid, and $3 \times 3 \times 3$ pixel neighborhoods are searched for maximum values. An optional thresholding operation may be applied to the determinant values in order to control the number and quality of features found. Quadratic fitting, as suggested by Brown and Lowe [55], can be used to localize points of interest with sub-pixel and sub-scale precision.

3.1.2 Feature Description and Matching

The primary purpose of feature description is to characterize local properties of image patches surrounding the detected points of interest in a way that enables reliable matching of features. As will be discussed later, feature descriptors can optionally provide invariance to changes in illumination and in-plane image rotation at the stage of matching features. Among the existing descriptors, some also offer a degree of invariance to viewpoint changes. Scale invariance, on the other hand is achieved through scale-space processing during the feature detection stage.

Surely, a simple descriptor can be created by storing pixel intensities of a patch surrounding a point of interest. Such an approach, however, does not offer invariance to illumination changes or image rotation, and will only be viable if the viewpoint changes between the views being matched are mainly translational. A more feasible approach that handles image rotations is to assign each feature with an orientation vector and describe local image properties in a 2D coordinate frame defined with

respect to the feature orientation vector.

The feature description scheme of the SIFT algorithm [52], for instance, samples gradient magnitudes and orientations within a 16×16 region centered at feature's location at the pyramid level that corresponds to the scale at which the particular feature was found. First, a histogram of gradient orientations is created covering the range of orientations in 10-degree increments. During binning each orientation is multiplied by a weight value that is computed based on sample's distance from feature's location and its associated gradient magnitude. A dominant gradient orientation is then determined from the histogram that becomes the orientation of the feature. Next, the 16×16 region is divided into 4×4 subregions for which 8-bin local gradient orientation histograms are computed. Ultimately, the orientations within the local histograms are rotated relative to the feature's orientation and arranged into a 128-element descriptor vector. To provide invariance to illumination changes, the descriptor vector is normalized to a unit length. A modification of this scheme was proposed in [56] that uses log-polar binning when computing orientation histograms.

In a similar way, SURF's descriptors [39] use Haar wavelet responses to characterize local intensity changes around the detected keypoints; such responses are known to be invariant to illumination changes. To determine the feature's orientation, x and y components of the wavelet responses (at the appropriate scale) are Gaussian-weighted and summed within a wedge of $\frac{\pi}{3}$ radians that is rotated around the keypoint's location. The orientation associated with the maximum total response is assumed to be the feature's dominant orientation. Sums of individual responses as well as sums of their absolute values are computed along the two orthogonal directions, one of which is

defined by the orientation of the feature, within each of 16 subregions of a square patch centered at the point of interest (sized to match the feature's scale and rotated accordingly), resulting in 64-dimensional descriptors.

Several other descriptors have been developed over the course of evolution of the field of feature detection, description and matching. Notable approaches include techniques such as differential invariants [57], steerable filters [58], generalized moment invariants [59], complex filters [60], shape context [61], and spin images [62]. These approaches, however, were shown to be less distinctive and less robust to changes in viewing conditions than SIFT-like descriptors that characterize the structure of local image gradients around keypoints [56]. Simultaneously, research efforts have been invested towards making the existing descriptors invariant to significant viewpoint changes. A commonly used solution [50, 63–65] is to find the directions of the slowest and the fastest intensity changes through eigenvalue analysis of the auto-correlation matrix at the given pixel, then find an affine transformation that fits these directions and the ratio of the corresponding gradients, and apply an inverse transformation to a local image patch in order to remove affine distortion prior to computing the final descriptor.

Recently, a class of binary descriptors have been proposed as an alternative to the computationally complex feature description routines of SIFT and SURF. Binary descriptors are based on the concept of representing image patches using a number of pairwise intensity comparisons evaluated over a subset of pixels within the patch [66, 67]. A binary descriptor is thus a string of zeros and ones indicating the results of the said intensity comparisons. The sampling pattern, i.e., which points around the

feature point to include when computing the descriptor, and the pairing of sampled points for comparisons, determine the descriptor's ability to disambiguate individual features. Note that since pixels are compared in a pairwise way, Gaussian smoothing is required prior to computation of descriptors in order to mitigate the impact of image noise and compression artifacts.

The method of Calonder *et al.* [68], termed BRIEF (Binary Robust Independent Elementary Features), used random sampling patterns to form descriptor vectors with as little as 128 bits that performed well in feature recognition tests, despite the lack of rotational invariance of descriptors. A variant of BRIEF with orientation compensation was later presented in [69] along with a method for learning optimal sampling paterns from training sets of pre-matched features. The sampling pattern of BRISK (Binary Robust Invariant Scalable Keypoints) [70] is composed of concentric circles centered at the keypoint's location; sampling points are equally spaced on the circles. The amount of Gaussian smoothing applied at each sampling point is proportional to its distance to the center. Furthermore, the BRISK descriptor divides the set of sampling pairs into short pairs, i.e., pairs of points whose distance is within a given threshold, and long pairs, i.e., pairs with point-to-point distance exceeds the threshold. Gradients estimated between points in the short pairs are used to determine feature orientation, while the coordinates of points in the long pairs are rotated according to the feature orientation, then sampled and compared to form binary feature descriptors. The Fast Retina Kepoint (FREAK) detector/descriptor [71] uses a circular sampling pattern that follows the distribution of photoreceptors over the human retina, in that the density of sampling points decays exponentially along the radius of the pattern.

Unlike the approach of BRISK, the sizes (standard deviations) of Gaussians applied at sampling points are selected such that the receptive fields, i.e., circular regions across which smoothing and sampling is performed, centered at neighboring samples have significant overlap.

Once the descriptors have been extracted from the images, feature matching can be performed by evaluating pair-wise distances between the descriptor sets and choosing minimum-distance pairs as matches. When working with SIFT-like descriptors that are composed of floating-point values, Euclidean distance is typically computed, while the Hamming distance is used in case of binary descriptors. Note that the Hamming distance can be implemented using the XOR operation and bit counting, making matching of features represented using binary descriptors generally more computationally efficient than those represented using floating-point descriptors.

There are two disadvantages associated with selecting matches by finding pairs of descriptors that are similar according to some distance metric. First, the process of evaluating and comparing pair-wise distances between descriptors is known to have superlinear complexity in the number of features [44]. Second, this strategy will cause a significant number of mismatches, since a match is found for every feature, regardless whether a true match exists or not, and also since many features in one image are allowed to be matched to the same feature in the other image. The former can be addressed by reducing the dimensionality of SIFT-like feature descriptors through principal component analysis [72].

A different approach is possible using FREAK descriptors [71], where the binary comparisons are ordered in a coarse-to-fine way, from long-distance to short-distance

pairs. Likewise, matching can be performed in a coarse-to-fine way by dividing the descriptors into sections and then comparing the descriptors section-by-section, starting with the sections that contain coarse information. This way non-matches can be discarded after comparing only a subset of descriptor values. The latter of the mentioned problems can be remedied by setting a maximum descriptor distance threshold, although better results were observed if the thresholded quantity is chosen as the nearest neighbor distance ratio [56], i.e., the ratio of distances corresponding to the closest and the second closest descriptor. In [73], improvements in the number of correct matches were achieved by weighting individual components of description vectors when evaluating distances; the weight values were learned from a set of known good matches.

3.2 Stereo Matching

As indicated at the beginning of this chapter, stereo matching operates on the images acquired using a setup of two cameras situated side by side and facilitates applications that benefit from the knowledge of depth, including robotic navigation and structure reconstruction. The setup is assumed to have the properties of an ideal stereo camera configuration. Precisely, it is assumed that the cameras are identical, the optical axes of both cameras are parallel, and that the offset between the cameras is purely lateral. As such, the camera setup produces pairs of images where the epipolar lines are collinear and parallel to the x-axis, and where the disparities are horizontal. These properties are typically achieved through stereo rectification of images, since a truly

(a) Left image (b) Right image (c) Disparity map

Figure 3.2: A pair of rectified stereo images and the corresponding disparity map [12]. Brighter shades of gray correspond to larger disparities (smaller depths); pixels with unknown disparities are marked with black.

ideal stereo camera configuration is hard to build in practice. Stereo rectification is discussed in detail in section 2.5.

Unlike feature detection and matching, stereo matching is a dense problem. Given a pair of rectified images I_0 and I_1, sometimes referred to as the left image and the right image due to the specific alignment of the cameras, the goal is to recover a *disparity map* D that associates each pixel of the reference image (usually the left one) with a scalar disparity value d. For viewing purposes, disparity maps are typically translated into images by color-coding the disparities. A pair of rectified stereo images and the corresponding disparity map are shown in Figure 3.2. Note that the goal of stereo matching is ill-posed since, for a subset of pixels in the left image, there will not exist correspondences in the right image mainly due to occlusions.

The recovery of disparities can be defined as a labelling problem where each pixel is to be assigned a disparity from a set of disparity hypotheses. The set of disparity hypotheses contains subsequent integer-valued disparities ranging from 0

to a maximum disparity value d_{max} chosen based on the size of the images and the displacement between the cameras. A variant of the dissimilarity metric in (3.1) is typically used to quantify the cost of selecting individual disparity hypotheses.

Szeliski and Scharstein, the authors of an excellent survey paper on stereo matching [12], differentiate three classes of stereo matching methods based on the optimization technique used to solve the labeling problem. The classes are: (i) local methods that examine a limited scope of pixels to find locally optimal disparities, (ii) global methods that attempt to find a globally optimal and smooth disparity assignment, and (iii) a class of methods that are either hybrids of the two previous classes, or belong to neither of these. Global and local stereo algorithms are discussed in sections 3.2.1 and 3.2.2, respectively, and other approaches are covered in section 3.2.3.

3.2.1 Global Stereo Matching

Global algorithms, which are based upon the energy minimization framework, seek a disparity assignment D that minimizes a global energy function

$$E(D) = E_d(D) + \lambda E_s(D)\,, \tag{3.11}$$

where the data term $E_d(D)$ encodes the global cost associated with choosing a particular solution D, and the smoothness term $E_s(D)$ explicitly integrates disparity smoothness constraints into the minimization process; the coefficient λ balances the data and smoothness terms.

In practice, the data term $E_d(D)$ represents some cumulative measure of photometric consistency of the solution, and is often defined as a sum of per-pixel dissimilarity

metrics evaluated over the entire set of correspondences, i.e.,

$$E_d(D) = \sum_{\boldsymbol{p}} C(\boldsymbol{p}, D(\boldsymbol{p}))\,, \tag{3.12}$$

where $C(\boldsymbol{p}, D(\boldsymbol{p}))$ is an arbitrary dissimilarity metric computed between the pixel $\boldsymbol{p} = (x, y)$ in the reference image and its corresponding pixel at location $(x - D(\boldsymbol{p}), y)$ in the matched image, e.g., a sum of absolute or squared intensity differences, normalized cross-correlation, or census transform. A listing and comparison of commonly used dissimilarity metrics can be found in [74]. To enable effective evaluation of the data term in (3.12), the per-pixel dissimilarity metrics can be precomputed for the range of considered disparities and organized in a volume C, where the element $C(x, y, d)$ contains the dissimilarity metric associated with selecting the disparity d at pixel $\boldsymbol{p} = (x, y)$ in the reference image. A shorthand notation $C(\boldsymbol{p}, d)$ is often used when referring to the element $C(x, y, d)$.

The smoothness term $E_s(D)$, which plays a role in regularization of the solution, takes the form of a penalty function that discourages large variations of disparities within local pixel neighborhoods. For performance reasons, evaluation of the penalty function is often restricted to a set of all standard 4-connected pixel neighborhoods in the reference image. The standard 4-connected neighborhood of a pixel $\boldsymbol{p} = (x, y)$, denoted by $N_4(\boldsymbol{p})$, contains the four immediate neighbors of $\boldsymbol{p}$, that is, $N_4(\boldsymbol{p}) = \{(x \pm 1, y), (x, y \pm 1)\}$. Similarly to the way the data term accumulates pair-wise intensity differences, the smoothness terms accumulates penalty values arising from differences in disparities assigned to connected pixels, i.e,

$$E_s(D) = \sum_{\boldsymbol{p}} \sum_{\boldsymbol{q} \in N_4(\boldsymbol{p})} s(\boldsymbol{p}, \boldsymbol{q})\,, \tag{3.13}$$

where $s(\boldsymbol{p}, \boldsymbol{q})$ represents a penalty computed with respect to the disparities at pixels $\boldsymbol{p}$ and $\boldsymbol{q} \in N_4(\boldsymbol{p})$.

Early formulations of the smoothness term make the penalty values proportional to the disparity difference between connected pixels, e.g., by applying a linear penalty function $s(\boldsymbol{p}, \boldsymbol{q}) = |D(\boldsymbol{p}) - D(\boldsymbol{q})|P$, where P is a constant. When using the linear penalty function, however, a series of gradual disparity transitions results in the same penalty value as a single sharp transition, limiting the ability of such formulations to preserve discontinuities in the disparity map. This problem can be alleviated by using the so-called robust penalty functions [75, 76], or by including an intensity-dependent term in the penalty function to encourage discontinuities nearby object edges [77, 78]. Modern formulations adopt the truncated linear model $s(\boldsymbol{p}, \boldsymbol{q}) = \min(|D(\boldsymbol{p}) - D(\boldsymbol{q})|, T)P$, where the disparity difference is not allowed to exceed a threshold T, or the Potts model given by

$$s(\boldsymbol{p}, \boldsymbol{q}) = \begin{cases} 0 & \text{if } D(\boldsymbol{p}) = D(\boldsymbol{q}) \\ P & \text{otherwise} \end{cases}, \tag{3.14}$$

that applies a penalty value P every time a disparity difference occurs between a pair of connected pixels. Both of these have discontinuity-preserving properties.

Minimization of the energy function given in equation (3.11) is known to be an NP-hard problem [79]. Since a polynomial-time algorithm producing the optimal solution is believed not to exist, sub-optimal solutions within some small distance of the optimum are found using approximation algorithms. Historically, the optimization was achieved using techniques such as simulated annealing [80] or iterated conditional

mode [81]. Both of these techniques, however, are inefficient as they change the disparity assignment of one pixel at a time, which leads to exponential time complexity. Approximation algorithms used in recent approaches include graph cuts [79], belief propagation [82], and tree-reweighted message passing [83].

In 2001, Boykov, Veksler and Zabih [79] published a family of graph-based energy minimization algorithms with applications in image restoration and stereo correspondence problems formulated using metric or semimetric smoothness terms (e.g., the Potts model). These so-called move-making algorithms change the disparities of large number of pixels at a time using disparity re-assignment operations known as moves. Two types of moves were proposed: the α-expansion move that can change the disparity of any pixel to α, and the α-β-swap move that overwrites the disparity of some pixels initially assigned the disparity α with the value β and vice versa. Correspondingly, two variants of the iterative optimization procedure were proposed that perform a series of α-expansions or α-β-swaps (exclusively) in order to successively reduce the energy of the solution. In each iteration, the procedure cycles through disparity hypotheses or pairs of hypotheses and determines the α-expansion if individual disparities are considered, or α-β-swap if disparities are processed in pairs, that leads to the highest energy decrease. The problem of finding such a move is converted into a binary labeling problem, where the labels indicate whether the disparity assignment should be overwritten in a subset of pixels under consideration (0 - the value of a pixel remains unchanged, 1 - the value of the pixel is overwritten). A graph is then constructed that contains the subset of pixels under consideration where the edges are weighted according to the pair-wise dissimilarity metrics and smoothness metrics defined with

respect to the binary labels. It was shown in the original paper that the optimum move can be determined by solving the min-cut/max-flow problem on the resulting graph. Despite its high computational complexity, the algorithm has been demonstrated to quickly converge to low-energy solutions, achieving the most significant reductions of the global energy in the first few iterations.

In [84], a fusion move was introduced that chooses one of two options when overwriting disparities; option selection was integrated into the labeling problem. Using fusion moves, the number of graph cuts necessary to compute the labeling in each iteration of the minimization procedure was shown to grow logarithmically with the number of labels, leading to even more rapid convergence.

Sun *et al.* [82] were the first ones to apply belief propagation in the context of stereo matching. The belief propagation (BP) algorithm, originally developed as a method for performing probabilistic inference on trees and later extended to general graphs, is an iterative message passing procedure, where nodes in a graph exchange "beliefs" regarding their state, i.e., assignment to a label in a set of discrete labels, allowing every node to successively update its own beliefs based on the incoming information. In Sun's formulation, a max-product variant of belief propagation is used that approximates marginal probabilities of pixels having particular disparity assignments (conditional on the observations from neighboring pixels in a 4-connected image lattice) as a product of incoming messages. The solution, i.e., the final disparity assignment, is chosen that maximizes the marginals.

A commonly used formulation of stereo matching through belief propagation is that of Felzenszwalb and Huttenlocher [85], where the marginals are replaced with

negative log probabilities, resulting in a computationally equivalent min-sum problem. A message $m^i_{\boldsymbol{p},\boldsymbol{q}}$ sent from pixel $\boldsymbol{p}$ to pixel $\boldsymbol{q}$ in the i-th iteration of message passing is a vector, whose size corresponds to the number of disparities, such that the value at $m^i_{\boldsymbol{p},\boldsymbol{q}}(d)$ is inversely proportional to the probability of $\boldsymbol{p}$ being assigned a disparity d. Starting with all elements $m^0_{\boldsymbol{p},\boldsymbol{q}}(d)$ set to zeros, the messages passed in successive iterations of belief propagation are computed as

$$m^i_{\boldsymbol{p},\boldsymbol{q}}(d) = \min_d \left\{ s(d, D(\boldsymbol{q}))) + C(\boldsymbol{p}, d) + \sum_{\boldsymbol{r} \in N_4(\boldsymbol{p}) \setminus \boldsymbol{q}} m^{i-1}_{\boldsymbol{r},\boldsymbol{p}}(d) \right\}, \tag{3.15}$$

where $s(d, D(\boldsymbol{q})))$ is a measure of smoothness between d and the current disparity assignment at pixel $\boldsymbol{q}$, $C(\boldsymbol{p}, d)$ is the cost of assigning a disparity d to pixel $\boldsymbol{p}$, and $m^{i-1}_{\boldsymbol{r},\boldsymbol{p}}$ is the message received from pixel $\boldsymbol{r} \in N_4(\boldsymbol{p}) \setminus \boldsymbol{q}$ in the previous iteration. After all N iterations of message passing are completed, the final disparity assignment is obtained using

$$D(\boldsymbol{p}) = \operatorname*{argmin}_d \left\{ C(\boldsymbol{p}, d) + \sum_{\boldsymbol{r} \in N_4(\boldsymbol{p})} m^N_{\boldsymbol{r},\boldsymbol{p}}(d) \right\}. \tag{3.16}$$

While global convergence of the BP algorithm is not guaranteed on graphs containing cycles, e.g., images, the algorithm typically reaches a near-optimal solution after 50 - 100 iterations. The high number of iterations, which is necessary to propagate disparity information across large image regions, is considered to be a disadvantage of BP-based stereo algorithms and modifications have been proposed that address this issue. In [85], a multi-scale message passing scheme was presented that allows for accelerated exchange of disparity information. Further improvements to this scheme were reported by Yang *et al.* [86], who performed selective updating of messages in successive iterations, thus saving computations.

The tree-reweighted message passing (TRW) [83], the most recent among optimization algorithms applicable in global stereo, is closely related to the max-product belief propagation. TRW approximates and maximizes the lower bound on the energy function, which is dual to the original energy minimization problem. The image grid is first divided into trees, then, in alternating steps, the procedure performs belief propagation on trees followed by averaging of messages at nodes belonging to multiple trees. Compared to BP, the number of iterations required by the tree-reweighted message passing is usually larger. In its original form, the algorithm is not guaranteed to converge on cyclic graphs; a convergent variant of TRW was later presented in [87]. While graph cuts are known to produce lower-energy solutions than belief propagation (assuming 4-connected grids and the Potts smoothness model), in a recent study [16] convergent TRW was shown to surpass both of these methods.

Simultaneously, the key strength of the BP-based approaches is that additional disparity cues can be easily integrated into the minimization procedure by introducing extra penalty terms into the expression in (3.15). In [86, 88, 89], stereo matching was enhanced with image segmentation and plane fitting operations, resulting in disparity maps that are comparable to those obtained using graph cuts. In [90], ground control points (GCPs), i.e., sparse estimates of disparity obtained, for instance, by feature matching, were used to improve matching via graph cuts.

3.2.2 Local Stereo Matching

Local stereo algorithms avoid global energy minimization schemes in favor of the Winner-Takes-All (WTA) disparity selection framework, where disparities are extracted directly from the cost volume. Precisely, the disparity assignment at a given pixel is performed by enumerating the cost values associated with individual disparity hypotheses, and choosing the one that corresponds to the mimim cost, i.e.,

$$D(\boldsymbol{p}) = \underset{d}{\operatorname{argmin}}\, C(\boldsymbol{p}, d)\,. \tag{3.17}$$

Note that this formulation does not impose any smoothness constraints on the solution. Furthermore, operating on pixel-wise costs results in noisy, inconsistent disparity maps, as this approach fails to resolve ambiguous matches. To obtain a more robust cost measure and effectively regularize the solution, local stereo algorithms perform cost aggregation prior to selection of disparities. In the cost aggregation step, each element of the initial cost volume is recalculated as a weighted sum of the nearby elements. Cost aggregation can be viewed as a filtering operation represented by the convolution

$$C(\boldsymbol{p}, d)' = W(\boldsymbol{p}) * C(\boldsymbol{p}, d)\,, \tag{3.18}$$

where $C(\boldsymbol{p}, d)'$ is the aggregated cost, $C(\boldsymbol{p}, d)$ is the initial cost, and $W(\boldsymbol{p})$ is the support window of $\boldsymbol{p}$, i.e., an arbitrarily shaped and sized window of filter coefficients (weights) that are applied to the cost values associated with $\boldsymbol{p}$.

Much like the smoothness term in global methods, cost aggregation in local methods enables the integration of smoothness assumptions into matching. The shape, size,

and weights of the support window $W(\boldsymbol{p})$ can vary at individual pixels, hence the explicit dependency on $\boldsymbol{p}$. As will be discussed later, the selection of the size, shape, and weights of the support window is critical for preserving edges in disparity maps. Note that the support windows can be 2- or 3-dimensional: 2D windows are chosen to favor surfaces that are approximately parallel to the image planes, whereas 3D windows are chosen to favor slanted surfaces.

To alleviate the problem of disparity recovery around object boundaries, methods have been proposed for determining the optimal shape, size, and position of support windows. In [91], statistical models were devised that enabled iterative disparity estimation combined with the adjustment of the shape and size of the support windows based on local disparity and intensity variations. Boykov *et al.* [92] established a set of probabilistic tests to verify plausibility of disparity hypotheses at a given pixel and proposed constructing windows of connected pixels that support each of the plausible hypotheses; the hypothesis producing a support window with the largest size was chosen at each pixel. In a related work, Veksler [93] employed the minimum ratio cycle algorithm, previously developed as a method for image segmentation, to acquire appropriately sized and shaped support windows that are optimal with respect to the average photometric error within the window and the ratio of its perimeter and area. In [94], the same author used integral images to efficiently evaluate a set of variably sized square windows that contain (but are not necessarily centered at) the pixel of interest. Alternatively, the multiple-window methods in [95] and [78] adjust the position of the support window of predefined shape and size relative to the pixel under consideration. While these methods greatly improved the local smoothness of

disparity maps, their accuracy near disparity discontinuities was still limited.

At the same time, approaches have been developed that adapt the weight coefficients in windows of fixed shape and size. Prazdny [96] postulated that pixels assigned to the same disparity support each other. In his formulation, weight coefficients were computed as Gaussian functions of disparity differences between a pixel of interest and every pixel located within its corresponding support window. The expression for weights was also made dependent on the geometric distance between pixels, to reduce the influence of pixels with larger distances. In [97], a more general class of radial support functions was considered that are based on the certainty measures associated with disparity assignment at individual pixels. Note that these methods require initial disparity estimates and thus are sensitive to errors in thereof.

Due to difficulties in recovering disparities around object boundaries, local stereo matching methods were, for a long time, deemed incapable of achieving the accuracy of global approaches. That is until 2005, when Yoon and Kweon introduced the adaptive support weights [17]. In their approach, the weight relating a pixel of interest $\boldsymbol{p}$ and another pixel $\boldsymbol{q}$ within the support region of $\boldsymbol{p}$ was made proportional to the probability that $\boldsymbol{p}$ and $\boldsymbol{q}$ are at the same distance from the viewer and thus have the same disparity. Since this probability is not known prior to matching, an approximation was proposed that is based on the principles of shape grouping in the human visual system. Precisely, the approximation considers the geometric proximity and color similarity between pixels $\boldsymbol{p}$ and $\boldsymbol{q}$. The use of the adaptive support weights greatly improved the accuracy of disparities nearby object boundaries and allowed Yoon and Kweon's method to achieve an overall accuracy comparable to global methods.

The concept of adaptive weights was since then adopted in a myriad of methods [98–105]. Many of these methods achieve real-time operation by approximating the adaptive support weights to further reduce the computational complexity of cost aggregation. Similarly, the stereo matching method proposed in this dissertation employs an approximation of adaptive weights. This approximation is first used in the cost aggregation stage and then re-applied to refine the disparity map. Detailed description of adaptive support weights and their approximation techniques is deferred until Chapter 4 where the proposed method is derived.

3.2.3 Other Approaches

Cooperative algorithms [8, 106, 107], which date back to the times when stereo matching was still in its infancy, operate by iteratively diffusing local disparity evidence in order to achieve a global consensus on disparity assignment between the right and left view. What is interesting about such algorithms is that matching is not explicitly defined as a minimization problem. Instead, in each iteration nearby pixels interact with one another and update their disparity assignments in a way that promotes uniqueness of matches and local smoothness of disparities. Despite a recent attempt to integrate adaptive support weights into a cooperative disparity assignment scheme [108], such algorithms are currently of interest only in the historical context.

Another subclass of stereo algorithms employ a 1D optimization technique known as dynamic programming (DP) to find the disparities for independent image scanlines [109–111]. To do so, dynamic programming computes a cost matrix by evaluating some

dissimilarity metric between pairs of pixels in the two scanlines being matched, then traverses the matrix from the top-left to the bottom-right and accumulates neighboring cost values, and finally back-tracks to find the minimum-cost path that determines the optimum disparity assignment. This path is usually subject to two constraints, i.e., the continuity constraint that requires any two subsequent elements along the path to be directly connected in the cost matrix, and the monotonicity constraint that forces relative ordering of pixels to be preserved in the matched sequence. The former is necessary for dense matching and, in contrast to cooperative algorithms that favor unique matches, allows many pixels in one image to be matched to a single pixel in the other. Such behavior is seen mainly in occluded regions. The latter improves local consistency of the solution, however, may prohibit the algorithm from capturing narrow foreground objects.

Since inter-scanline consistency is not enforced, stereo matching using DP is known to produce streaking artifacts in the disparity maps. This problem has received a lot of attention and improvements have been proposed that reduce the streaking effects. One of the possible solutions is to use a more sophisticated cost metric that takes into account information from nearby scanlines [110, 112]. A different approach was taken by Veksler [113], who adapted the DP algorithm to operate on trees, precisely, the minimum spanning trees computed for image grids with edge weights assigned according to the intensity difference between connected pixels. Dynamic programming on a tree significantly reduced streaking and also improved the ability to correctly recover disparities around object boundaries when compared to the original algorithm. More recently, strong results were obtained by performing multiple passes of DP in

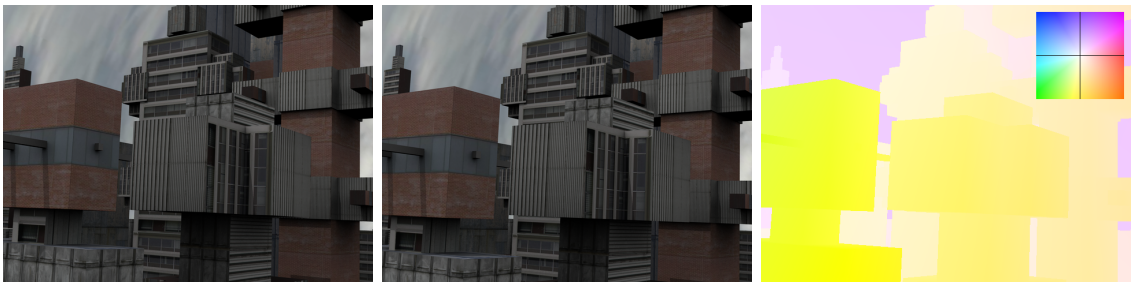
both horizontal and vertical direction [114, 115], or by applying adaptive weights in the cost accumulation operation [98].

About a decade ago, cooperative optimization was introduced into stereo matching [116, 117]. Algorithms based on cooperative optimization decompose the disparity assignment problem into a number of region-based subproblems, which are first solved independently and then individual solutions are fused using a message-passing procedure that enforces inter-region consistency. The approach presented in [117] for instance, which performs remarkably well in stereo matching benchmarks, uses the mean-shift algorithm (a common image segmentation method) to divide the reference image into regions. A local stereo algorithm is used to obtain initial disparity estimates and, under the assumption that disparity assignments are piece-wise planar, a set of parameters defining a disparity plane is generated for each region. Disparity plane parameters are optimized locally in every region based on the information from the adjacent regions. The results of local optimization are propagated across the image enabling the algorithm to reassign disparities in a way that reduces the global energy of the solution, ultimately leading to a coherent disparity map where discontinuities are aligned with object edges.

Some of the most recent advances in stereo matching focus on extending WTA-based algorithms with non-local cost aggregation schemes, while maintaining reasonably low computational complexity. Among such advances is the semiglobal matching (SGM) algorithm proposed by Hirschmuller [118]. The aggregation scheme of SGM performs directional accumulation of cost values, which is similar to the DP-based approaches. Rather than operating along a single scanline, the cost values for a particular disparity

hypothesis are accumulated along 16 straight lines radially converging at the pixel of interest, and then summed to obtain the aggregated (smoothed) cost. In another work [119], minimum spanning trees were constructed in an edge-aware manner for the purpose of cost aggregation, where distances along the paths in the tree correspond to the support values between pairs of pixels. In this scheme, each pixel receives support from all other pixels, which can be efficiently implemented in a series of hierarchical operations and provides truly global cost aggregation.

The latest and arguably the most interesting is the method in [120], coined Patch Match filter, that combines randomized nearest neighbor field estimation with edge-aware filtering. The method operates on superpixel representation of images, i.e., a collection of roughly equally sized patches (superpixels) obtained through constrained image segmentation and the corresponding patch adjacency matrix. Starting with an initial disparity assignment, further matching is performed iteratively, by interleaving steps of superpixel-level cost aggregation and pixel-level correspondence searching. Weight coefficients in the cost aggregation step are computed based on the disparity and intensity samples acquired from the superpixels adjacent to the pixel of interest, which enables the method to quickly reject unlikely matches and find approximate locations of matching pixels. Correspondence searching comes down to the application of the Patch Match algorithm [121] to determine matches with higher precision.



(a) I_0 (b) I_1 (c) Motion field

Figure 3.3: A pair of video frames and the corresponding motion field [122]. In (c), the color encodes the direction of the motion vectors, while the saturation encodes the magnitude. Both the horizontal and vertical flows are normalized to $[-1, 1]$.

3.3 Optical Flow Estimation

When the camera and the objects in the scene experience relative motion, the 3D trajectories of all visible surface points are captured on the image plane as 2D paths. Such paths are composed of displacement vectors arising from frame-to-frame motion. A dense set of displacement vectors relating the pixels in a pair of views is known as a flow field, motion field, or optical flow, and individual displacements are known as motion vectors, flow vectors, or flows. Optical flow estimation focuses on recovering motion fields between subsequent frames in a video sequence, typically with no assumptions regarding the motion of the camera and/or objects, or properties of the scene. The motion field relating a sample pair of frames is shown in Figure 3.3.

Conceptually, the optical flow problem is a generalization of stereo matching to the case of an arbitrary epipolar geometry where the disparities (here, the motion vectors) are allowed to have non-zero vertical components. The applications of optical

flow include motion segmentation of video frames, motion stabilization of videos, and recognition of objects, actions and events, particularly in traffic analysis. Structure reconstruction of a rigid scene is still possible from motion field, however, unlike in stereo matching, triangulation is required to recover depths.

While stereo correspondence problems are solved using discrete optimization techniques, methods for optical flow are based on continuous optimization. Two such methods are now discussed. Denoting the motion vector by $\boldsymbol{u} = (u, v)$ and making it explicitly dependent on $\boldsymbol{x}$, motion vectors $\{\boldsymbol{u}(\boldsymbol{x}_i)\}$ at pixels belonging to an image patch $\{\boldsymbol{x}_i\}$ in the reference frame can be found by minimizing a reformulation of (3.1) given by

$$E(\{\boldsymbol{x}_i\}, \{\boldsymbol{u}(\boldsymbol{x}_i)\}) = \sum_i w(\boldsymbol{x}_i)\Big[I_1(\boldsymbol{x}_i + \boldsymbol{u}(\boldsymbol{x}_i)) - I_0(\boldsymbol{x}_i)\Big]^2, \tag{3.19}$$

where $w(\boldsymbol{x}_i)$ is some window function. Note that this problem is underconstrained, since each motion vector has two unknown components, while each image measurement $I_1(\boldsymbol{x}_i + \boldsymbol{u}(\boldsymbol{x}_i)) - I_0(\boldsymbol{x}_i)$ adds only a single constraint.

One solution is to assume that the pixels within the image patch centered at some pixel of interest $\boldsymbol{x}$ undergo displacement by the same vector [11]. Using $I_t(\boldsymbol{x})$ as a short-hand notation for the temporal difference $I_1(\boldsymbol{x}) - I_0(\boldsymbol{x})$ and using the Taylor series expansion $I_1(\boldsymbol{x} + \boldsymbol{u}) \approx I_1(\boldsymbol{x}) + \nabla I_1(\boldsymbol{x}) \cdot \boldsymbol{u}$, (3.19) is rewritten as

$$E(\boldsymbol{x}, \boldsymbol{u}) = \sum_i w(\boldsymbol{x}_i)\Big[\nabla I_1(\boldsymbol{x}_i) \cdot \boldsymbol{u} + I_t(\boldsymbol{x}_i)\Big]^2. \tag{3.20}$$

Since generally there will not exist a flow vector that satisfies all of the constraints, one should find a solution that minimizes the total squared error in (3.20). This is done by setting the derivatives of the error with respect to the horizontal and vertical

components of the motion vector to zero, which leads to the system of equations

$$\frac{\partial E(\boldsymbol{x},\boldsymbol{u})}{\partial u} = \sum_i w(\boldsymbol{x}_i)\Big[uI_x^2(\boldsymbol{x}_i) + vI_x(\boldsymbol{x}_i)I_y(\boldsymbol{x}_i) + I_x(\boldsymbol{x}_i)I_t(\boldsymbol{x}_i)\Big] = 0 \tag{3.21}$$

$$\frac{\partial E(\boldsymbol{x},\boldsymbol{u})}{\partial v} = \sum_i w(\boldsymbol{x}_i)\Big[vI_y^2(\boldsymbol{x}_i) + uI_x(\boldsymbol{x}_i)I_y(\boldsymbol{x}_i) + I_y(\boldsymbol{x}_i)I_t(\boldsymbol{x}_i)\Big] = 0\,, \tag{3.22}$$

where I_x and I_y are the directional derivatives of I_1.

Equations (3.21) and (3.22) can be written in a matrix form $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b}$, where $\boldsymbol{A}$ is the auto-correlation matrix (evaluated around pixel $\boldsymbol{x}$ in I_1) and $\boldsymbol{b}$ is given by

$$\mathbf{b} = -\begin{bmatrix} \sum_i w(\boldsymbol{x}_i)I_x(\boldsymbol{x}_i)I_t(\boldsymbol{x}_i) \\ \\ \sum_i w(\boldsymbol{x}_i)I_y(\boldsymbol{x}_i)I_t(\boldsymbol{x}_i) \end{bmatrix}. \tag{3.23}$$

If $\boldsymbol{A}$ is full-rank, the vector $\boldsymbol{u} = \boldsymbol{A}^{-1}\boldsymbol{b}$ is the least-squares solution to the problem. Note that this will not be the case if the patch lacks distinct two-dimensional image structures. This is known as the aperture problem and results in rank-deficient auto-correlation matrix when computing flow vectors at pixels located on the edges of objects or in textureless areas, as these cannot be unambiguously matched. Due to Taylor series approximation in (3.20) that only holds for small displacements, and the fact that this formulation considers windows of pixels that are limited in size, the method is unable to recover larger motion vectors.

To enable recovery of large flows, Ananadan [40] adopted multi-scale processing of frames. Progressing from the most coarse to the finest scale, his method alternates between steps of evaluating local displacements to obtain initial flow vectors and steps of least-squares estimation in order to arrive at a more accurate solution. A confidence measure was defined that is used to propagate flow estimates to pixels where matching is highly ambiguous, providing a way to overcome the aperture problem. A universal

multi-scale framework was later established in [123] where the flow estimates are used to warp the reference image, allowing the fine-scale flows to be recovered incrementally in a series of local correspondence searches at subsequent scales. In the same work, the least-squares approach was extended to enable the estimation of parametric optical flow modeled using affine transformations. Spline-based representations of motion fields were later proposed in [124].

An alternative formulation introduces smoothness constraints into the problem by penalizing local variations of the motion field. To recover the motion filed, the total variation (TV) energy functional is minimized given by

$$E = \int \Big\{ \lambda\phi(I_1(\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})) - I_0(\boldsymbol{x}))) + \psi(\boldsymbol{u}, \nabla\boldsymbol{u}, \ldots) \Big\} d\boldsymbol{x} \tag{3.24}$$

where $\phi(\boldsymbol{x})$ and $\psi(\boldsymbol{u}, \nabla\boldsymbol{u}, \ldots)$ are two penalty functions applied to the image intensity difference and the motion field (or its derivatives), respectively. Notice the evident similarity of this formulation to global formulations of the stereo matching problem. In one of the pioneering works, Horn and Schunck [125] considered a variant of the problem with $\phi(\boldsymbol{x}) = (I_1(\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})) - I_0(\boldsymbol{x}))^2$ and $\psi(\nabla\boldsymbol{u}) = |\nabla\boldsymbol{u}|^2$ and demonstrated how this problem can be solved using methods of calculus of variations.

More commonly found TV energy functionals involve data and smoothness terms based on the L^1 norms, i.e.,

$$E = \int \Big\{ \lambda|I_0(\boldsymbol{x}) - I_1(\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})| + |\nabla\boldsymbol{u}| \Big\} d\boldsymbol{x}\,, \tag{3.25}$$

A clever solution technique was proposed by Zach, Pock and Bishof [126] that employs numerical schemes originally developed for the problems of image denoising and restoration in order to minimize (3.25). Their technique is as follows. The image

intensity function I_1 is first linearized around a point $\boldsymbol{x} + \boldsymbol{u}_0$ (for some choice of the initial displacement $\boldsymbol{u}_0$) allowing the image residual $\rho(\boldsymbol{x}) = |I_0(\boldsymbol{x}) - I_1(\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})|$ to be written as

$$\rho(\boldsymbol{u}) = I_1(\boldsymbol{x} + \boldsymbol{u}_0(\boldsymbol{x})) + \nabla I_1(\boldsymbol{x}) \cdot (\boldsymbol{u}(\boldsymbol{x}) - \boldsymbol{u}_0(\boldsymbol{x})) - I_0(\boldsymbol{x})\,. \tag{3.26}$$

A convex energy functional is considered given by

$$E = \int \left\{ |\nabla \boldsymbol{u}| + \frac{1}{2\theta}\left[(u - \tilde{u})^2 + (v - \tilde{v})^2\right] + \lambda |\rho(\tilde{\boldsymbol{u}})| \right\} d\boldsymbol{x}\,, \tag{3.27}$$

where $\tilde{\boldsymbol{u}} = (\tilde{u}, \tilde{v})$ is an auxiliary flow estimate assumed to closely approximate the solution $\boldsymbol{u}$, and θ is a small constant. Note that as $\tilde{\boldsymbol{u}}$ approaches $\boldsymbol{u}$, the middle term vanishes and the energy functional becomes the one in (3.25).

The minimization of (3.27) is performed using an iterative procedure that updates $\boldsymbol{u}$ and $\tilde{\boldsymbol{u}}$ in two alternating steps:

1. For fixed $\tilde{\boldsymbol{u}}$, solve

$$\min_{\boldsymbol{u}} \int \left\{ |\nabla \boldsymbol{u}| + \frac{1}{2\theta}\left[(u - \tilde{u})^2 + (v - \tilde{v})^2\right] \right\} d\boldsymbol{x}\,. \tag{3.28}$$

   The problem in (3.28) is further decomposed into two independent subproblems of minimizing the terms involving u and the terms involving v. The horizontal component of the solution is the minimizer of

$$\min_{u} \int \left\{ |\nabla u| + \frac{1}{2\theta}(u - \tilde{u})^2 \right\} d\boldsymbol{x}\,, \tag{3.29}$$

   which is reminiscent of the formulation of the image denoising problem in [127]. To solve (3.29), Chambolle's primal-dual algorithm [128] is adopted that performs a variant of projected gradient descent on u. The vertical flow component v can be found analogously.

2. For fixed $\boldsymbol{u}$, solve

$$\min_{\tilde{\boldsymbol{u}}} \left\{ \frac{1}{2\theta} \left[(u - \tilde{u})^2 + (v - \tilde{v})^2 \right] + \lambda |\rho(\tilde{\boldsymbol{u}})| \right\} \tag{3.30}$$

The above does not depend on the derivative of $\tilde{\boldsymbol{u}}$ and hence is solved point-wise by allowing $\tilde{\boldsymbol{u}}$ to make bounded steps towards $\boldsymbol{u}$ to decrease the residual.

In practice, to handle large flows and to account for non-linearities in the intensity function I_1, this iterative minimization procedure must be applied in a coarse-to-fine way. Again, this is done by embedding the procedure in the multi-scale framework, such that the flow estimates obtained at a given scale are upsampled and multiplied by the scaling factor to become the initial displacements $\boldsymbol{u}_0$ at a finer scale. An important extension to this method has been proposed in [129] where disparities and motion vectors are estimated jointly through total variation minimization. Note that the knowledge of the disparity and the flow vector at an image point allows for the recovery of the 3D motion vector associated with the corresponding scene point.

CHAPTER 4

Stereo Matching with Iterative Refinement

In this chapter, a real-time stereo matching method implemented on parallel graphics hardware is introduced that uses an adaptive cost aggregation scheme based on edge-preserving filtering and a low-complexity iterative disparity refinement technique. The iterative disparity refinement technique is constructed using a probabilistic framework through a series of approximations of the aggregated matching cost. In the proposed method, both cost aggregation and iterative disparity refinement make use of a two-pass approximation of the adaptive support weights introduced in [17]. First, the adaptive support weights are used in combination with the joint bilateral filter [130] to efficiently aggregate matching costs. Second, another set of adaptive support weights is generated to iteratively refine the disparity map. Disparity refinement operates by computing the expected value of the disparity during the current iteration based on nearby pixel disparities from previous iterations. The matching cost is then penalized

when the initially computed disparity deviates from the expected value.

4.1 Adaptive Support-Weight Correspondences

Adaptive support weights were introduced in 2005 as a new way of aggregating the cost in window-based stereo matching [17]. This method was shown to improve the accuracy of matching when compared to window-based methods that attempt to compute the optimal position or shape of the support window [95, 131]. The accuracy of the adaptive support-weight method made it one of the first local methods capable of competing with global algorithms that use graph cuts [132] or belief propagation [82], and it has since been used in many of the algorithms listed on the online Middlebury stereo benchmark [12].

The adaptive support-weight stereo matching algorithm mimics the process of visual grouping in the human vision system through the application of the gestalt principles of perception, among which the principle of proximity and the principle of similarity are particularly relevant to the problem of stereo correspondence. The principle of proximity assumes scene surfaces to be locally continuous. Consequently, the likelihood that pixel $\boldsymbol{p}$ belongs to the same surface as pixel $\boldsymbol{q}$ decreases as the Euclidean distance between $\boldsymbol{p}$ and $\boldsymbol{q}$ increases. The principle of similarity states that typical scene surfaces have locally consistent color. Thus, it is likely that pixel $\boldsymbol{p}$ is on the same surface as pixel $\boldsymbol{q}$ when their color and shade are similar.

In order to make use of the gestalt principles of perception for the purposes of stereo matching, the adaptive support-weight stereo algorithm considers a support

window $\Omega_{\boldsymbol{p}}$ of pixel $\boldsymbol{p}$ in the reference image I_0, that is, a square region centered at pixel $\boldsymbol{p}$, and assigns a support weight to each pixel $\boldsymbol{q} \in \Omega_{\boldsymbol{p}}$. Let $\Delta_c(I_0(\boldsymbol{p}), I_0(\boldsymbol{q}))$ denote the color difference and let $\Delta_g(\boldsymbol{p}, \boldsymbol{q})$ denote the Euclidean distance between pixels $\boldsymbol{p}$ and $\boldsymbol{q}$. The support weight, or simply weight, assigned to pixel $\boldsymbol{q}$ in the support window of $\boldsymbol{p}$ is given by

$$w(\boldsymbol{p}, \boldsymbol{q}) = \exp\left(-\frac{\Delta_c(I_0(\boldsymbol{p}), I_0(\boldsymbol{q}))}{\gamma_c} - \frac{\Delta_g(\boldsymbol{p}, \boldsymbol{q})}{\gamma_g}\right), \tag{4.1}$$

where the parameters γ_c and γ_g regulate the strength of grouping by similarity and proximity, respectively.

To identify a match for a particular pixel of interest $\boldsymbol{p} = (x, y)$ in the reference image, adaptive support-weight matching costs are calculated between $\boldsymbol{p}$ and pixels in the matched image whose coordinates belong to the set $S_{\boldsymbol{p}} = \{(x - d, y) \mid 0 \leq d \leq d_{max}\}$, where d_{max} is the maximum disparity value. The set $S_{\boldsymbol{p}}$ is known as the correspondence search domain of $\boldsymbol{p}$. Given the pixel $\boldsymbol{p}$, some pixel $\bar{\boldsymbol{p}} \in S_{\boldsymbol{p}}$, and their support windows $\Omega_{\boldsymbol{p}}$ and $\Omega_{\bar{\boldsymbol{p}}}$, respectively, the matching cost is computed as

$$C(\boldsymbol{p}, \bar{\boldsymbol{p}}) = \frac{\sum_{\boldsymbol{q} \in \Omega_{\boldsymbol{p}}, \bar{\boldsymbol{q}} \in \Omega_{\bar{\boldsymbol{p}}}} w(\boldsymbol{p}, \boldsymbol{q}) w(\bar{\boldsymbol{p}}, \bar{\boldsymbol{q}}) \delta(\boldsymbol{q}, \bar{\boldsymbol{q}})}{\sum_{\boldsymbol{q} \in \Omega_{\boldsymbol{p}}, \bar{\boldsymbol{q}} \in \Omega_{\bar{\boldsymbol{p}}}} w(\boldsymbol{p}, \boldsymbol{q}) w(\bar{\boldsymbol{p}}, \bar{\boldsymbol{q}})}, \tag{4.2}$$

where $w(\boldsymbol{p}, \boldsymbol{q})$ is as in (4.1), $w(\bar{\boldsymbol{p}}, \bar{\boldsymbol{q}})$ is calculated analogously for pixels $\bar{\boldsymbol{p}}$ and $\bar{\boldsymbol{q}}$ in the matched image I_1, and $\delta(\boldsymbol{q}, \bar{\boldsymbol{q}})$ is an arbitrary distance measure between pixels $\boldsymbol{q}$ and $\bar{\boldsymbol{q}}$. Typically $\delta(\boldsymbol{q}, \bar{\boldsymbol{q}})$ is the sum of absolute color differences.

Once the matching costs have been computed for all candidate pixels $\bar{\boldsymbol{p}} \in S_{\boldsymbol{p}}$, a match for pixel $\boldsymbol{p}$ can be obtained using the Winner-Take-All (WTA) decision criteria that selects the candidate pixel characterized by the minimum matching cost.

Precisely, the match $m(\boldsymbol{p})$ for pixel $\boldsymbol{p}$ is given by

$$m(\boldsymbol{p}) = \underset{\bar{\boldsymbol{p}} \in S_{\boldsymbol{p}}}{\operatorname{argmin}} C(\boldsymbol{p}, \bar{\boldsymbol{p}}) \,. \tag{4.3}$$

4.2 Cost Aggregation as a Filtering Operation

The formulation of cost aggregation using the adaptive support weights closely resembles the edge-preserving bilateral image filter proposed by Tomasi and Manduci [133]. In fact, dropping the factors dependent on the matched image in equation (4.2) results in a guided variant of the bilateral filter, known as the joint bilateral filter [130]. In this section a disparity-oriented stereo matching approach is outlined that uses the joint bilateral filter to perform cost aggregation in an adaptive way. This framework is then adopted by the proposed stereo matching method.

Assume that the pixel-wise dissimilarity metrics are organized in a volume C, such that the element $C(x, y, d)$, also interchangeably denoted as $C(\boldsymbol{p}, d)$ or $C(\boldsymbol{p}, \bar{\boldsymbol{p}})$, contains the dissimilarity metric evaluated between pixels $\boldsymbol{p} = (x, y)$ and $\bar{\boldsymbol{p}} = (x - d, y)$. The proposed method uses a cost metric that incorporates both color and gradient information. Denoting the directional gradients computed along the x-direction for both images as $\frac{\partial I_0}{\partial x}$ and $\frac{\partial I_1}{\partial x}$ and the corresponding y-direction gradients as $\frac{\partial I_0}{\partial y}$ and $\frac{\partial I_1}{\partial y}$, the cost metric is given by

$$\begin{aligned} C(\boldsymbol{p}, d) = \alpha \cdot \min\Big(&\mathrm{SAD}(I_0(\boldsymbol{p}), I_1(\bar{\boldsymbol{p}})), \tau_c\Big) \\ &+ (1 - \alpha) \cdot \min\left(\left|\frac{\partial I_0}{\partial x}(\boldsymbol{p}) - \frac{\partial I_1}{\partial x}(\bar{\boldsymbol{p}})\right| + \left|\frac{\partial I_0}{\partial y}(\boldsymbol{p}) - \frac{\partial I_0}{\partial y}(\bar{\boldsymbol{p}})\right|, \tau_g\right) , \end{aligned} \tag{4.4}$$

where the scalar α balances the color and gradient terms, $\mathrm{SAD}(I_0(\boldsymbol{p}), I_1(\bar{\boldsymbol{p}}))$ denotes the sum of absolute color differences evaluated at pixel coordinates $\boldsymbol{p}$ and $\bar{\boldsymbol{p}}$ in the

input images, and τ_c and τ_g are truncation values that limit the maximum values of both terms. Truncation of cost values allows stereo matching to better cope with image noise, while the presence of gradient terms makes the cost metric invariant to illumination changes, which is advantageous if the two imagers deliver images of inconsistent color parameters, such as brightness and saturation.

Consider the joint bilateral image filter given by

$$F'(\boldsymbol{p}) = \frac{\sum\limits_{\boldsymbol{q}\in\Omega_{\boldsymbol{p}}} R(G(\boldsymbol{p}), G(\boldsymbol{q})) \cdot S(\boldsymbol{p}, \boldsymbol{q}) \cdot F(\boldsymbol{q})}{\sum\limits_{\boldsymbol{q}\in\Omega_{\boldsymbol{p}}} R(G(\boldsymbol{p}), G(\boldsymbol{q})) \cdot S(\boldsymbol{p}, \boldsymbol{q})} \tag{4.5}$$

where F and G are the input and guidance images, F' is the output image, and functions R and S are the range and spatial filter kernels, respectively. The range kernel assesses the likelihood of pixels $\boldsymbol{p}$ and $\boldsymbol{q} \in \Omega_{\boldsymbol{p}}$ belonging to the same surface by evaluating their range distance, i.e., the dissimilarity of color intensities. Analogously, the spatial kernel assesses the the likelihood of pixels $\boldsymbol{p}$ and $\boldsymbol{q}$ belonging to the same surface based on their geometric distance within the spatial domain of the image. It follows from equation (4.5) that, like many other image filters, the joint bilateral filter computes the output as a weighted sum of the pixel intensities within the support region of the considered pixel.

The use of the range kernel together with the usual spatial kernel, however, provides a mechanism for adaptive shaping of the support region that forces the edges in the output image to coincide with those in the guidance image. Note that choosing

$$R(G(\boldsymbol{p}), G(\boldsymbol{q})) = \exp(-\frac{\Delta_c(I_0(\boldsymbol{p}), I_0(\boldsymbol{q}))}{\gamma_c}) \tag{4.6}$$

and

$$S(\boldsymbol{p}, \boldsymbol{q}) = \exp(-\frac{\Delta_g(\boldsymbol{p}, \boldsymbol{q})}{\gamma_g}) \tag{4.7}$$

makes the product $R(G(\boldsymbol{p}), G(\boldsymbol{q})) \cdot S(\boldsymbol{p}, \boldsymbol{q})$ equivalent to the adaptive support weight introduced in equation (4.1).

In view of the above, using the reference view as an implicit guidance image, the cost can be adaptively aggregated by iterating over all (x, y)-slices of the cost volume and filtering each slice according to

$$C'(\boldsymbol{p}, d) = \frac{\sum\limits_{\boldsymbol{q} \in \Omega_{\boldsymbol{p}}} w(\boldsymbol{p}, \boldsymbol{q}) C(\boldsymbol{q}, d)}{\sum\limits_{\boldsymbol{q} \in \Omega_{\boldsymbol{p}}} w(\boldsymbol{p}, \boldsymbol{q})} \,. \tag{4.8}$$

In general, the weight coefficients in equation (4.8) can be replaced with the coefficients of an arbitrary filter, although edge-preserving filters are particularly useful in stereo matching. For instance, uniform weights, as in the box filter, can be used to simply average cost values across the support region. Furthermore, in the cost filtering framework it is convenient to think of matches using the notion of disparities rather than pairs of matching pixels. Once the bilateral filter has been applied to the slices of the cost volume, disparities are assigned using

$$D(\boldsymbol{p}) = \underset{d}{\operatorname{argmin}}\, C'(\boldsymbol{p}, d) \,, \tag{4.9}$$

which is a disparity-oriented reformulation of the match selection criteria in equation (4.3).

The effects of cost filtering using the bilateral filter are illustrated in Figure 4.1 using the `cones` Middlebury stereo images as inputs [12]. Ground truth disparities are provided for reference. Additionally, the disparities obtained from an unfiltered

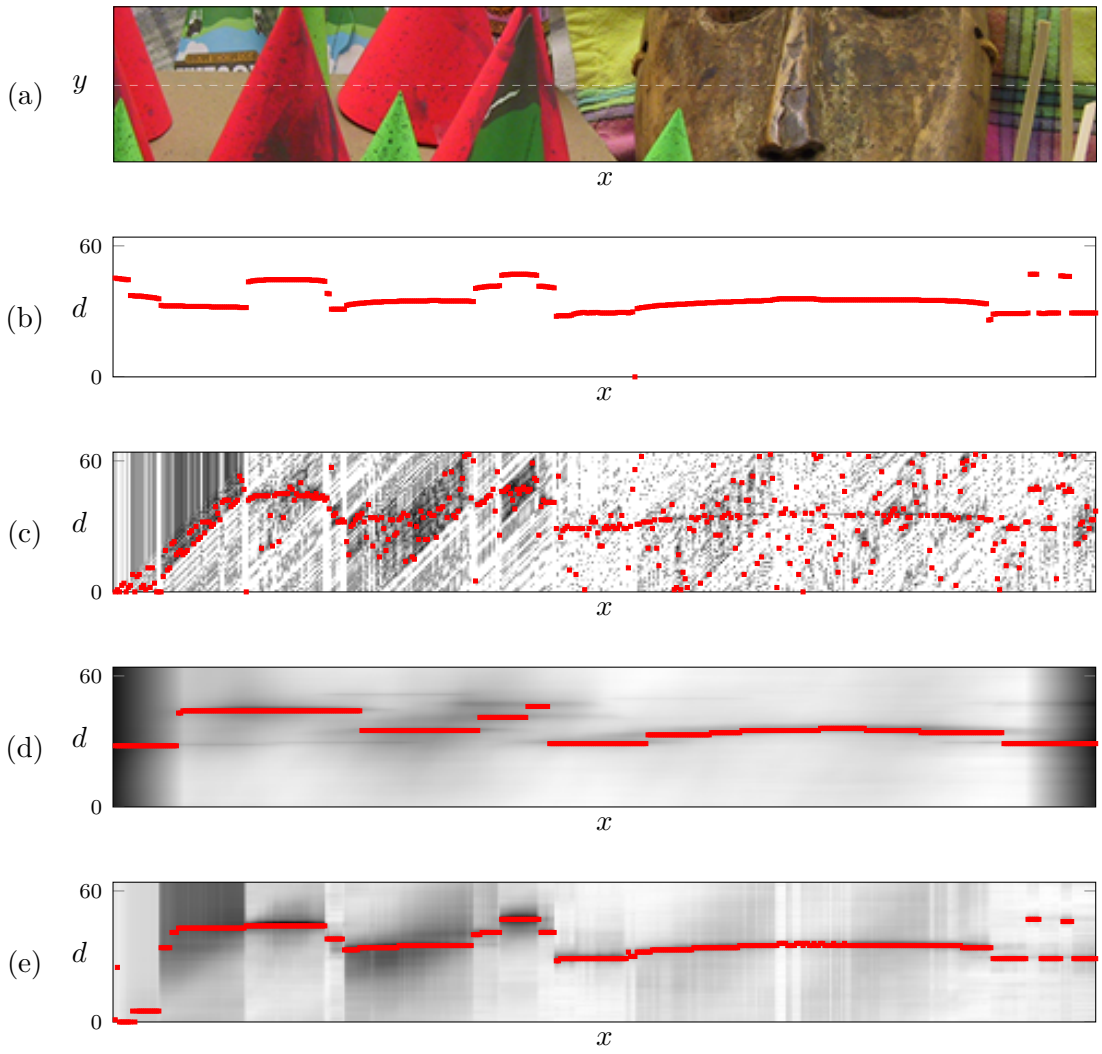


Figure 4.1: The role of edge-preserving filtering in cost aggregation: an excerpt from the `cones` Middlebury dataset (left view) with a scanline highlighted (a), true disparities (b), cost slices and the corresponding disparities obtained without filtering (c), using box filter (d), and using the joint bilateral filter (e). Darker shades of gray indicate lower cost values.

cost and the disparities obtained through box filtering of the cost volume are shown with the corresponding cost slices. The slices shown are the (x, d)-slices containing the cost metrics associated with the image scanline highlighted in Figure 4.1a, while cost filtering is performed along the x and y dimensions. Observe that operating on raw cost values produces noisy disparities, which is a key argument for cost aggregation using image filters. The box filter effectively counteracts the noise, yet its application leads to oversmoothing of the cost values around the edges visible in the image. As a consequence, objects appear dilated (fattened) or eroded (shrunk) in the resulting disparity maps. This oversmoothing also prevents stereo matching based on box filtering from capturing small objects, such as the sticks close to the right boundary of the image. The bilateral filter, on the other hand, has the capability to capture small objects and avoids the dilation and erosion of objects by forcing major disparity discontinuities to be aligned with object boundaries.

4.3 Efficient Edge-Preserving Cost Filtering

While adaptive cost aggregation through joint bilateral filtering enhances the accuracy of stereo matching, its high computational complexity makes it unsuitable for use in real-time applications. More specifically, it is the evaluation of the adaptive support weights across a broad support region that requires substantial computational effort. Thus, in order to achieve real-time performance, it is necessary to reduce the complexity of cost aggregation using the joint bilateral filter. To address this issue, a plethora of approximations, modifications and alternatives of the bilateral filter have

been proposed in the literature and these solutions are now reviewed. In addition, a thorough comparison is given of the solutions that allow for efficient cost aggregation in stereo matching.

The bilateral grid [134, 135] introduced by Chen et al. is a supplementary data structure designed to facilitate efficient bilateral filtering and was among the first approaches that enabled stereo matching to achieve interactive frame rates [136] when processing grayscale images on parallel hardware. The high memory requirements of the bilateral grid, however, prevent its application to color images of practical sizes. Soon after the creation of the bilateral grid several constant-time approximations of the bilateral filter were proposed that have computational complexity that is linear in the image size and independent of the filter radius. These so called $O(1)$ bilateral filters rely on precomputed auxiliary variables to perform filtering (typically some form of integral images), including integral histograms [137], integral cosine images [138], or integral images of shiftable kernels [139, 140]. Computation of integral images requires the computation of cumulative sums across rows and columns of the input images that is sequential in nature, and, while there exist parallel algorithms for the evaluation of cumulative sums [141], the time overhead associated with initializing and populating the auxiliary data structures outweighs the performance gain of cost filtering in stereo matching. In a related work, He et al. proposed an alternative edge-preserving filter with applications in stereo matching that introduced a novel way of generating support weights. This new filter, called the guided image filter [142], can be computed in constant time regardless of the filter radius by evaluating a sequence of box filters. When employed for cost filtering in stereo matching, the guided filter

enabled the recovery of accurate disparity maps in real-time [143].

Recently, a recursive formulation of the bilateral filter has been proposed [119]. The recursive bilateral filter, which extends upon Deriche's notion of recursive filtering [144], features a range kernel that does not compare disconnected pixels directly. Instead, this range kernel considers similarity between pairs of connected pixels located on the path from the pixel of interest to a pixel within its support region. It was shown that using the proposed range kernel the filter output can be computed in a separable way by performing forward and backward scans along the rows and columns of the input image and combining the intermediate results. The key concepts behind the recursive bilateral filter have lately been refined in Yang's hardware-efficient bilateral filter [105] that is well-suited for implementation on parallel graphics hardware. The hardware-efficient bilateral filter is integrated into the multi-scale image processing framework, where the input image is first downsampled and then successively filtered and upsampled until the original scale is reached. This parallel variant of the recursive bilateral filter was shown to provide highly accurate disparity maps and real-time stereo matching.

While many approaches focused on reducing or fully eliminating the impact of the filter radius on the performance of filtering, other major research efforts have been oriented towards achieving efficient bilateral filtering by approximating the adaptive support weights. The proposed approximations, all of which have been developed specifically for the purpose of cost filtering in stereo matching, include two-pass adaptive support weights [98], block-based adaptive weights [103], exponential step-size adaptive weights [100], and cross-based support weights [102]. In the remaining

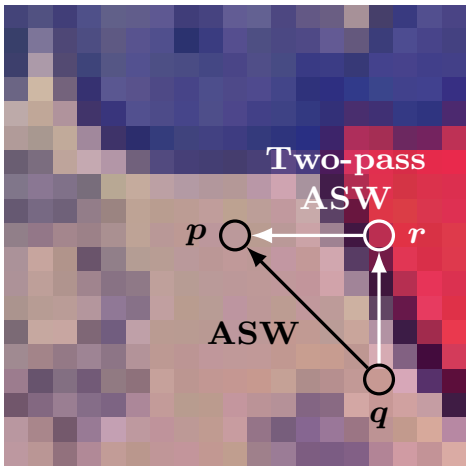


Figure 4.2: Support weights relating the pixel of interest $\boldsymbol{p}$ to its neighbor $\boldsymbol{q} \in \Omega_{\boldsymbol{p}}$ using regular adaptive support weight ("ASW") and a two-pass approximation ("Two-pass ASW") passing through an intermediate pixel $\boldsymbol{r} \in \Omega_{\boldsymbol{p}}$.

part of this section these approximations are thoroughly explained, and the support weights generated using individual approximation schemes are compared.

Instead of using square windows, the two-pass approach approximates the output of the joint bilateral filter by performing cost filtering in the vertical and then the horizontal direction [98]. During the vertical step, input values (here, the pixel-wise costs metrics) are added within a one-dimensional column of pixels centered at the pixel of interest, whereas in the horizontal step a weighted sum of matching costs along the rows is computed using the adaptive support weights. As a result, the complexity of aggregating the matching cost within a window of size $\omega \times \omega$ is reduced from $O(\omega^2)$ to $O(\omega)$. The two-pass approach is sometimes referred to as a separable implementation of the bilateral filter, although the outputs produced using the original filter and the two-pass version are not exactly the same.

While the two-pass approximation significantly improves computational efficiency,

it fails to accurately approximate the support weights under certain conditions. Figure 4.2 illustrates an example where the two-pass algorithm fails to capture the relationship between the pixel of interest $\boldsymbol{p}$ and pixel $\boldsymbol{q}$ in its support window. The original adaptive weights method compares these two pixels directly to generate a support weight, but the two-pass approximation compares the neighboring pixel $\boldsymbol{q}$ to an intermediate pixel $\boldsymbol{r}$ during the vertical step and then compares $\boldsymbol{r}$ to $\boldsymbol{p}$ in the horizontal step, thus the normalized weight is approximated by $w(\boldsymbol{p},\boldsymbol{q}) \approx w(\boldsymbol{q},\boldsymbol{r}) \times w(\boldsymbol{r},\boldsymbol{p})$. The two pixels $\boldsymbol{p}$ and $\boldsymbol{q}$ in Figure 4.2 are very similar, both in terms of color and shade, however, the pairs of pixels $(\boldsymbol{q},\boldsymbol{r})$ and $(\boldsymbol{r},\boldsymbol{p})$ are not similar. As a result, the two-pass approximation wrongly assumes strong dissimilarity between $\boldsymbol{p}$ and $\boldsymbol{q}$.

Although inherently sequential, the recursive formulation of the bilateral filter [119] delivers a useful range kernel that, when combined with the two-pass approach in a fixed-size window, can further accelerate cost filtering on parallel hardware. Starting at the central pixel with a weight of a unity and proceeding away from the window's center, the weight is subsequently multiplied by the value of a feedback coefficient (a fractional scalar replacing the usual space kernel) and the range distance, given in equation (4.6), is evaluated between the current and the previous pixel. While the computational complexity of this approach is still $O(\omega)$, the pair-wise range distances can be precomputed and reused in the evaluation of the filter output at neighboring pixels thus allowing for efficient parallel implementation. In this way, the number of exponentiations required to compute the filtered cost at a single pixel is significantly reduced.

A modified adaptive support-weight aggregation scheme using block-based approx-

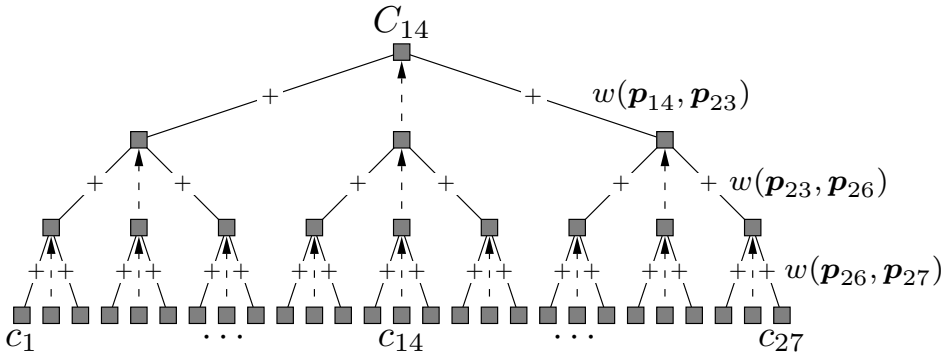


Figure 4.3: Pixel-wise costs $(c_1, \ldots, c_{27})$ aggregated recursively in groups of three using exponential step-size adaptive weights (ESAW). Each edge operation '+' represents a normalized weighted sum using adaptive support weights.

imated joint bilateral filtering was introduced in [99]. This scheme, which is a vital component of Mattoccia's fast bilateral stereo (FBS) algorithm [103], operates by dividing the support region into blocks of $\omega_b \times \omega_b$ pixels. Instead of assigning each pixel in the block a unique support weight, a single support weight is assigned to the entire block. The spatial component of the support weight is generated using the distance between the pixel in the center of the block and the pixel of interest, and the color component of the support weight is generated using the difference between the average block color and the color of the pixel of interest. By computing only one support weight per block, the computational complexity associated with generating the support weights is reduced to $O(\omega^2/\omega_b^2)$. In addition to a reduction in complexity, this aggregation scheme has been shown to provide a higher level of robustness to image noise than the original adaptive support-weight method.

An alternative method for approximating the adaptive support weights was presented in [100]. This method also uses a two-pass approximation to square-window

adaptive support-weight stereo matching, but the aggregation is further accelerated by recursively combining the matching costs over subsets of pixels. This approach, referred to as exponential step-size adaptive weights (ESAW), reduces the complexity of aggregating the matching cost of ω pixels in both the vertical and horizontal directions from $O(\omega)$ to $O(\log(\omega))$. To illustrate the operations of the ESAW method, the example shown in Figure 4.3 considers a set of pixel-wise costs $\{c_1, \ldots, c_{27}\}$ that are used to compute the matching cost C_{14} between pixel $\boldsymbol{p}_{14}$ and some arbitrary pixel in the target image. In this example, the pixel-wise cost c_{27} is aggregated into C_{14} by performing a weighted sum using the normalized adaptive support weights relating pixel $\boldsymbol{p}_{27}$ to $\boldsymbol{p}_{26}$, $\boldsymbol{p}_{26}$ to $\boldsymbol{p}_{23}$, and finally $\boldsymbol{p}_{23}$ to $\boldsymbol{p}_{14}$. The resulting adaptive support weight that is used to evaluate the similarity between pixels $\boldsymbol{p}_{14}$ and $\boldsymbol{p}_{27}$ is approximated by $w(\boldsymbol{p}_{14}, \boldsymbol{p}_{27}) \approx w(\boldsymbol{p}_{14}, \boldsymbol{p}_{23}) \times w(\boldsymbol{p}_{23}, \boldsymbol{p}_{26}) \times w(\boldsymbol{p}_{26}, \boldsymbol{p}_{27})$. Thus, the similarity of pixels $\boldsymbol{p}_{14}$ and $\boldsymbol{p}_{27}$ estimated using ESAW depends on a chain of three intermediate comparisons. Because these approximations are used within both the horizontal and vertical aggregation, ESAW is even more susceptible to erroneous approximations of support weights than the two-pass method.

Figure 4.4 compares the previously discussed methods for computing support weights of the bilateral filter. Figure 4.4b shows the original adaptive support weights generated for the five image patches given in Figure 4.4a. The intensity images illustrating the support weights have been obtained through a linear mapping of the weight values to the interval $[0.0, 1.0]$. The support weights generated using the two-pass approximation, which are given in Figure 4.4b, display similar patterns when compared to the original adaptive support weights. However, as demonstrated in

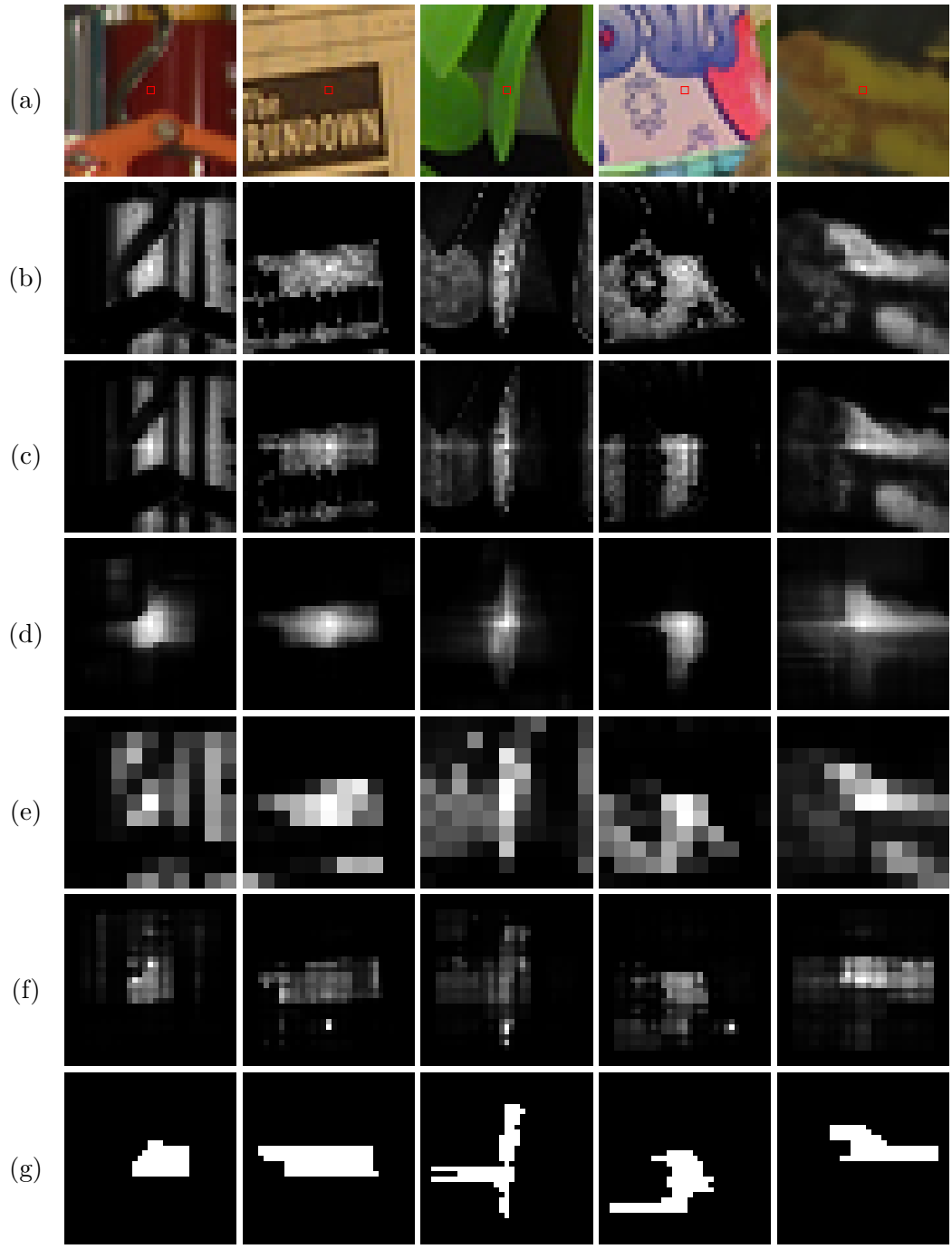


Figure 4.4: A comparison of adaptive support weights and their approximations: sample image patches (a), adaptive support weights (b), two-pass approximation (c), recursive approximation (d), block-based support weights (e), exponential step-size support weights (f), cross-based binary support weights (g).

Figure 4.2, the limitations of this method can be seen in areas where the path from neighboring pixels to the pixel of interest passes through areas of strong dissimilarity. In order to adjust the shape of the support region through adaptive weights, the recursive approximation illustrated in Figure 4.4d relies on multiplication of color dissimilarities measured between pairs of connected pixels located along the path from the central pixel to a pixel under consideration. Note that this method poses a very stringent constraint on the photometric consistency of pixels within the same object, making it more vulnerable to image noise and, as a result, unable to fully capture the shape of objects in the scene. Figure 4.4b demonstrates the block-based support weights obtained using a block size of 3×3. Unlike the other approximations considered, the block-based approximation does not use a two-pass approach and thus it does not suffer from the disadvantages of algorithms that rely on intermediate weights. However, due to averaging within the blocks, the support weights are less accurate along object boundaries. The support weights produced by ESAW, shown in Figure 4.4f, weakly resemble those of the original adaptive support weights and include disproportionally high weights for pixels that are isolated from their neighbors in terms of color. These high weights result from the concatenation and normalization of weight values when estimating the support weights of such isolated pixels. Finally, the binary cross-based support weights are given in Figure 4.4g. Although it captures nearby surface similarities around the pixel of interest, this method is incapable of crossing color boundaries, struggles to define edges, and produces streaking artifacts due to the thresholding operation.

ESAW, cross-based support weights, and the recursive method are intended to

reduce the computational complexity of cost filtering in stereo matching, however, they introduce a tradeoff between processing time and the accuracy of support weight approximation. As illustrated in Figure 4.4, the two-pass method outperforms both ESAW and cross-based support weights in terms of the overall accuracy of the support weight approximation. In contrast, the block-based method presents the opposite tradeoff, producing accurate support weights at the expense of increased computational complexity when compared to the two-pass approach. Therefore, to achieve both highly reduced computational complexity and accurate support weight approximations, the two-pass approach is used by the stereo matching method proposed in the next section.

4.4 Iterative Disparity Refinement

In this section, a new method for iterative disparity refinement is derived that improves the accuracy of adaptive support-weight stereo matching. Let $\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}}$ denote the probabilistic event that pixel $\bar{\boldsymbol{p}}$ in the target image is the correct match for pixel $\boldsymbol{p}$ in the reference image. Recall that the decision criteria for finding a match for pixel $\boldsymbol{p}$ is given by equation (4.3). In the ideal case, the resulting match $\bar{\boldsymbol{p}} = m(\boldsymbol{p})$ is the candidate with the highest probability of being the correct match given the two images, i.e.,

$$m(\boldsymbol{p}) = \underset{\bar{\boldsymbol{p}} \in S_{\boldsymbol{p}}}{\operatorname{argmax}} P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}} \mid I, \bar{I}) \,. \tag{4.10}$$

In general, the size of the images prohibits the evaluation of equation (4.10) and it is necessary to reduce the computational complexity associated with selection of

matches. To obtain a more manageable expression, the class of window-based stereo matching methods considers only the support windows $\Omega_{\boldsymbol{p}}$ and $\Omega_{\bar{\boldsymbol{p}}}$ surrounding pixels $\boldsymbol{p}$ and $\bar{\boldsymbol{p}}$, respectively. A reduced-complexity approximation of equation (4.10), achieved using the window-based approach, is given by

$$m(\boldsymbol{p}) \approx \underset{\bar{\boldsymbol{p}} \in S_{\boldsymbol{p}}}{\operatorname{argmax}} P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}} \mid \Omega_{\boldsymbol{p}}, \Omega_{\bar{\boldsymbol{p}}}) \,. \tag{4.11}$$

In order to further simplify the expression, and eventually lead to a formulation that facilitates iterative processing, Bayes' theorem is applied to the a posteriori probability on the right-hand side of equation (4.11). Under the assumption that $\Omega_{\boldsymbol{p}}$ and $\Omega_{\bar{\boldsymbol{p}}}$ are independent and equiprobable, i.e., all image patches are expected to be observed with the same frequency, the a posteriori probability can be expressed as

$$P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}} \mid \Omega_{\boldsymbol{p}}, \Omega_{\bar{\boldsymbol{p}}}) = P(\Omega_{\boldsymbol{p}}, \Omega_{\bar{\boldsymbol{p}}} \mid \boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}}) \times P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}}) \,, \tag{4.12}$$

where $P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}} \mid \Omega_{\boldsymbol{p}}, \Omega_{\bar{\boldsymbol{p}}})$ is the likelihood that $\bar{\boldsymbol{p}}$ is the match for $\boldsymbol{p}$ given the support windows $\Omega_{\boldsymbol{p}}$ and $\Omega_{\bar{\boldsymbol{p}}}$, and $P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}})$ is the a priori probability that $\bar{\boldsymbol{p}}$ is the match for $\boldsymbol{p}$. Because $\Omega_{\boldsymbol{p}}$ and $\Omega_{\bar{\boldsymbol{p}}}$ reside in high-dimensional probability spaces, it is computationally intractable to evaluate their probabilities. Therefore, it is often assumed that pixels located within these support windows are pairwise-independent and the likelihood is approximated by

$$P(\Omega_{\boldsymbol{p}}, \Omega_{\bar{\boldsymbol{p}}} \mid \boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}}) \approx \prod_{\boldsymbol{q} \in \Omega_{\boldsymbol{p}},\, \bar{\boldsymbol{q}} \in \Omega_{\bar{\boldsymbol{p}}}} P(\boldsymbol{q}, \bar{\boldsymbol{q}} \mid \boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}}) \,. \tag{4.13}$$

Combining the approximations given in (4.12) and (4.13) with the matching criteria of (4.11) yields

$$m(\boldsymbol{p}) \approx \underset{\bar{\boldsymbol{p}} \in S_{\boldsymbol{p}}}{\operatorname{argmax}} \prod_{\boldsymbol{q} \in \Omega_{\boldsymbol{p}},\, \bar{\boldsymbol{q}} \in \Omega_{\bar{\boldsymbol{p}}}} P(\boldsymbol{q}, \bar{\boldsymbol{q}} \mid \boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}}) \times P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}}) \,. \tag{4.14}$$

Stereo matching is typically performed by using an additive distance metric, arbitrarily denoted by $\delta(\boldsymbol{q}, \bar{\boldsymbol{q}})$, to measure the dissimilarity between pixels $\boldsymbol{q} \in \Omega_{\boldsymbol{p}}$ and $\bar{\boldsymbol{q}} \in \Omega_{\bar{\boldsymbol{p}}}$. This is equivalent to approximating the probability distribution of the pixel likelihoods by

$$P(\boldsymbol{q}, \bar{\boldsymbol{q}} \mid \boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}}) \approx \exp\Big(-\delta(\boldsymbol{q}, \bar{\boldsymbol{q}})\Big)\,. \tag{4.15}$$

Substituting this approximation into (4.14) results in

$$m(\boldsymbol{p}) \approx \underset{\bar{\boldsymbol{p}} \in S_{\boldsymbol{p}}}{\operatorname{argmax}} \prod_{\boldsymbol{q} \in \Omega_{\boldsymbol{p}},\, \bar{\boldsymbol{q}} \in \Omega_{\bar{\boldsymbol{p}}}} \exp\Big(-\delta(\boldsymbol{q}, \bar{\boldsymbol{q}})\Big) \times P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}})\,. \tag{4.16}$$

Taking the logarithm of this expression eliminates the exponential operation, and negating the result leads to

$$m(\boldsymbol{p}) \approx \underset{\bar{\boldsymbol{p}} \in S_{\boldsymbol{p}}}{\operatorname{argmin}} \sum_{\boldsymbol{q} \in \Omega_{\boldsymbol{p}},\, \bar{\boldsymbol{q}} \in \Omega_{\bar{\boldsymbol{p}}}} \delta(\boldsymbol{q}, \bar{\boldsymbol{q}}) - \log P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}})\,. \tag{4.17}$$

Replacing the arbitrary distance metric $\delta(\boldsymbol{q}, \bar{\boldsymbol{q}})$ with the adaptive support-weight matching cost of equation (4.2) results in a new matching criteria given by

$$m(\boldsymbol{p}) \approx \underset{\bar{\boldsymbol{p}} \in S_{\boldsymbol{p}}}{\operatorname{argmin}}\, C(\boldsymbol{p}, \bar{\boldsymbol{p}}) - \log P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}})\,. \tag{4.18}$$

For non-iterative stereo matching methods, there exist no disparity estimates prior to matching. Consequently, the additive term $-\log P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}})$ in the matching criteria in (4.18) is constant for all $\boldsymbol{p}$ and $\bar{\boldsymbol{p}}$, and can be ignored since it does not affect the minimization. The nature of iterative methods allows them to incorporate the additive term by considering the disparity estimates produced after the first iteration of stereo matching. In the following, a method is given for iterative disparity refinement that uses an approximation of the additive term $-\log P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}})$.

Let $D^i(\boldsymbol{p})$ be the disparity estimate for pixel $\boldsymbol{p}$ obtained in the i^{th} iteration of matching and let $F^i(\boldsymbol{p}) \in [0,1]$ be the fractional decrease from the second lowest matching cost to the minimum matching cost. $F^i(\boldsymbol{p})$ is used to express the confidence level associated with the disparity estimate of pixel $\boldsymbol{p}$. Essentially, the confidence level rates the uniqueness of the minimum-cost match obtained with the WTA approach by comparing its cost against the nearest competitor. Once the first iteration of stereo matching is complete, disparity estimates along with confidence levels can be used to guide matching in subsequent iterations. This is done by adding a cost penalty to candidate disparities that deviate from their expected values.

Using the disparity estimate $D^{i-1}(\boldsymbol{p})$ from the previous iteration, the probability that pixel $\bar{\boldsymbol{p}}$ is the correct match for pixel $\boldsymbol{p}$ in the current iteration is approximated by

$$P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}}) \approx \exp\Big(-\alpha \times \Big|D^{i-1}(\boldsymbol{p}) - D^i(\boldsymbol{p})\Big|\Big), \tag{4.19}$$

where the scalar α is determined empirically. Thus, the approximation of the additive term $-\log P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}})$ is given by

$$-\log P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}}) \approx \alpha \times \Big|D^{i-1}(\boldsymbol{p}) - D^i(\boldsymbol{p})\Big|. \tag{4.20}$$

Note that the approximation in (4.20) only considers a single disparity estimate from the previous iteration computed exclusively for the pixel of interest and does not facilitate message passing between pixels. To allow for the exchange of disparity information, a revised estimate of the additive term is obtained using the adaptive support weights and the confidence levels of the disparity estimates in the support

window of $\boldsymbol{p}$. This results in (4.20) being reformulated as

$$-\log P(\boldsymbol{p} \leftrightarrow \bar{\boldsymbol{p}}) \approx \alpha \times \left| \frac{\sum\limits_{\boldsymbol{q} \in \Omega_{\boldsymbol{p}} \setminus \boldsymbol{p}} w(\boldsymbol{p}, \boldsymbol{q}) F^{i-1}(\boldsymbol{q}) D^{i-1}(\boldsymbol{q})}{\sum\limits_{\boldsymbol{q} \in \Omega_{\boldsymbol{p}} \setminus \boldsymbol{p}} w(\boldsymbol{p}, \boldsymbol{q}) F^{i-1}(\boldsymbol{q})} - D^{i}(\boldsymbol{p}) \right| . \tag{4.21}$$

To discourage pixel disparities from deviating from their predicted values, a cost penalty is added to the adaptive support-weight matching cost. The penalty is a weighted sum of the additive terms, as defined in (4.21), over the support region of $\boldsymbol{p}$. The weighted sum is calculated using both the similarity of neighboring pixels and their respective confidence levels, using

$$\Lambda^{i}(\boldsymbol{p}, \bar{\boldsymbol{p}}) = \alpha \times \sum_{\boldsymbol{q} \in \Omega_{\boldsymbol{p}} \setminus \boldsymbol{p}} w(\boldsymbol{p}, \boldsymbol{q}) F^{i-1}(\boldsymbol{q}) \left| D^{i-1}(\boldsymbol{q}) - D^{i}(\boldsymbol{p}) \right| . \tag{4.22}$$

Note that this cost penalty is similar to the smoothness constraint used by the nonlinear diffusion technique described in [106]; however, the proposed formulation incorporates the confidence term, uses color similarity to enforce disparity continuity, and the penalty is evaluated using a broad support region instead of using only the adjacent pixels. Moreover, the central pixels in each support region are not considered when generating disparity estimates and cost penalties in (4.21) and (4.22), which prevents erroneous matches from reinforcing themselves.

Finally, the penalty is incorporated into iterative disparity refinement for stereo matching. Denoting the adaptive support-weight matching cost of equation (4.2) as $C^{0}(\boldsymbol{p}, \bar{\boldsymbol{p}})$, the matching cost in subsequent iterations is calculated using

$$C^{i}(\boldsymbol{p}, \bar{\boldsymbol{p}}) = C^{0}(\boldsymbol{p}, \bar{\boldsymbol{p}}) + \Lambda^{i}(\boldsymbol{p}, \bar{\boldsymbol{p}}) , \tag{4.23}$$

for every pair of pixels $\boldsymbol{p}$ and $\bar{\boldsymbol{p}}$ that are matching candidates.

After the matching costs are computed, the minimum-cost matches are found for both the reference and target images using the WTA decision criteria given in equation (4.3) and by substituting the updated cost $C^i(\boldsymbol{p}, \bar{\boldsymbol{p}})$ for $C(\boldsymbol{p}, \bar{\boldsymbol{p}})$. The resulting matches are then used to construct disparity maps associated with both images. If $\bar{\boldsymbol{p}} = m(\boldsymbol{p})$ and $\boldsymbol{p}' = m(\bar{\boldsymbol{p}})$, disparity $D^i(\boldsymbol{p})$ is assigned to the reference disparity map and disparity $D^i(\boldsymbol{p}')$ is assigned to the target disparity map. Because this back-and-forth mapping is not always one-to-one, the following procedure is applied to the resulting disparity maps to determine if the two disparity maps are consistent. Pixel $\boldsymbol{p}$ is deemed inconsistent if $|D^i(\boldsymbol{p}) - D^i(\boldsymbol{p}')| > 1$, and if so, its confidence $F^i(\boldsymbol{p})$ is set to zero. The confidence metric produced in every iteration of refinement is therefore given by

$$F^i(\boldsymbol{p}) = \begin{cases} \dfrac{\min\limits_{\bar{\boldsymbol{p}} \in S_{\boldsymbol{p}} \setminus m(\boldsymbol{p})} \{C^i(\boldsymbol{p}, \bar{\boldsymbol{p}})\} - \min\limits_{\bar{\boldsymbol{p}} \in S_{\boldsymbol{p}}} \{C^i(\boldsymbol{p}, \bar{\boldsymbol{p}})\}}{\min\limits_{\bar{\boldsymbol{p}} \in S_{\boldsymbol{p}} \setminus m(\boldsymbol{p})} \{C^i(\boldsymbol{p}, \bar{\boldsymbol{p}})\}}, & \text{if } |D^i(\boldsymbol{p}) - D^i(\boldsymbol{p}')| > 1 \\ 0, & \text{otherwise} \end{cases} . \quad (4.24)$$

Similarly, this back-and-forth mapping is also used to determine inconsistencies in the disparities of the matched image.

What is important, is that that the two-pass approximation of the bilateral filter, which is necessary in both the cost aggregation and disparity refinement, can be evaluated independently at individual pixels. At the same time, the image data used to evaluate the filter can be shared between spatially connected pixels. The possibility to process pixels independently while reusing image data makes the proposed method well-suited for parallel implementation on graphics hardware. As will be shown later, by combining the two-pass approximation of the bilateral filter in cost aggregation

with a low-complexity iterative disparity refinement technique implemented on high-performance graphics hardware, the proposed method is able to achieve a high level of accuracy while maintaining real-time operation. The next section covers the NVIDIA CUDA programming framework that enabled the implementation of the proposed method on graphics hardware.

4.5 Overview of the NVIDIA CUDA Framework

Announced in the late 2006, the Compute Unified Device Architecture (CUDA) is a general-purpose parallel computing architecture and programming framework implemented in NVIDIA's graphics processing hardware, which includes the GeForce, Quadro and Tesla product families. The framework delivers an application programming interface that enables developers to access the instruction set and memory of the Graphics Processing Unit (GPU) via calls to provided software libraries or directly through compiler directives and extensions to industry-standard programming languages. Since its first release, the CUDA framework has been widely used as a tool for the implementation of real-time image processing algorithms, and has found its applications in scientific computing, including computational biology, data compression and encryption, data mining, and physics simulations.

4.5.1 The CUDA Execution Model.

The CUDA execution model complements the design of modern massively parallel, multithreaded, manycore graphics processors. Unlike conventional general purpose

CPUs, the majority of circuitry within GPUs serves purely data processing purposes, while flow control and data caching subsystems are minimal. This makes GPUs well-suited for heavy data processing tasks extending beyond their original application in accelerated rendering of 2D and 3D graphics.

Figure 4.5: Simplified architecture of the NVIDIA GK110 streaming multiprocessor. A single GK110 series die contains 192 CUDA cores specialized in single-precision arithmetic, 64 double-precision units (DP), 32 Load/Store (LD/ST) and Special Functions Units (SFU).

A typical CUDA-compatible GPU embodies up to 14 streaming multiprocessors, each of which is essentially a collection of single-precision floating point arithmetic logic units (double-precision coprocessors are optional), on-chip memory, and interfaces to off-chip memory sources along with the necessary controlling hardware. An illustration of the architecture implemented in the recent NVIDIA GPUs is shown in Figure 4.5.

The streaming multiprocessors are available to the programmer through the CUDA application programming interface, and accessing their computational resources is a four-step process:

1. First, the programmer needs to encapsulate data processing routines in a special function known as a *kernel*. Invoked from the operating system of the controlling machine or *host*, kernels are executed across many parallel threads on the graphics card, hereafter referred to as the *device*.

2. Since kernels cannot in general operate directly on the host memory, device memory must be allocated and data must be copied from the host to the device prior to executing a kernel. Besides the off-chip random access memory, the device offers on-chip registers, which are thread-private, and fast on-chip shared memory that can be accessed by any thread within a block. Different types of device memory are covered in Section 4.5.2.

3. In order to invoke a kernel, the programmer needs to specify a configuration of threads that, once spawned on the device, will perform data manipulation tasks. CUDA threads are organized in 2D or 3D blocks, where each thread is uniquely identified and indexed. A single thread can thus be thought of as an abstract representation of a scalar arithmetic processor, whereas a block of threads virtualizes a stream multiprocessor composed of many scalar arithmetic processors. Similarly, thread blocks are organized in a 2D or 3D grid, where each block has its unique index. Thread and block indexing is a vital part of the CUDA specification and is extensively used by the programmers for the purpose

of mapping individual threads to input/output data.

4. When the kernel call completes, host and device programs can be synchronized and the data can be transferred back to the operating memory of the host.

Upon invocation of a kernel, the graphics hardware takes full control of the execution process. The warp schedulers shown in Figure 4.5 are responsible for organizing threads into groups of 32 called *warps*, to which instructions are issued through dispatch units. Each streaming multiprocessor incorporates 4 warp schedulers and 8 dispatching units (each of which relays instructions to a single half-warp), making it possible to concurrently execute 4 thread warps. The execution follows the Single Instruction Multiple Threads (SIMT) model that guarantees that individual instructions are executed in parallel across all 32 threads in a warp, as long as the threads do not experience divergence due to the presence of branching instructions. In the later case, the code branches are executed sequentially, however, intra-branch instruction parallelism is maintained.

The multiprocessors have the ability to choose which warps to issue instructions to that can be thought of as a context switching mechanism. However, unlike traditional CPU context switching occurring in multitasking operating systems, the state of thread execution does not need to be stored/restored, since the variables allocated by the threads/blocks (whether in registers, shared or global memory) exist at least as long as the individual threads.

Memory type	Location	Access Type	Cached	Lifetime	Scope
Texture memory	off-chip	read-only	yes	application	grid
Global memory	off-chip	read/write	yes	application	grid
Registers	on-chip	read/write	n/a	kernel	thread
Shared memory	on-chip	read/write	n/a	kernel	block
Constant memory	off-chip	read-only	yes	application	grid

Table 4.1: CUDA memory types and their characteristics.

4.5.2 The GPU Memory Hierarchy.

As previously indicated, graphics cards come equipped with several types of memory providing specialized data storages spaces, which most generally can be classified as on- or off-chip memory. These two classes of memory serve fundamentally different purposes and differ in access latency, availability of caching, and scope and lifetime of variables. Understanding the differences between these memory types and their proper usage is a key to achieving high performance of computations. CUDA memory types and their basic characteristics are compared in Table 4.1.

Global memory, which is composed of high-frequency DDR memory modules located outside of the graphics processing unit, serves as the primary random-access data storage with capacity measured in gigabytes, which makes it suitable for storing large arrays or volumes of data. Variables allocated in the global memory remain available throughout the lifetime of the application, and can be accessed by any block of threads being executed by any of the streaming multiprocessors. The off-chip

location of global memory necessitates that communication between multiprocessors and global memory be carried over a data bus. The addressing overhead introduced by the data bus and the latency of the DDR modules put global memory access latency in the range of 400 - 600 clock cycles.

Due to the DDR modules operating in data bursting mode, global memory is read or written in chunks of 32, 64 or 128 bytes in what is referred to as memory transactions. As a consequence, if threads being executed attempt to access a sequence of variables allocated continuously within a memory region of 32, 64 or 128 bytes, data retrieval can be *coalesced* into as little as one memory transaction, as long as the first requested memory index is a multiple of 32 bytes. Conversely, reading a of set variables that are scattered in global memory requires many transactions to be issued, resulting in wasted memory bandwidth.

In situations when the data access pattern prevents full coalescing, *texture* and *surface memory* constitute a feasible alternative. Rather than being separate physical memory types, textures and surfaces occupy global memory and often utilize opaque memory layouts known as CUDA arrays, which lend themselves well to spatial caching. Texture and surface memory are accessed through GPU's texturing units that enable additional functionality, including interpolation of adjacent data values, normalization of data values or array indices, and boundary handling. These access benefits make textures and surfaces particularly useful in image processing tasks; lookup tables are often implemented using textures. Textures allow for read-only access, while surfaces can be written to.

Registers are the fastest available memory type, with access latency 100 times lower

than global memory. Incorporated into the GPU die, registers are designed to store scalar variables or smaller arrays as long as the indexing can be computed at compile time; registers do not support random access. In contrast to variables allocated in global memory, register variables can be read or written exclusively by the thread that allocated them, and their lifetime is determined by the lifetime of individual threads. The amount of registers available is limited per block, and thus per thread. If threads within a block reach the register allocation limit, the compiler automatically moves allocation of variables into global memory, which effectively decreases memory access performance.

Shared memory is another type of on-chip memory that can be used to facilitate the exchange of data between threads within a block. Instantiated during invocation of a kernel, shared memory supports simultaneous reads/writes from any thread in a block without the need for synchronization. Given that no two threads write to the same memory bank, shared memory attains access latencies similar to those of registers. This type of memory is particularly useful in scenarios where threads exhibit high locality (both spatial and temporal) of global memory references. An example of such a scenario is any signal filtering task where threads within a block operate on substantially overlapping windows of data. In such instances, regions of data can be preloaded from global memory to shared memory to enable fast and repetitive data access by the threads, and thus minimizing the number of global memory reads. Similarly to the case of registers, the amount of shared memory is limited per block.

In addition to the previously described memory types, CUDA-compatible devices offer *constant memory* that is a 64KB address space within global memory that

can only be read from. This makes constant memory a feasible option for storing arguments that are not changed during the execution of kernels. Variables in constant memory space behave essentially like global memory variables in that they exist over the lifetime of the application and can be accessed by any block of threads, yet accesses to constant memory benefit greatly from caching. Once the cache is filled after initial reads, accessing constant variables can be as fast as reading from registers, and subsequent data retrieval from constant memory only occurs in case of cache-miss events.

4.6 Real-time Stereo Matching on CUDA

The implementation of the proposed method utilizes the NVIDIA GeForce Titan Black graphics card, equipped with 2880 CUDA cores and 6GB of DDR5 global memory. To fully utilize the computational capabilities of the Titan Black graphics card, it is imperative to decompose the algorithm in a way that provides high occupancy of the streaming multiprocessors, measured as a ratio of the number of active warps and the maximum number of active warps per multiprocessor. High occupancy enables the multiprocessors to take advantage of context switching, in that the execution of thread warps awaiting for data retrieval from global memory can be put on hold, allowing for the execution of the remaining warps. This way the multiprocessors are able to maintain high usage of the memory bus that, when combined with context switching, aids in reducing the latency associated with global memory access. Note that multiprocessor occupancy is strictly determined by the usage of shared memory

and registers per block, since these resources are limited.

In the following section, the decomposition of the proposed method into CUDA kernels is discussed and relevant implementation details of individual kernels are given. Additionally, the complexity of the core stereo matching operations is analyzed; the complexity analysis is supplemented with profiling data demonstrating the performance of each kernel. Finally, the CUDA implementation of the proposed method is compared against a sequential implementation and speedups attained by individual operations are computed.

4.6.1 Algorithm Decomposition and Implementation

The proposed method was designed to process RGBA stereo images with 32 bits per pixel (bpp) color depth. While the alpha channel remains unused, the 32bpp RGBA images fulfill the alignment constraints necessary for fully coalesced access, which is not always possible with the commonly used 24bpp RGB images. The input images are stored in global memory with the option of access through textures in situations when the access pattern is highly irregular, or when border handling functionality is needed.

The flow diagram in Figure 4.6 illustrates the organization of computations within the proposed stereo matching algorithm. The processes shown reflect the decomposition of the algorithm into CUDA kernels, with the exception of the cost filtering and disparity refinement operations, that for performance reasons have been further decomposed into two steps (horizontal and vertical) and implemented in separate

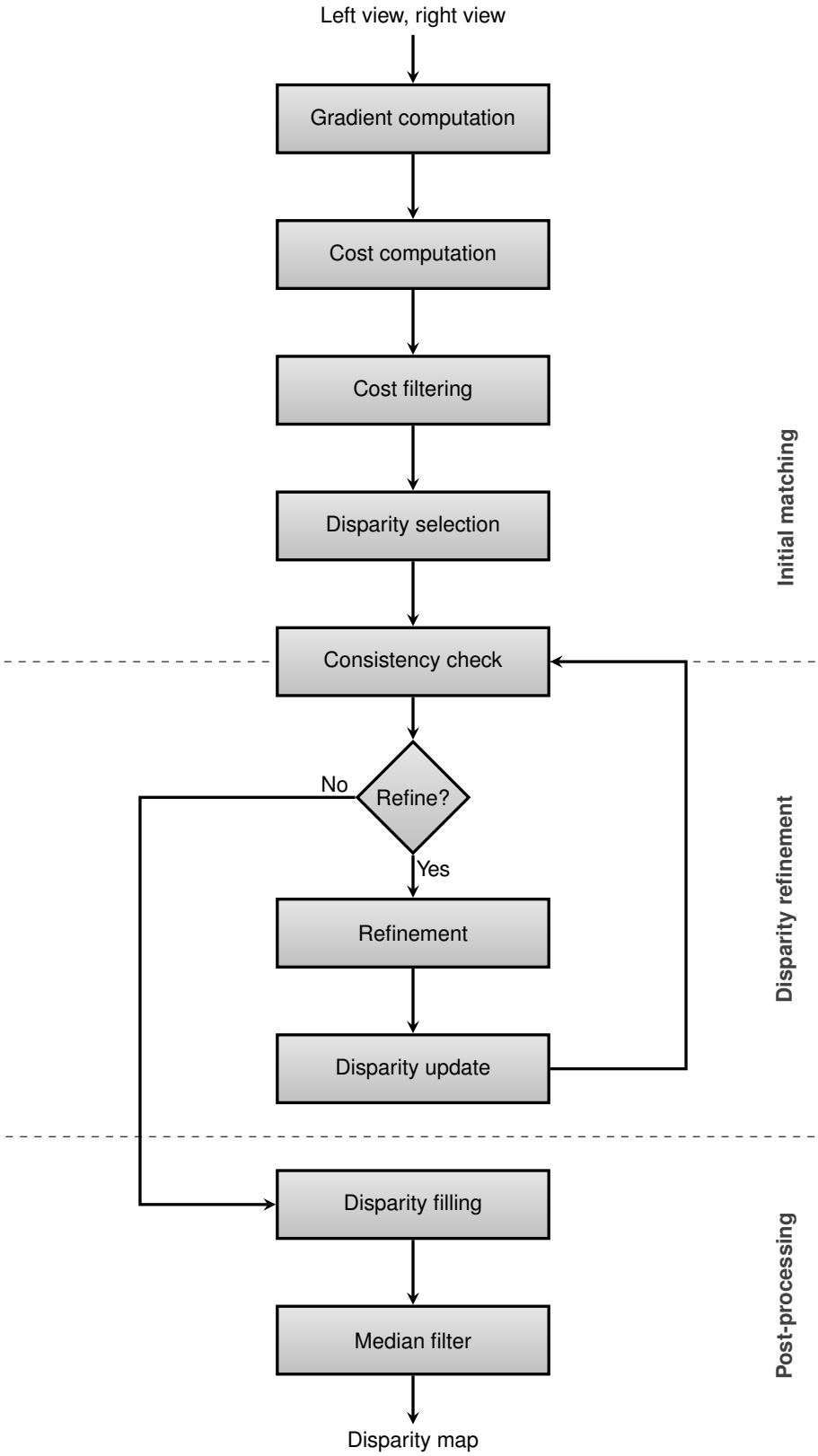


Figure 4.6: The order of operations within the proposed stereo matching method.

kernels. This scheme allows for the computations to be carried with pixel-wise granularity, i.e., all kernels are designed such that each thread within the active block performs operations necessary to evaluate the set of disparity hypotheses at a single pixel. The key benefit associated with choosing such a granularity is that it leads to regularization of global memory access patterns. This is a result of the fact that threads in a warp are responsible for processing adjacent pixels, and will typically request subsequent memory addresses. With a properly selected block size, this scheme enables the majority of kernels to access global memory in a fully coalesced way.

The algorithm begins by computing gradients of both images and then forms the matching cost volume that contains raw (unfiltered) cost metrics evaluated for all disparity hypotheses at every pixel of the reference image. In particular, a single layer of the cost volume is a collection of cost values associated with an individual disparity hypothesis. Next, cost values are aggregated through the application of the two-pass approximation of the joint bilateral filter to the subsequent layers of the cost volume. The cost filtering process is crucial to the accuracy of matching and also largely affects the overall computational performance of the algorithm.

Since there exists significant overlap in pixels accessed by adjacent threads in both vertical and horizontal steps of cost filtering, each block allocates extended windows of shared memory for storing guidance data and input cost values in order to reduce the latency associated with repeated global memory reads. The concept of the extended shared memory window is shown in Figure 4.7. Immediately after invocation of the cost filtering kernel, shared memory is populated with pixel intensities of the guidance image necessary to evaluate the filter of a chosen radius, which includes the region

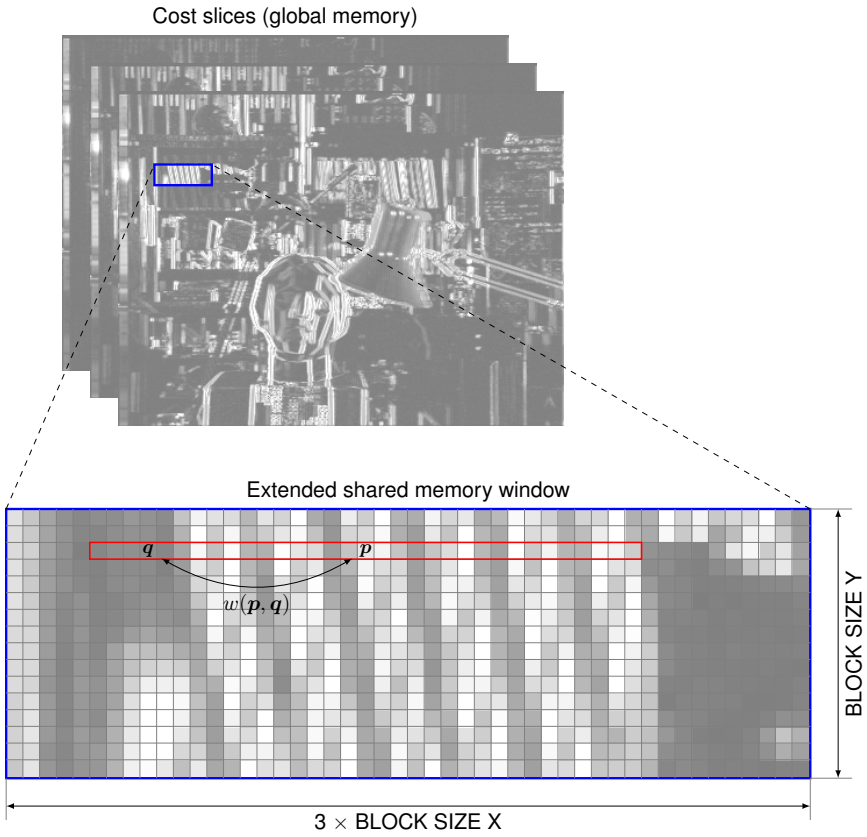


Figure 4.7: The concept of the extended shared memory window in the horizontal step of cost filtering. The region highlighted in red contains cost values processed by the thread associated with pixel $\boldsymbol{p}$. The weigh $w(\boldsymbol{p}, \boldsymbol{q})$ is computed based on the guidance data preloaded into an analogous shared memory window.

corresponding to the thread block and the surrounding pixels located within the filter radius. The filtering kernels are designed to process the cost volume in a layer-by-layer manner. Once fetched, guidance data remains unchanged, while the cost values are loaded into shared memory and filtered as the threads sweep through subsequent layers of the cost volume.

Upon completion of the cost filtering, disparity selection and consistency check kernels are executed to conclude the initial matching stage. The disparity selection

kernel implements the Winner-Take-All matching criteria by iterating over all disparity hypotheses and choosing those corresponding to the minimum matching cost values for both left and right views; the disparity selection kernel is also responsible for evaluating confidence levels associated with the chosen disparities. Note that although the cost volume is constructed with respect to the reference view, it contains sufficient information to generate disparity maps for both views. The complementary consistency check kernel performs cross-checking of disparity assignments and zeroes confidence levels linked with inconsistent pixels. Since memory access patterns within the consistency check operation depend on individual disparity values, the corresponding kernel is the only kernel in the CUDA implementation of the proposed method that cannot take full advantage of memory coalescing.

The cost volume, disparity maps and respective confidence levels are supplied as inputs to the iterative refinement stage. The refinement stage is a combination of three kernels, out of which two generate disparity estimates and the third one re-evaluates disparity hypotheses based on these precomputed disparity estimates. Disparity estimates are generated for both the reference and target images by evaluating (4.21) in a two-pass (vertical-horizontal) approach, similar to the method used for aggregation of matching costs. The CUDA kernels responsible for computing disparity estimates are outlined in Algorithms 4.1 and 4.2. To facilitate later evaluation of the cost penalty function, both the adaptively weighted sums of disparities and the corresponding normalizing factors, i.e., the disparity estimate and the denominator in (4.21), are stored in global memory. Observe that the horizontal step of disparity estimation does not rely on confidence values, since these are integrated into the intermediate

results obtained from the vertical step.

1: map the active thread to the corresponding pixel $\boldsymbol{p} = (x, y)$

2: $Num(\boldsymbol{p}) \leftarrow 0$ // denominator in (4.21)

3: $\hat{Den}(\boldsymbol{p}) \leftarrow 0$ // numerator in (4.21)

4: **for all** pixels $\boldsymbol{q} \in \{(x, y-r), \ldots, (x, y+r)\} \setminus \{(x, y)\}$ **do** // r is the filter radius

5: $N(\boldsymbol{p}) \leftarrow N(\boldsymbol{p}) + w(\boldsymbol{p}, \boldsymbol{q}) F^{i-1}(\boldsymbol{q}) D^{i-1}(\boldsymbol{q})$

6: $D(\boldsymbol{p}) \leftarrow D(\boldsymbol{p}) + w(\boldsymbol{p}, \boldsymbol{q}) F^{i-1}(\boldsymbol{q})$

7: **end for**

8: $\hat{D}_{est}(\boldsymbol{p}) \leftarrow \frac{Num(\boldsymbol{p})}{Den(\boldsymbol{p})}$ // intermediate disparity estimate

Algorithm 4.1: A single iteration of disparity refinement: vertical step.

The disparity update kernel implements a variant of the Winner-Take-All operation that applies a penalty to the matching costs when assigning disparities and computing confidence levels (see (4.22) and (4.23) for details). An important property of this operation is that both the penalty terms and the penalized costs are calculated and processed locally as the algorithm iterates over disparity hypotheses, thus eliminating the need to store the penalty values or overwrite the original cost values. The disparity update kernel is summarized in Algorithm 4.3. Prior to advancing to the next iteration of refinement, disparity maps for both views are verified using the consistency checking kernel introduced in the initial matching stage. Likewise, the confidence of inconsistent pixels is set to zero, effectively preventing them from affecting disparity estimation in the next iteration.

Finally, the disparity map associated with the reference view is subjected to post-

1: map the active thread to the corresponding pixel $\boldsymbol{p} = (x, y)$

2: $Num(\boldsymbol{p}) \leftarrow 0$

3: $Den(\boldsymbol{p}) \leftarrow 0$

4: **for all** pixels $\boldsymbol{q} \in \{(x - r, y), \ldots, (x + r, y)\} \setminus \{(x, y)\}$ **do**

5: $\quad Num(\boldsymbol{p}) \leftarrow w(\boldsymbol{p}, \boldsymbol{q}) \hat{Den}(\boldsymbol{q}) \hat{D}_{est}(\boldsymbol{q})$

6: $\quad Den(\boldsymbol{p}) \leftarrow w(\boldsymbol{p}, \boldsymbol{q}) \hat{Den}(\boldsymbol{q})$

7: **end for**

8: $D_{est}(\boldsymbol{p}) \leftarrow \frac{Num(\boldsymbol{p})}{\hat{Den}(\boldsymbol{p})}$ // final disparity estimate

Algorithm 4.2: A single iteration of disparity refinement: horizontal step.

processing through repeated, alternating applications of a median filter and a disparity fill operation. The 3×3 median filter is used to eliminate spurious noise artifacts, i.e., isolated mismatches, from the disparity map with minimal alteration of object boundaries. A 9-input sorting network is implemented that requires 19 comparisons to efficiently find the median within each 3×3 region. The disparity fill operation assigns valid disparity values to pixels that have been classified as inconsistent in the last iteration of disparity refinement. These disparity values are generated based on proximity, color similarity and confidence of consistent pixels located within horizontal windows centered at the pixels of interest, similarly to the way disparity estimates are generated in the refinement stage.

1: map the active thread to the corresponding pixel $\boldsymbol{p} = (x, y)$

2: $\text{Min}_1 \leftarrow \infty$

3: $\text{Min}_2 \leftarrow \infty$

4: **for** $d = 0$ to d_{max} **do**

5: $\quad \Lambda \leftarrow \alpha \times Den(\boldsymbol{p}) \times |D_{est}(\boldsymbol{p}) - d|$ // cost penalty in (4.22)

6: $\quad$ **if** $\text{Min}_1 > C(x, y, d) + \Lambda$ **then**

7: $\quad\quad D^i(\boldsymbol{p}) \leftarrow d$

8: $\quad\quad \text{Min}_2 \leftarrow \text{Min}_1$

9: $\quad\quad \text{Min}_1 \leftarrow C(x, y, d) + \Lambda$

10: $\quad$ **else if** $\text{Min}_2 > C(x, y, d) + \Lambda$ **then**

11: $\quad\quad \text{Min}_2 \leftarrow C(x, y, d) + \Lambda$

12: $\quad$ **end if**

13: **end for**

14: $F^i(\boldsymbol{p}) \leftarrow \frac{\text{Min}_2 - \text{Min}_1}{\text{Min}_2}$

Algorithm 4.3: Reassignment of disparities based on penalized matching cost.

4.6.2 Launch Configuration, Complexity and Speedup.

The computational complexity of the proposed method, launch configurations, and execution times of individual kernels are now discussed. Let h be the number of disparity hypotheses, k be the total number of refinement iterations, and m and n be the dimensions of the stereo images. Assume that window sizes in the cost filtering and refinement stages are related by a factor s, that is, a window of size ω is used

for cost filtering and a window of size $s\omega$ is used for refinement of disparities. With p threads operating in parallel, the complexity of computing and filtering the cost volume is $\mathcal{O}(mn\omega h/p)$, i.e., cost values are computed for each of h disparity hypotheses at each of $m \times n$ pixels that are aggregated using a two-pass approximation of the bilateral filter with a window of size ω; the division operation signifies the parallelism of computations using p threads. Alternatively, the complexity of iterative refinement is $\mathcal{O}(mns\omega k/p)$. Thus, the increase in complexity associated with incorporating iterative refinement into the stereo matching framework based on cost filtering is sk/h. It has been determined that $k = 3$ iterations of refinement are adequate for convergence of the error rate, while the number of disparity hypotheses is typically much larger. In practical CUDA implementation $s \in \{1, 2\}$, that is, cost filtering and refinement are performed either using support windows of the same size, or the window in the refinement stage is twice the size of that used for cost filtering.

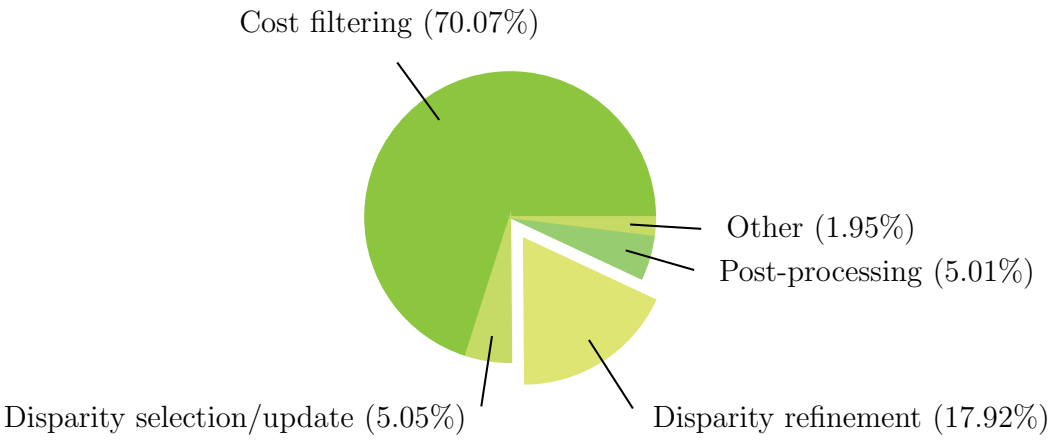


Figure 4.8: Distribution of execution times of stereo matching operations.

Execution profiles of individual kernels were obtained by profiling the proposed

Table 4.2: Decomposition of the proposed stereo matching method into CUDA kernels and their corresponding execution profiles. The profiling data were obtained by processing a 640×480 stereo image pair on the NVIDIA GeForce Titan black graphics card.

Kernel	Calls	Block size	Time (sec.)[1]	Total time	Time plot[2]
`ComputeGradient`	2	32×32	0.024	0.048	
`ComputeCost`	1	16×16	0.521	0.521	
`CostFilterColumns`	1	16×16	11.838	11.838	
`CostFilterRows`	1	16×16	12.277	12.277	
`WinnerTakesAll`	1	32×32	0.406	0.406	
`ConsistencyCheck`	4	32×32	0.025	0.1	
`RefinementColumns`	6	32×8	0.546	3.28	
`RefinementRows`	6	16×32	0.481	2.89	
`DisparityUpdate`	3	32×32	0.444	1.332	
`DisparityFill`	3	32×32	0.546	1.638	
`MedianFilter`	3	32×32	0.029	0.089	

[1] Average time per kernel invocation.

[2] Dark blue = average time per call, light blue = total time.

stereo matching method on a NVIDIA GeForce Titan Black graphics card. Matching was performed on a pair of 640×480 stereo images, considering 60 disparities, with 3 iterations of disparity refinement. The filter radii were set to 16 for cost filtering, and 32 for disparity refinement. The GeForce Titan Black graphics card allows a maximum of 1024 threads per block, which is equivalent to a block size of 32×32. According to the occupancy calculator supplied with the CUDA framework, choosing this block size results in 100% multiprocessor occupancy in case of simple kernels that do not require significant amounts of shared memory, including gradient computation, disparity selection/update, consistency check and median filtering. However, due to limitations in shared memory available to multiprocessors, kernels that heavily depend on shared memory, such as the ones responsible for cost calculation and filtering, computation of disparity estimates, and disparity filling, benefit from smaller block sizes, allowing them to achieve multiprocessor occupancies in the range of 60% to 85%. Block sizes, numbers of invocations and execution times of specific kernels are listed in Table 4.2. In addition, percentages of the total execution time taken by the key operations of the proposed method are given in Figure 4.8.

The profiling results conform to the complexity analysis. The majority of the processing time, i.e, 70.07%, is consumed by the cost filtering operation, while the iterative disparity refinement only constitutes 17.92% of the total processing time. Note, however, that disparity refinement entails reassignment of disparities in every iteration, which introduces additional processing overhead. Consequently, the disparity selection and update kernels consume 5.05% of the total processing time. When compared to a non-optimized sequential implementation of the proposed method, executed on

Table 4.3: Execution times of both CPU and GPU implementations of the operations within the proposed stereo matching method using 640×480 images with 60 disparity levels and 3 iterations of disparity refinement.

Operation	Execution time (seconds)		Speedup (×)
	CPU[1]	GPU[2]	
Gradient computation	0.010	0.000049	206
Cost computation	0.537	0.000521	1030
Cost filtering	17.365	0.024115	720
Disparity selection	1.234	0.000406	3037
Consistency check	0.128	0.000101	1268
Disparity refinement	4.044	0.006167	655
Disparity update	3.748	0.001332	2813
Disparity fill	0.440	0.001638	268
Median filter	0.008	0.000089	89
	27.51	0.0344	799

[1] AMD Phenom II X6 1075T 3.0 GHz (using a single core).

[2] NVIDIA GeForce Titan Black.

a 3.0GHz AMD Phenom II X6 1075T processor, the GPU implementation attains an overall speedup by a factor of 799, thus reducing the processing time from 27.51 seconds to 0.034 of a second. Speedups achieved by particular GPU operations over their CPU equivalents are given in Table 4.3.

CHAPTER 5

Stereo Performance Evaluation

Still-frame accuracy of stereo matching algorithms is traditionally evaluated using the Middlebury online stereo performance benchmark [12] available online at `http://vision.middlebury.edu/stereo/eval/`. In version 2.0, the benchmark delivers a variety of test stereo image sets with ground truth disparity maps, among which the `tsukuba`, `venus`, `teddy`, and `cones` datasets are most commonly used for evaluation of stereo algorithms. Accuracy of matching is quantified by a ratio of the number of pixels whose disparity value differs from the true disparity by more than a given threshold (typically set to 1), to the total number of pixels. For ease of interpretation this ratio is usually expressed as a percentage of incorrectly assigned disparities. What is important is that each dataset provides occlusion and discontinuity maps, i.e., binary mask images isolating occluded regions or regions near disparity discontinuities, respectively, that enable categorization of errors.

In section 5.1 of this chapter, a procedure is outlined that was used to adjust the parameters of the proposed method for the purpose of stereo performance evaluation

using the Middlebury benchmark. Qualitative and quantitative results obtained using the Middlebury 2.0 datasets follow in section 5.2. In section 5.3, the proposed method is evaluated using an updated methodology and new challenging datasets introduced by the Middbury stereo performance benchmark in version 3.0, whose preliminary release was made available to researchers during writing of this dissertation. Updates to the evaluation methodology from version 2.0, as well as improvements made to the proposed method to better address the new challenges are also discussed in this section. Finally, in section 5.4 the proposed stereo matching method is paired with motion estimation and depth map fusion algorithms to create a structure reconstruction pipeline capable of recovering detailed geometric models of the scene.

5.1 Parameter Tuning

The evaluation methodology used by the Middlebury stereo benchmark requires a consistent set of parameters to be used with all of the standard datasets, allowing only the number of disparity hypotheses under consideration to be adjusted with respect to the distance between the two cameras and the image resolution associated with each dataset. While many stereo matching methods rely on empirically determined parameters, the proposed method has been embedded in a particle swarm optimization (PSO) framework [145, 146], and its parameters are optimized to ensure the lowest possible error rates. The parameters of the proposed method, their symbols, ranges, and recommended values are given in Table 5.1. Note that window sizes together with the numbers of iterations in refinement and post-processing steps were excluded

from optimization. These parameters were fixed and their values have been chosen to enable stereo matching in real time using iterative refinement on the graphics hardware. Precisely, as suggested in Section 4.6.2, window sizes are set to 33×33 for cost filtering, 67×67 for disparity refinement and filling, and 3 iterations of both refinement and post-processing are used. Also note that the cost filtering, refinement, and disparity filling stages have dedicated parameters controlling the strength of shape grouping by proximity and color similarity; this way the shape grouping process can be controlled independently in each of these stages.

Unlike the classic gradient-based optimization techniques, the PSO makes no explicit assumptions regarding the optimization problem. Consequently, for the purpose of parameter optimization the proposed stereo matching method is viewed as a black box accepting the set of parameters marked as tunable in Table 5.1, and computing an error metric characterizing the fitness of the parameter set. The optimizer maintains a population (swarm) of software agents, referred to as particles, that move through the solution search space in order to find the minimum of the error function. Each particle is defined by its position vector $\boldsymbol{x}_i$ representing a candidate solution to the optimization problem, and its velocity vector $\boldsymbol{v}_i$ that governs the movement of the particle. Here, the position vector is of the form $\boldsymbol{x}_i = (\alpha, \tau_c, \tau_g, \sigma_c, \sigma_g, \sigma'_c, \sigma'_g, \sigma''_c, \sigma''_g, \lambda)$, $\boldsymbol{v}_i$ is sized accordingly, and the error function $e(\boldsymbol{x}_i)$ being minimized takes a particular parameter vector as an argument, and quantifies the percentage of incorrectly assigned disparities on the four standard Middlebury datasets. The minimization procedure is delineated as follows:

Table 5.1: Optimized stereo matching parameters and their recommended values.

Parameter	Symbol	Tunable?	Range	Value
Number of disparity hypotheses	h	No	N/A	Image-specific
Window size (cost aggregation)	ω	No	N/A	33
Window size (refinement, disparity filling)	ω'	No	N/A	67
Number of refinement iterations	k	No	N/A	3
Number of post-processing steps	N/A	No	N/A	3
Cost terms balance parameter	α	Yes	[0, 1.0]	0.039979
Color difference truncation value	τ_c	Yes	[0, 255]	45.775986
Gradient difference truncation value	τ_g	Yes	[0, 255]	3.129875
Strength of grouping by similarity (aggr.)	σ_c	Yes	[0, 255]	30.986084
Strength of grouping by proximity (aggr.)	σ_g	Yes	[0, 255]	4.665747
Strength of grouping by similarity (ref.)	σ'_c	Yes	[0, 255]	9.288885
Strength of grouping by proximity (ref.)	σ'_g	Yes	[0, 255]	16.276456
Strength of grouping by sim. (disp. fill.)	σ''_c	Yes	[0, 255]	0.785288
Strength of grouping by prox. (disp. fill.)	σ''_g	Yes	[0, 255]	12.409276
Disparity difference penalty	λ	Yes	[0, 1.0]	0.012458

1. A swarm of N particles is generated. Elements of $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$ are assigned random values uniformly sampled from the corresponding parameter value ranges.

2. Initial positions of individual particles become their best known positions, i.e., $\boldsymbol{p}_i = \boldsymbol{x}_i$. In addition, the swarm's best known position $\boldsymbol{g}$ is obtained as

$$\boldsymbol{g} = \underset{\boldsymbol{x}_i}{\operatorname{argmin}}\, e(\boldsymbol{x}_i)\,. \tag{5.1}$$

3. Velocity vectors are updated according to

$$\boldsymbol{v}_i = w\boldsymbol{v}_i + c_p\phi_p(\boldsymbol{p}_i - \boldsymbol{x}_i) + c_g\phi_g(\boldsymbol{g} - \boldsymbol{x}_i)\,, \tag{5.2}$$

where c_p and c_g are random numbers uniformly distributed on the interval $[0, 1]$ (redrawn with every update), and the parameters w, ϕ_p and ϕ_g are used to control the direction of motion of individual particles. Specifically, the acceleration coefficients ϕ_p and ϕ_g determine the amount of motion towards the particle's best known position and the swarm's best position, respectively, and the inertia weight w regulates the impact of the previous velocity vector on the current one. The corresponding position update is given by

$$\boldsymbol{x}_i = \boldsymbol{x}_i + \boldsymbol{v}_i\,. \tag{5.3}$$

Note that the updated velocity vector may require scaling to prevent elements of the position vector calculated using equation (5.3) from exceeding their bounds.

4. The particle's best known positions are updated in the case that $e(\boldsymbol{x}_i) < e(\boldsymbol{p}_i)$, and the swarm's best known position is set to $\boldsymbol{g} = \boldsymbol{p}_i$ if the condition $e(\boldsymbol{p}_i) < e(\boldsymbol{g})$ is satisfied for any of the particles.

5. Steps 3 and 4 are repeated until no improvements in error rate are observed for a specified number of iterations.

The optimization was performed in two steps, each using $N = 1000$ particles and repetitively running up to 30 iterations of the minimization procedure. In the first step, initial tuning of parameters was performed by setting both acceleration coefficients ϕ_p and ϕ_g to 1.5, and the inertia weight w to 0.25. Such a choice of ϕ_p, ϕ_g, and w provides rapid convergence of particles' positions and prevents particles from getting stuck at sub-optimal solutions. In addition, having ϕ_p and ϕ_g set to equal values balances the amounts of motion towards the particles' and the swarm's best known positions, respectively. Starting with randomly generated position vectors, each of which represents a distinct set of parameters for the proposed stereo matching method, the optimizers were able to successively reduce the associated error rates, i.e., the percentages of incorrectly assigned disparities, to within 5%.

Having observed convergent behavior of both the error rates and the particles' positions, the parameter bounds were tightened and another step of optimization was performed in order to fine-tune the stereo matching parameters. The inertia weight w in the fine-tuning step was decreased to 0.1, resulting in decreased particle velocities, which allowed for the solution space to be searched more thoroughly. Many solutions were found with error rates around 4.5%, among which the values listed in Table 5.1 resulted in the minimum error rate of 4.46%. The convergence of error rates and individual parameter values during subsequent iterations of the fine-tuning is shown in Figure 5.1. Each iteration took approximately 200 seconds to complete.

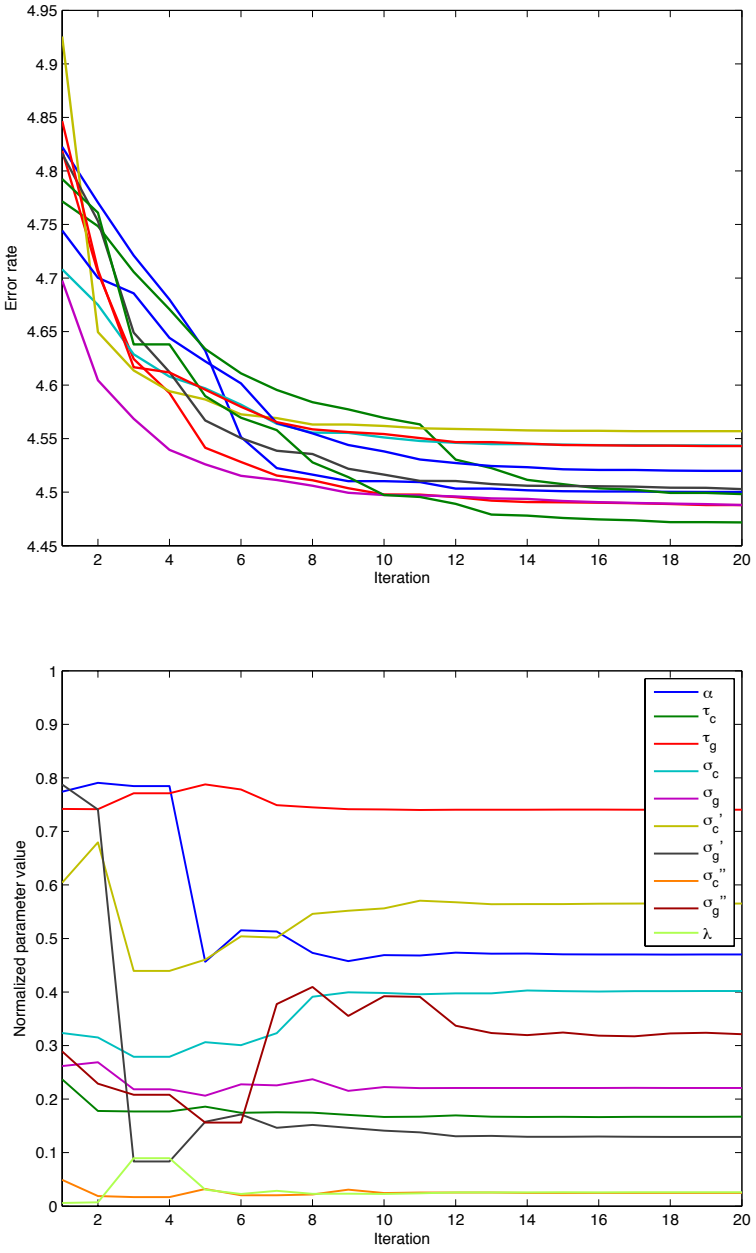


Figure 5.1: Stereo error rate (top) and individual parameters (bottom) plotted versus the iteration number during fine-tuning of the proposed method. Parameter values have been normalized to the interval $[0, 1]$.

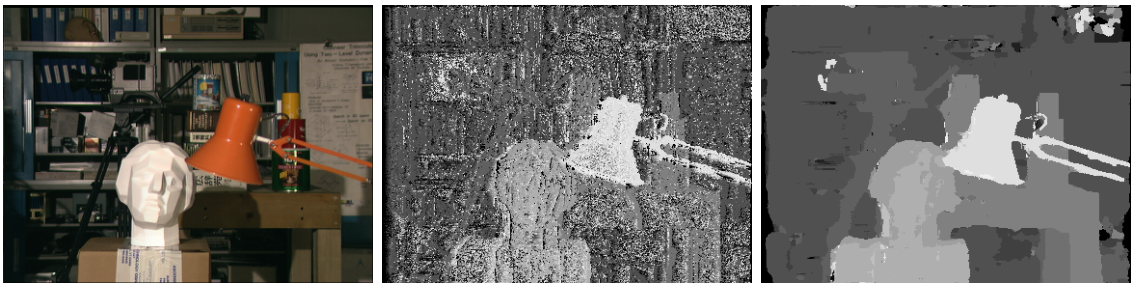
5.2 Evaluation using Middlebury 2.0 Benchmark

To better illustrate the operation of the proposed method, disparity maps obtained using the `tsukuba` dataset at subsequent stages of matching are given in Figure 5.2, along with the initial and refined confidence maps, and an example of disparity estimate generated in the refinement stage. The percentage of incorrectly assigned pixels in non-occluded regions amounts to 39.284% when matches are selected from non-filtered cost metrics, and is successively reduced to 3.174% after cost filtering, 1.15% after the last (3rd) iteration of disparity refinement, and 0.89% after post-processing. The corresponding error rates of the disparity maps obtained in the 1st and the 2nd iteration of refinement, not shown in Figure 5.2, are 1.40% and 1.19%. The reduction in error rate is made possible by penalizing disparities that deviate from the estimates of their similar neighbors, i.e., the expected disparity values, computed in every iteration of refinement.

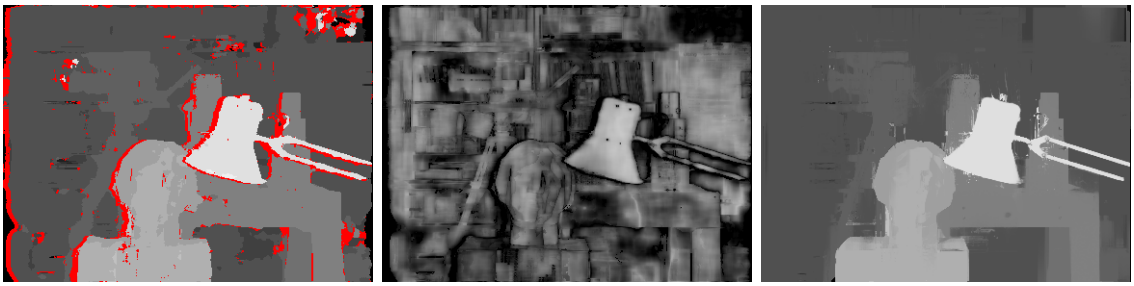
Note that object boundaries within the disparity estimate given in Figure 5.2f are well-defined, whereas the initial disparity map in Figure 5.2c does not always fully capture the shape of objects in the scene. This is a consequence of using different sets of parameters governing the grouping of shapes in the cost filtering and the disparity refinement stages. Specifically, a larger value of the range kernel parameter γ_c poses a less stringent constraint on color similarity, allowing for robust cost aggregation by increasing the contributions of individual pixels within each support region at the expense of slightly decreased accuracy around depth discontinuities. Conversely,

choosing a lower value of the equivalent parameter in the refinement stage provides highly accurate disparity estimates in depth-discontinuous regions that, when combined with penalization of costs, enable the algorithm to overwrite erroneous disparities and improve overall accuracy. The corrective effects of disparity refinement are most evident around the contours of the lamp and the sculpture in the foreground of the `tsukuba` image, as well as nearby the bottom edge of the stand behind the sculpture. Besides directly reducing the error rate, the iterative refinement also adjusts the confidence maps, in that the confidence levels of reliable matches approach the value of 1.0 in successive iterations, which supports propagation of disparity values to nearby pixels. The adjusted confidence maps are used in combination with the adaptive weights to generate disparity values for filling of occlusions and inconsistencies in the post-processing stage, allowing for further accuracy improvements.

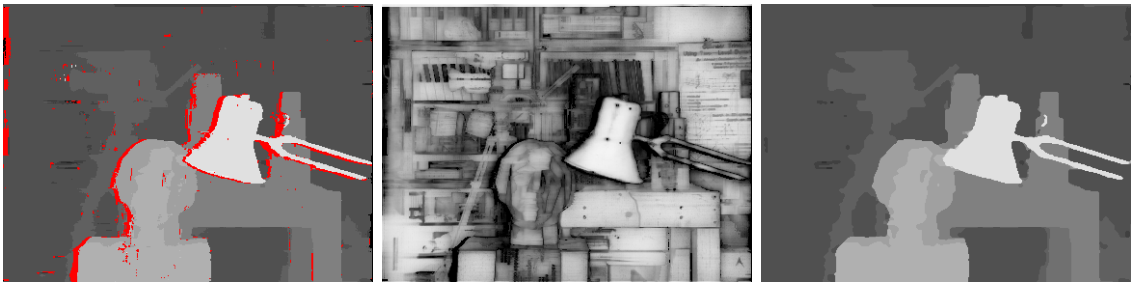
Figure 5.3 shows reference views of the four standard Middlebury stereo datasets along with the ground-truth disparity maps, disparity maps obtained using the proposed method, and the corresponding absolute disparity errors. The error rates and ranks on individual datasets are tabulated in Table 5.2, where the proposed method is compared with the top-performing stereo algorithms listed on the Middlebury benchmark's website. In accordance with the benchmark's evaluation methodology, the error rates are computed for three categories of pixels in each dataset. These categories include pixels in non-occluded regions ("nonocc"), all pixels for which the true disparities are known ("all"), and pixels located around depth discontinuities ("disc"). Additionally, Table 5.3 contrasts the error rates achieved using the proposed method with the error rates of the local stereo methods listed on the Middlebury



(a) Reference image (b) Raw WTA matches (c) After cost filtering



(d) Initial disparity (e) Initial confidence (f) Disp. estimate, 3rd iteration



(g) Disparity, 3rd iteration (h) Confidence, 3rd iteration (i) Post-processed disparity

Figure 5.2: Evolution of the `tsukuba` disparity map during execution stages of the proposed method. Inconsistent pixels are marked in red in figures (d) and (g). Additionally, confidence maps prior to refinement and in the last (3rd) iteration of refinement are shown; white indicates higher confidence values.

benchmark that are capable of operating in real time.

The percentages of incorrectly assigned disparities achieved by the proposed method in non-occluded regions are 0.89% on `tsukuba`, 0.23% on `venus`, 4.85% on `teddy`, and 2.05% on the `cones` dataset; correspondingly, the method ranks 8th, 42nd, 21st, and 2nd in the category of non-occluded pixels. When all three categories ("nonocc", "all" and "disc") are considered, the proposed method achieves an average error rate of 4.46% and an average rank of 21.5, making it the 5th most accurate stereo algorithm among approximately 150 methods listed on the Middlebury benchmark's website at the time of evaluation (March 24, 2014). With an error rate of 4.46%, iterative refinement is also the top-performing method based on local minimization.

Out of the datasets used in the evaluation, lower ranked error rates are observed in the case of the `venus` and `teddy` images. In the `venus` disparity map, errors can be seen around the corner of the plane containing a newspaper page. Due to high similarity of pixel intensities within the corner region, the bilateral filter cannot correctly determine the boundaries of the newspaper plane, which results in unconstrained aggregation of cost metrics, and the method is unable to distinguish some of the lighter, uniformly colored background pixels from the pixels in the newspaper's corner region. Likewise, the strip of darker pixels next to the right edge of the plane containing the infographic is falsely assumed to be a part of the painting in the background. The quantization of disparities, an inherent aspect of matching using local stereo algorithms, is also noticeable in the `venus` disparity map.

In the `teddy` disparity map, errors are concentrated in the region to the left of the teddy bear with the periodic table of elements in the background and in the region

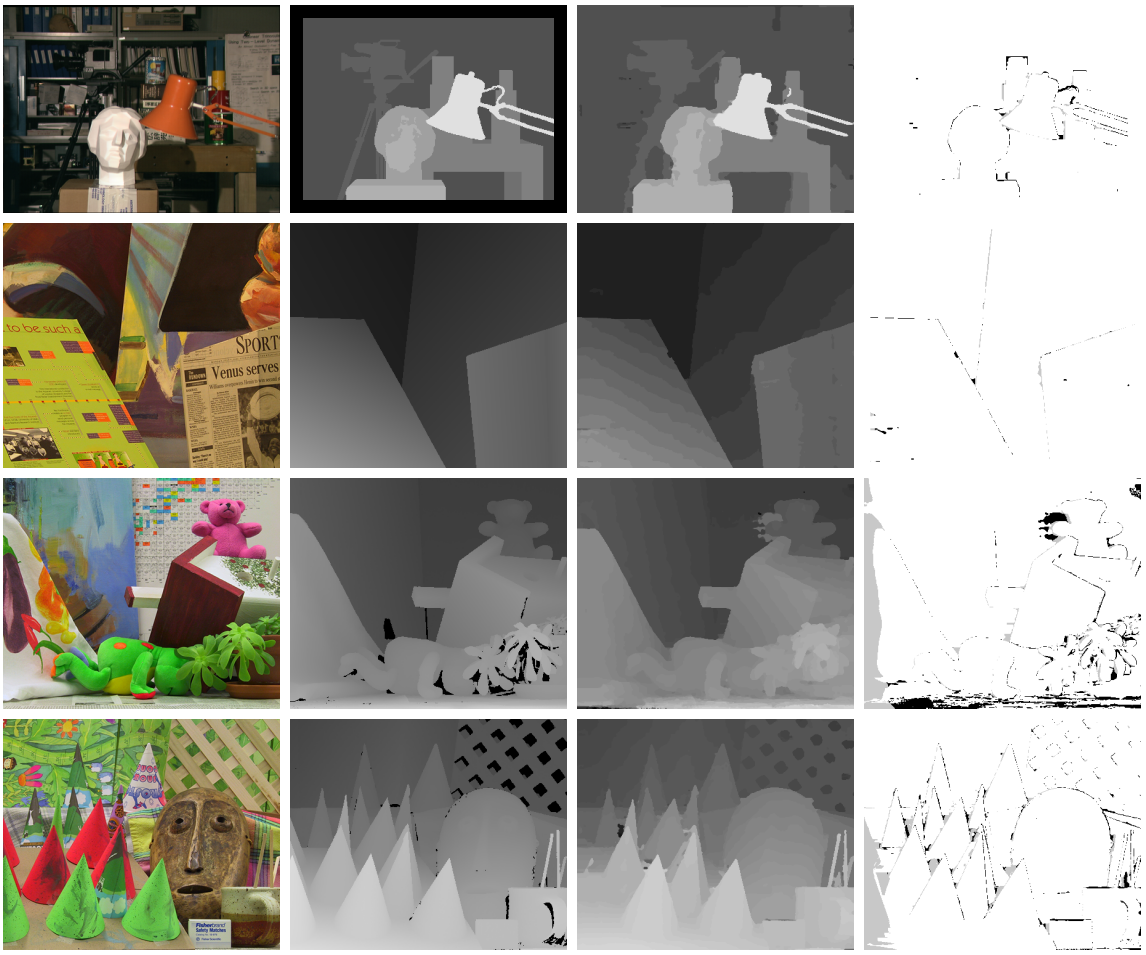


Figure 5.3: Results of the proposed stereo matching method on standard datasets from the Middlebury stereo performance benchmark. Black pixels in the ground truth images indicate regions for which no true disparity value is provided.

Table 5.2: The top 15 most accurate stereo algorithms according to the Middlebury benchmark, as of March 24, 2014.

Method	Avg. rank[1]	Tsukuba			Venus			Teddy			Cones			Avg. % BP[3]
		nonocc	all[2]	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	
TSGO	9.6	0.87 3	1.13 1	4.66 6	0.11 6	0.24 8	1.47 7	5.61 36	8.09 16	13.8 29	1.67 1	6.16 1	4.95 1	4.06
ADCensus [147]	13.1	1.07 18	1.48 15	5.73 21	0.09 2	0.25 11	1.15 2	4.10 15	6.22 7	10.9 13	2.42 19	7.25 14	6.95 20	3.97
AdaptingBP [88]	16.3	1.11 21	1.37 9	5.79 23	0.10 4	0.21 7	1.44 6	4.22 17	7.06 14	11.8 17	2.48 23	7.92 27	7.32 28	4.23
CoopRegion [117]	16.8	0.87 5	1.16 2	4.61 4	0.11 5	0.21 5	1.54 11	5.16 29	8.31 19	13.0 24	2.79 40	7.18 13	8.01 45	4.41
Iterative Refinement	21.5	0.89 8	1.45 14	4.65 5	0.23 42	0.57 57	2.31 36	4.85 21	10.4 33	12.8 22	2.05 2	7.44 16	5.88 2	4.46
RDP [148]	21.8	0.97 11	1.39 11	5.00 11	0.21 36	0.38 26	1.89 21	4.84 20	9.94 31	12.6 20	2.53 26	7.69 20	7.38 29	4.57
MultiRBF [149]	22.2	1.33 44	1.56 19	6.02 30	0.13 10	0.17 2	1.84 18	5.09 27	6.36 8	13.4 28	2.90 47	6.76 8	7.10 25	4.39
DoubleBP [86]	22.7	0.88 7	1.29 6	4.76 9	0.13 11	0.45 43	1.87 20	3.53 12	8.30 18	9.63 8	2.90 46	8.78 55	7.79 37	4.19
MDPM	22.8	1.15 22	1.59 22	6.14 33	0.14 16	0.36 24	1.52 10	3.79 13	5.78 5	11.1 15	2.74 34	8.38 40	7.91 40	4.22
OutlierConf [150]	23.4	0.88 6	1.43 13	4.74 8	0.18 25	0.26 13	2.40 37	5.01 23	9.12 27	12.8 23	2.78 39	8.57 46	6.99 21	4.60
SegAggr	24.6	1.99 85	2.39 76	8.59 84	0.12 7	0.21 6	1.68 13	2.19 2	3.73 1	7.02 2	2.16 8	6.52 3	6.37 8	3.58
AdaptiveGF	27.2	1.04 14	1.53 16	5.62 16	0.17 24	0.41 34	1.98 24	5.71 39	11.3 45	14.3 36	2.44 21	8.22 34	7.05 24	4.98
SOS	27.7	1.45 55	1.63 26	7.83 73	0.21 34	0.32 17	2.29 35	3.13 8	8.45 21	9.74 9	2.43 20	7.10 12	7.02 22	4.30
SubPixSearch [151]	28.6	2.04 89	2.48 80	6.40 40	0.14 15	0.40 33	1.74 15	4.00 14	6.39 9	11.0 14	2.24 12	6.87 10	6.50 12	4.18
SubPixDoubleBP [152]	29.1	1.24 30	1.76 39	5.98 29	0.12 9	0.46 45	1.74 15	3.45 11	8.38 20	10.0 11	2.93 49	8.73 52	7.91 39	4.39

[1] Average rank over all datasets and categories. Individual ranks are given next to error rates

[2] nonocc = non-occluded regions, all = all pixels with known ground truth, disc = depth-discontinuous regions

[3] Average percentage of incorrectly assigned disparities over all datasets

Table 5.3: Accuracy of local stereo algorithms capable of real-time operation, as listed on the Middlebury benchmark.

Method	Tsukuba			Venus			Teddy			Cones			Avg. % BP[2]
	nonocc	all[1]	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	
Iterative Refinement	0.89	1.45	4.65	0.23	0.57	2.31	4.85	10.4	12.8	2.05	7.44	5.88	4.46
DTAggr-P [153]	1.75	2.10	7.09	0.24	0.45	2.59	5.70	11.5	13.9	2.49	7.82	7.30	5.24
HEBF [105]	1.10	1.38	5.74	0.22	0.33	2.41	6.54	11.8	15.2	2.78	9.28	8.10	5.41
CostFilter [143]	1.51	1.85	7.61	0.20	0.39	2.42	6.16	11.8	16.0	2.71	8.24	7.66	5.55
iFBS [154]	1.78	2.10	7.57	0.31	0.50	2.17	7.94	12.8	17.1	3.07	8.73	8.46	6.05
FastBilateral [99, 103]	2.38	2.80	10.4	0.34	0.92	4.55	9.83	15.3	20.3	3.10	9.31	8.59	7.31
RealTimeBFV [102]	1.71	2.22	6.74	0.55	0.87	2.88	9.90	15.0	19.5	6.66	12.3	13.4	7.65
ESAW [100]	1.92	2.45	9.96	1.03	1.65	6.89	8.48	14.2	18.7	6.56	12.7	14.4	8.21
RealTimeGPU [98]	2.05	4.22	10.6	1.92	2.98	20.3	7.23	14.4	17.6	6.41	13.7	16.5	9.82
DCBGrid [136]	5.90	7.26	21.0	1.35	1.91	11.2	10.5	17.2	22.2	5.34	11.9	14.9	10.90

[1] nonocc = non-occluded regions, all = all pixels with known ground truth, disc = depth-discontinuous regions

[2] Average percentage of incorrectly assigned disparities over all datasets

corresponding to the surface covered with newspapers, i.e., the ground plane. Due to limited image resolution and the sampling that occurred during acquisition of the `teddy` dataset, elements in the periodic table form repeating patterns and introduce ambiguity into matching. With the gradient term strongly outweighing the color term within the cost metric and severe truncation of gradient differences, the algorithm is unable to make a correct choice between alternative matches, resulting in erroneous disparity assignment. Increasing the impact of the color term within the cost metric, while simultaneously relaxing the gradient truncation value, mitigates the problem of mismatches caused by this repeating pattern. However, such an adjustment of the parameters leads to a decrease in accuracy when used with the remaining datasets.

The challenges posed by the ground plane in the `teddy` image are two-fold. First, the ground plane contains fine-scale repeating patterns, making it susceptible to improper disparity assignment due to the absence of strong, definite minima in matching costs, akin to the area by the teddy bear. Second, the ground plane is slanted with respect to the image plane. Like many local stereo algorithms, the proposed method operates under the assumption that scene surfaces captured by the adaptive support windows are approximately parallel to the image plane, and in that event pixels within these windows are likely to have a common disparity. This assumption is explicitly implemented by the cost aggregation stage, where the layers corresponding to distinct integer-valued disparity hypotheses are filtered independently. Slanted surfaces violate this assumption and hence the accuracy of disparities within the ground plane segment is limited.

Stereo matching results on the 2005 supplemental Middlebury datasets [155, 156]

are shown in Figure 5.4, and the results on selected 2006 datasets are shown in Figure 5.5. The error rates obtained with and without the disparity filling operation are listed in Table 5.4. Since the 2005 and 2006 Middlebury datasets contain images of higher resolution and with broader disparity ranges than the standard datasets, results using error thresholds of both 1 and 2 are given in Table 5.4. The set of parameters used when processing the supplemental datasets is the same as the one used with the standard datasets. With an error threshold of 1, the proposed method achieves an average error rate of 17.35% with the disparity fill enabled, and 11.36% leaving inconsistent areas unfilled. Setting the error threshold to 2, the error rates decrease to 12.53% with disparity filling and 7.5% without disparity filling. Significant errors are observed in regions affected by the disparity filling operation, i.e., occlusions, areas nearby image boundaries and inconsistently matched patches. Errors is these regions amount to 35% and 40% of the overall error rate when the error threshold is set to 1 and 2, respectively.

Note that abnormally high error rates were recorded for the `Midd1` (up to 52.32%), `Midd2` (up to 45.85%), and `Monopoly` (up to 40.15%) datasets (see Table 5.4 for a more detailed listing of the error rates). The scenes depicted in the `Midd1` (3rd row in Figure 5.5), `Midd2`, and `Monopoly` (4th row in Figure 5.5) datasets, incorporate planar, uniformly colored and weakly textured backgrounds that lack distinctive visual features and thus do not allow for correct disparity assignment using the proposed method. Uniformly colored and weakly textured surfaces are a well-known challenge in stereo matching, where correspondences are selected based solely on the photometric content of the input images. The case of uniformly colored and weakly textured planar

surfaces, however, can be addressed by incorporating image segmentation and plane fitting techniques into the matching process.

The average runtime of the stereo matching method recorded during the evaluation is 16.175ms (6.5ms on `tsukuba`, 13.6ms on `venus`, and 22.3ms on both `teddy` and `cones` datasets). When applied to 640 × 480 images with 60 disparity levels, the proposed method is able to operate at 29.4 frames per second, which is equivalent to evaluating 524.12 million disparity estimates per second (MDE/s). A comparison of the speed and accuracy of real-time local stereo algorithms is given in Table 5.5. Since runtimes of relevant methods are reported in the literature for arbitrary image sizes, corresponding MDE/s metrics have been computed for each method by taking a product of the image dimensions and the number of disparity hypotheses, and dividing the result by the quoted runtime (in seconds). Additionally, the corresponding frame rate estimates are given for the case of processing 640 × 480 images with 60 disparities.

The results demonstrate that iterative refinement achieves the highest frame rate and processes the highest number of disparity estimates per second of all real-time local stereo algorithms listed on the Middlebury benchmark 2.0 to date. Note, however, that this outstanding performance is largely due to the use of NVIDIA's latest graphics card, the GeForce GTX TITAN Black. The TITAN Black offers more computing cores than any of the cards named in Table 5.5, which provides high parallelism of computations and enables high-performance stereo matching using iterative refinement despite the fact that each core is clocked at a lower frequency than most of NVIDIA's previously released graphics cards. If the performance of the methods listed in Table 5.5 were to be assessed using the same graphics hardware, the proposed method is

Figure 5.4: Disparity maps generated for the 2005 Middlebury datasets.

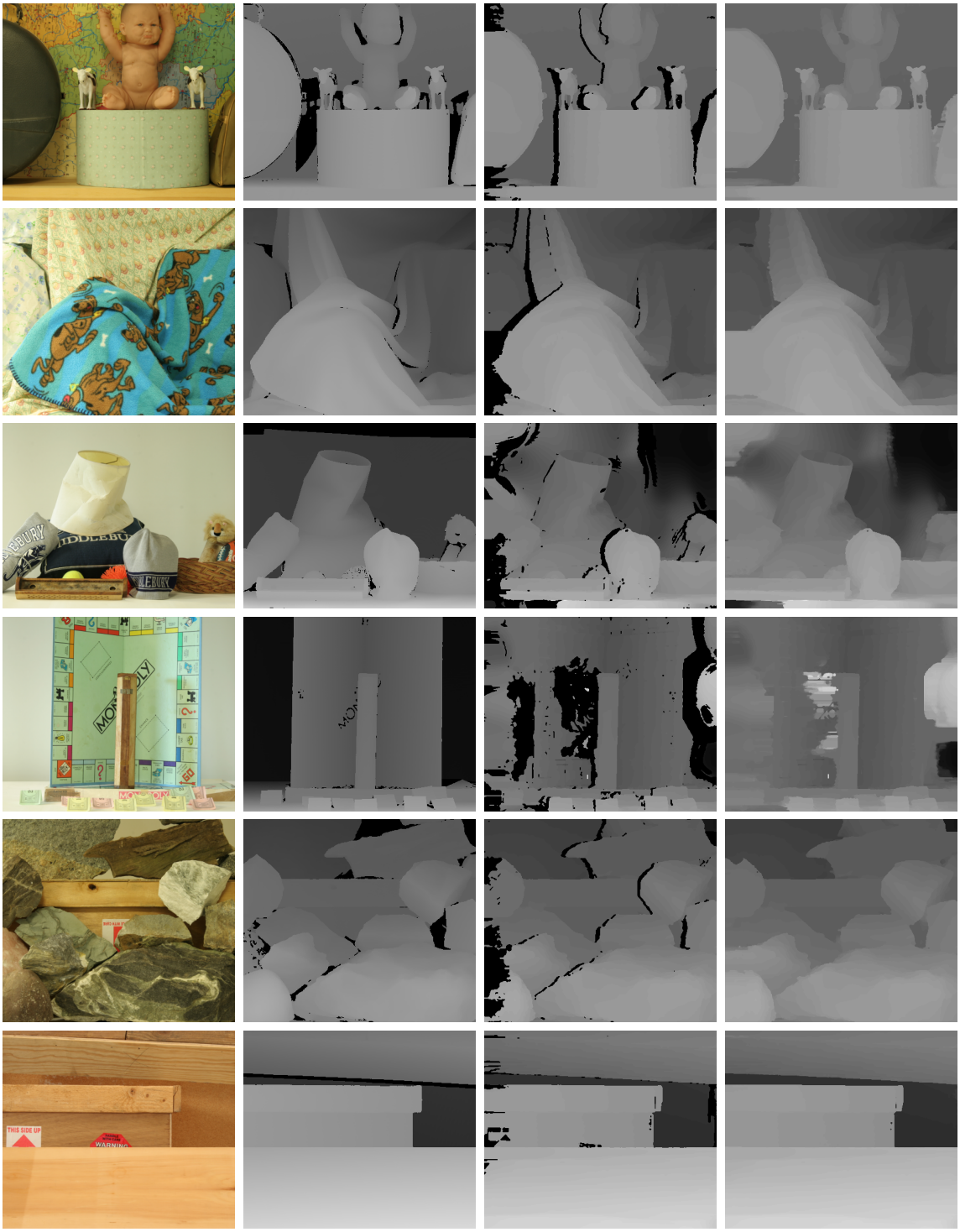


Figure 5.5: Disparity maps generated for selected 2006 Middlebury datasets.

Table 5.4: Stereo error rates achieved on the 2005 and 2006 Middlebury datasets.

Dataset	Fill rate (%)	% bad pixels (threshold = 1)		% bad pixels (threshold = 2)	
		consistent	all	consistent	all
Art	83.19	12.13	20.47	9.25	16.46
Books	91.16	11.92	18.11	7.02	12.66
Dolls	89.79	5.40	12.03	2.52	7.76
Laundry	85.82	11.27	18.38	5.58	10.63
Moebius	90.61	9.27	13.92	6.83	10.45
Reindeer	87.98	8.95	13.05	4.34	7.31
Aloe	89.82	3.57	7.55	2.11	5.43
Baby1	94.48	4.92	7.19	2.96	5.06
Baby2	94.80	5.19	7.27	1.43	3.34
Baby3	94.74	3.05	5.58	1.98	4.10
Bowling1	88.15	23.00	32.32	14.18	22.36
Bowling2	89.16	9.68	16.67	4.89	9.98
Cloth1	93.12	0.61	3.96	0.29	1.74
Cloth2	87.68	2.77	10.28	1.07	6.96
Cloth3	94.81	1.29	4.43	0.54	3.34
Cloth4	92.27	6.26	10.96	3.42	7.34
Flowerpots	94.65	14.59	17.85	7.16	9.29
Lampshade1	91.93	17.87	23.33	8.48	13.24
Lampshade2	84.78	14.98	25.17	8.82	17.96
Midd1	88.91	42.22	51.32	38.83	46.84
Midd2	85.16	33.35	45.84	29.49	40.92
Monopoly	79.03	21.81	40.15	20.61	37.38
Plastic	82.44	19.75	32.34	12.59	24.08
Rocks1	94.91	2.41	4.65	0.87	2.37
Rocks2	95.17	2.12	4.97	0.90	2.98
Wood1	91.92	11.96	14.16	5.43	7.30
Wood2	97.91	6.39	6.66	0.97	1.08

Table 5.5: A comparison of speed and accuracy of real-time local stereo algorithms.

Method	GPU	MDE/s[1]	Frame rate[2]	Avg. % BP
Iterative Refinement	Titan Black	542.1	29.41	4.46
DTAggr-P [153]	GeForce GTX 460	186.3	10.11	5.24
HEBF [105]	GeForce GTX 580	478.1	25.94	5.41
CostFilter [143]	GeForce GTX 480	110.3	5.98	5.55
FastBilateral [99, 103]	Tesla C2070	50.6	2.75	7.31
RealTimeBFV [102]	GeForce 8800 GTX	114.3	6.2	7.65
ESAW [100]	GeForce 8800 GTX	194.8	10.57	8.21
RealTimeGPU [98]	Radeon X1800	52.8	2.86	9.82
DCBGrid [136]	Quadro FX 5800	25.1	1.36	10.90

[1] Number of disparity estimates per second $\times 10^6$

[2] Estimated frame rate on 640 $\times$ 480 images, considering 60 disparities

expected to rank among the most efficient stereo algorithms due to its computational regularity.

Further performance gains in terms of frame rate can be achieved by decreasing the window size in the refinement and disparity filling operations to 33 $\times$ 33 (a filter radius of 16) and using 2 iterations of refinement. In this configuration, with appropriately tuned parameters, the proposed method processes 640 $\times$ 480 images (with 60 disparities) at 33.4 frames per second, while the average error rate on standard Middlebury datasets is kept within 4.6%.

5.3 Evaluation using Middlebury 3.0 Benchmark

In the summer of 2014, a preliminary release of the Middlebury stereo performance benchmark version 3.0 was made available to the researchers. The third version of the benchmark introduced updates into the evaluation methodology and presented imagery that poses a challenge to modern stereo matching methods. In this section, changes in the benchmark from version 2.0 are discussed, and improvements to the proposed stereo matching method are described that address some of the challenges posed by the new datasets. Later in this section, the proposed method is evaluated using the methodology and the datasets of the Middlebury 3.0 benchmark.

5.3.1 Changes from Version 2.0

The Middlebury 3.0 stereo performance benchmark provides 30 datasets of static indoor scenes, for which sub-pixel accurate, i.e., floating-point, ground truth disparities were obtained using structured lighting technique described in [157]. The precision of the ground truth data is estimated to be within 0.2 pixel. At full resolution (5-6 megapixels for most of the datasets), the maximum disparity values range between 240 and 800 pixels. Compared to the previous version of the benchmark, where left and right views in all of the datasets were acquired under consistent illumination and camera settings resulting in highly photoconsistent images, radiometric distortions were introduced into some of the new datasets by varying the lighting conditions within the scene or by changing camera exposure times. To further increase the

difficulty of matching, most of the new stereo image pairs are imperfectly rectified, which manifests itself in vertical offsets between the corresponding points in each pair of views. Perfectly rectified versions of the views most severely affected by imperfect rectification are also included in the new datasets.

The datasets in version 3.0 are divided into a training set and a test set, each containing 15 image pairs. The ground truth disparity maps for the training set are distributed in a floating-point PFM image format as a part of the Middlebury stereo evaluation software kit that can be used to asses the accuracy of a given stereo method on the training set, for instance during parameter adjustment. The ground truth disparity maps for the test set, on the other hand, are not disclosed and are used in the online evaluation system at http://vision.middlebury.edu/stereo/eval3/. The images, together with the corresponding occlusion maps and ground truth disparities, are available in 3 resolutions (full size, half size, and quarter size), to allow for evaluation of stereo algorithms that are not able to operate at full resolution, due to their high computational complexity or high memory requirements. Note that discontinuity maps are no longer provided, since the problem of aligning depth discontinuities with object edges has been largely addressed in the state-of-the-art stereo algorithms.

The online evaluation system allows submission of results generated using both the training and test sets. In contrast to version 2.0 which only accepts dense results, the new benchmark gives researchers an option to additionally submit sparse/semi-dense results that can be obtained, for instance, by leaving occlusions and inconsistencies unfilled. Results are submitted in the PFM format to enable comparison with the ground truth data. While the results can be submitted in half or quarter size, the

evaluation is always performed at full resolution, upsampling the disparity maps if necessary. By default, only non-occluded pixels ("nonocc") are included in the evaluation; it is still possible, however, to perform evaluation using all pixels with known ground truth disparities ("all"). The new benchmark defines a broader range of stereo performance metrics. These include:

- percentages of bad pixels, i.e., pixels whose disparity assignment differs from the ground truth disparity by a given threshold. Thresholds of 0.5, 1, 2, and 4 pixels are used. The corresponding metrics will hereafter be labeled as `bad0.5`, `bad1.0`, `bad2.0`, and `bad4.0`, respectively.

- average absolute error (`avgerr`) and average root-mean-square error (`rms`) between the computed disparities and the true disparities,

- the 50th, 90th, 95th, and 99th error percentiles, i.e., values of the absolute disparity error such that 50, 90, 95, and 99 percent of errors fall below these values. These metrics will hereafter be referred to using the mnemonics `A50`, `A90`, `A95`, and `A99`.

- matching time (`time`) in seconds recorded for individual datasets.

In the following sections, the `bad*`, `avgerr`, `rms` and `A*` stereo performance metrics will sometimes be referred to as error metrics, and their values as error rates.

The new benchmark abandons the idea of generating a single global ranking of stereo methods by averaging the ranks achieved on each of the datasets. In version 3.0, multiple rankings are generated based on weighted averages of individual metrics over

all image pairs in the training/test set, with the `bad2.0` chosen as the default metric. Currently, the majority of the datasets are assigned a weight value of 1.0, except for the 5 most challenging image pairs in both training and test sets, whose impact has been reduced by setting the weights to 0.5. Due to the floating-point representation of disparities and the fact that the maximum disparity value typically exceeds the number of intensity levels in grayscale images, the Middlebury 3.0 benchmark advocates color-coding of disparity maps for the purpose of visualization. A commonly used `Jet` color map is recommended that maps lower disparity values (background objects) to colder colors, e.g., various shades of blue, and higher disparity values (foreground objects) to hotter colors, such as yellow, orange, and red.

5.3.2 Improvements to Iterative Refinement

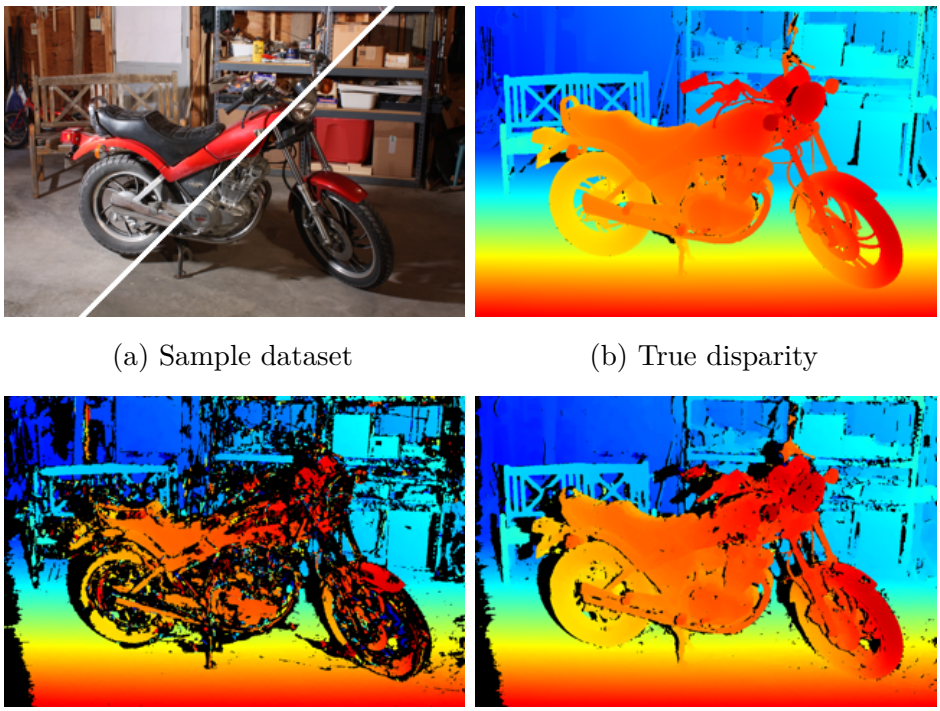
To address the challenges posed by the latest Middlebury datasets, a number of improvements and modifications have been introduced into the proposed method. The changes include a new cost metric that improves matching under radiometric distortions, a modified refinement scheme that facilitates recovery of sub-pixel accurate disparities, and a modified disparity filling operation capable of handling large unfilled regions that frequently occurr when matching high-resolution images. In the following, these changes are discussed in detail.

As described in section 4.2, the per-pixel matching costs evaluated by the proposed method are weighted sums of dissimilarity terms based on both the intensity values and local estimates of image gradients, respectively. While the gradient term provides

a certain degree of invariance to radiometric distortions, since pixel dissimilarity is computed with respect to the relative changes of intensities rather than using intensities directly, it is clear that the presence of the intensity-based cost term will cause invalid matches if the photometric consistency of the images is imperfect. In order to improve matching under radiometric distortions the intensity component of the cost metric is replaced with an alternative computed using census-transformed input images.

The census transform [158], which operates on grayscale images, performs a series of intensity comparisons between a pixel of interest and an ordered set of nearby pixels to form a binary-valued census vector whose elements contain the results of individual comparisons. Given a pair of census-transformed images, the dissimilarity of any two pixels is assessed by computing the Hamming distance between the corresponding census vectors, i.e., the number of bits at which the two vectors differ. Similarly to the sums of absolute color differences in the initial formulation, the Hamming distances are not allowed to exceed the truncation value τ_c. Note that global changes in image intensities will not affect the census vectors, nor the resulting dissimilarity metrics, as long as the pairwise pixel relations are preserved. This property makes the dissimilarity term based on distance between census vectors a suitable component of the matching cost if exposure and illumination changes are a possibility.

Rectangular 9×7 windows are considered when performing census transforms. This way, each of the resulting census vectors can be compacted into a single 64-bit integer, which minimizes the number of global memory reads necessary to compute the cost volume. Using the TITAN Black graphics card, the computation of census vectors



(a) Sample dataset (b) True disparity

(c) Intensity + gradient (d) Census + gradient

Figure 5.6: Stereo matching of images with radiometric distortions using the cost metrics with and without the census term. Consistent disparities are shown; disparity filling operation has been disabled.

takes about 0.37ms per every megapixel of image data, and on average amounts to 0.3% of the time necessary to match a pair of images.

The effects of including a census-based dissimilarity term in the matching cost are illustrated in Figure 5.6. The disparity maps shown in the figure were generated using the `MotorcycleE` stereo image pair included in the Middlebury 3.0 training set that was acquired using different exposure settings of the cameras. Compared to the cost formulation incorporating intensity and gradient differences, the formulation where the intensity difference is replaced with the census distance produces disparity maps with larger number of consistent disparities. The later also reduces the number of

mismatches, which is due to the fact that the census distance effectively compares windows rather than pairs of pixels, acting much like a cost aggregation scheme.

Matching with sub-pixel precision is made possible using a modified disparity refinement operation that computes expected disparity values based on local estimates of disparity gradients, followed by fitting of quadratic terms to cost values in order to obtain a floating-point disparity assignment. Instead of averaging initial disparity estimates within the support region, the refinement operation now generates predictions of the disparity value at the pixel of interest using the disparity gradient information at nearby pixels. Precisely, the disparity estimate $\tilde{D}^i(\boldsymbol{p})$ at pixel $\boldsymbol{p}$ is calculated as an adaptively weighted average of individual predictions, i.e.,

$$\tilde{D}^i(\boldsymbol{p}) = \frac{\sum\limits_{\boldsymbol{q}\in\Omega_{\boldsymbol{p}}\setminus\boldsymbol{p}} w(\boldsymbol{p},\boldsymbol{q})F^{i-1}(\boldsymbol{q})\left[D^{i-1}(\boldsymbol{q}) + \nabla D^{i-1}(\boldsymbol{q})(x_{\boldsymbol{p}} - x_{\boldsymbol{q}}, y_{\boldsymbol{p}} - y_{\boldsymbol{q}})^{\mathsf{T}}\right]}{\sum\limits_{\boldsymbol{q}\in\Omega_{\boldsymbol{p}}\setminus\boldsymbol{p}} w(\boldsymbol{p},\boldsymbol{q})F^{i-1}(\boldsymbol{q})}, \qquad (5.4)$$

where $\nabla D^{i-1}(\boldsymbol{q}) = \left(\frac{\partial D^{i-1}}{\partial x}(\boldsymbol{q}), \frac{\partial D^{i-1}}{\partial y}(\boldsymbol{q})\right)$ is the disparity gradient at pixel $\boldsymbol{q} \in \Omega_{\boldsymbol{p}} \setminus \boldsymbol{p}$. Equation (5.4) can be efficiently evaluated using the two-pass approach, similarly to the cost aggregation and the originally proposed disparity refinement.

Once the disparity estimates and the corresponding cost penalties are known, disparities are re-assigned and then adjusted by fitting a quadratic function to the penalized costs. The fitting procedure samples the cost values at three discrete disparities, i.e., the minimum-cost disparity and the two adjacent disparities, and then calculates the coefficients of a quadratic function that fits the samples. The axis of symmetry or, more specifically, the horizontal coordinate of the vertex of the quadratic function, determines the sub-pixel accurate disparity value. Note that, since the disparity update operation results in integer-valued disparities, fitting has to be

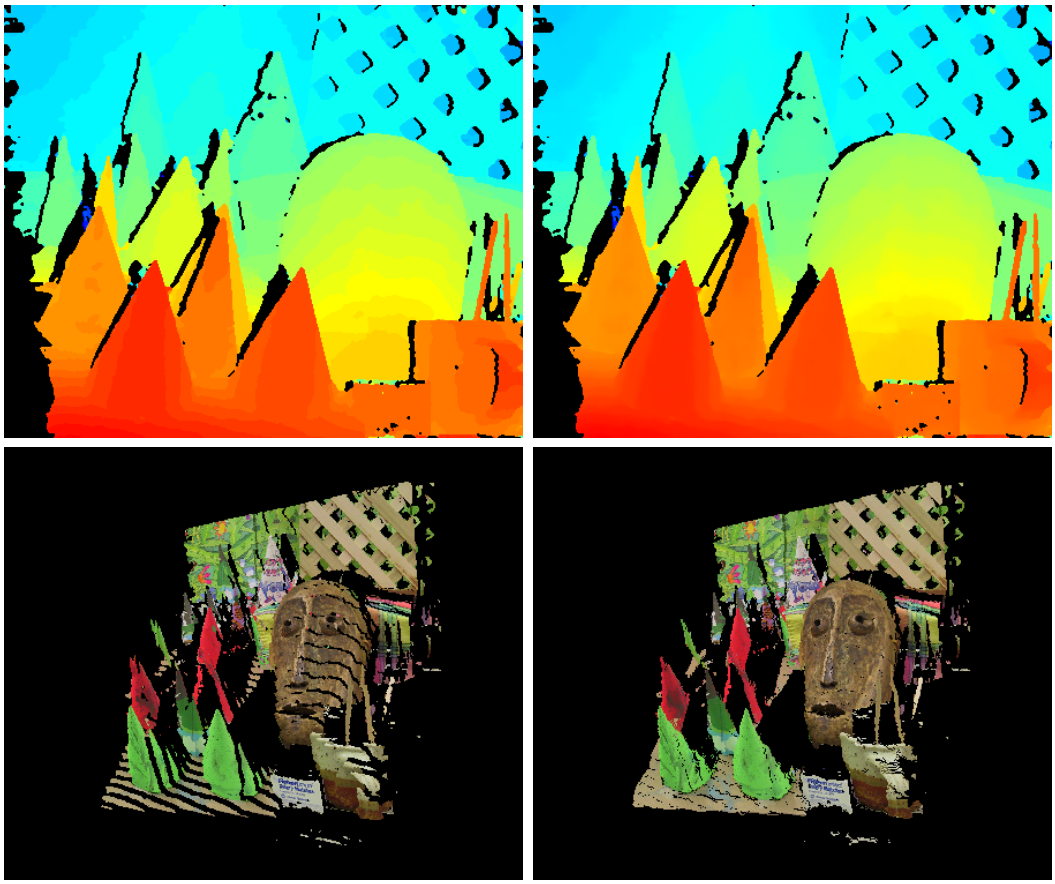


Figure 5.7: Discrete (left) versus sub-pixel disparities/depths (right).

performed after each iteration of refinement. The fitting operations is also required after the initial assignment of disparities.

The modifications that provide IDR with the capability to perform stereo matching with sub-pixel accuracy result in minuscule time overhead. Computation of disparity gradients requires no additional data to be fetched by the refinement kernels and hence takes a negligible amount of time. Despite the fact that the kernel responsible for quadratic fitting cannot take advantage of coalesced global memory reads, since the memory access pattern depends on the disparity assignment, the fitting operation constitutes only 0.3% of the total matching time.

A comparison of disparity maps and depth models generated with and without support for sub-pixel accuracy is given in Figure 5.7. It can be seen that the gradient-based disparity refinement in combination with fitting of quadratic functions to cost values greatly reduces the staircase effect in the disparity maps, resulting in smoother depth models. Unlike the originally proposed refinement operation that favors fronto-parallel surfaces where disparities are piece-wise constant, the version enhanced with disparity gradient information encourages gradual transitions of disparities and avoids flattening of the objects in the scene, which is key to accurate matching of slanted surfaces. In particular, the cost penalties produced by the gradient-based refinement enable the fitting operation to recover floating-point disparities that are consistent with the orientation of the surfaces.

Through the course of experiments, it has also been observed that the originally proposed disparity filling operation fails to generate valid fill values when the support window does not extend over pixels with non-zero confidence or, if such pixels are present in the support window, when the support weights are all zero. The former frequently occurs in high-resolution images where the unfilled areas, the majority of which are due to occlusions, are inherently large. The later pertains to the case of isolated unfilled patches, where strong color dissimilarity zeroes the supports from nearby pixels. In both cases the denominator of the bilateral filter equals or approaches zero, which makes computation of fill values poorly conditioned.

In order to avoid this, a filling operation is used that propagates the most reliable disparities along the rows of the disparity map. First, disparities with the highest non-zero confidence are searched for by scanning to the left of each unfilled area within

a radius of 32 pixels. If found, their values become the disparity assignments of unfilled pixels; the confidence levels of these pixels are also overwritten with non-zero values. To enable filling of large areas, the filling operation is iterated 3 times. Next, the process is repeated, this time scanning to the right of unfilled areas. While filling from the left has sound judgement when handling object occlusions, filling from the right is necessary to generates fill values for unmatched regions next to the left boundary of the image that cannot be filled otherwise. In addition, median filtering is applied prior to every step of filling to eliminate salt-and-pepper mismatches and minor streaking artifacts from the disparity map.

5.3.3 Evaluation Results

Following the recommendation of the benchmark's creators, the disparity maps for evaluation were generated at the highest resolution that the proposed method was able to handle. In the case of IDR, the maximum resolution of the images being matched is limited by the amount of global memory available to the GPU of the graphics hardware. The 6GB of global memory that the TITAN Black graphics card is equipped with allowed for processing of half-resolution datasets. On average, the method consumed 0.27GB of global memory per dataset when operating on quarter-size datasets; at half-resolution, the average memory consumption per dataset increased to 2.03GB, with some of the datasets requesting as much as 4.2GB. Up to 32GB of global memory is required to process full-resolution images.

Modifications introduced into the proposed method made it necessary to adjust

the values of the method's parameters. This was accomplished by optimizing the value of the default error metric (`bad2.0`) using the technique discussed in section 5.1. In order to obtain a general-purpose set of parameters, i.e., a set of parameters that provides accurate matching for a variety of scenes, the weights of individual datasets were set to equal values. The optimization yielded the census/gradient difference truncation values $\tau_c = 38.03$ and $\tau_g = 2.51$, and the cost term balance coefficient $\alpha = 0.09$ (91% of the cost value comes from the gradient term, 9% comes from the census term). The resulting filter parameters are $\sigma_R = 23.39$ and $\sigma_S = 7.69$ in the cost aggregation stage, and $\sigma'_R = 11.3$ and $\sigma'_S = 17.2$ in the disparity refinement stage. The refinement is performed using the cost penalty coefficient $\lambda = 0.0023$.

Previews of the training datasets, their associated numbers of disparities, nominal sizes in megapixels, and the corresponding ground truth disparity maps, along with the disparity maps (both dense and sparse) generated using the proposed method are given in Figures 5.8 and 5.9. Dataset name suffix '`E`' indicates changed exposure, '`L`' indicates changed lighting, and '`P`' stands for perfect rectification. The sparse disparity maps were obtained by thresholding the confidence maps (requiring a minimum confidence of 0.1, rejecting pixels otherwise), and with the disparity filling disabled.

Average error rates computed in the non-occluded regions of the dense disparity maps generated using the training datasets are quantified in Table 5.6. Detailed dense results in non-occluded areas of individual datasets using the `bad*`, `avgerr`, and `rms` error metrics are given in Tables 5.7 through 5.12. Average values of the error metrics evaluated across all pixels of the dense disparity maps for which the ground truth data are available follow in Table 5.13. Finally, in Table 5.14 average error rates are

given of the sparse disparity maps in non-occluded regions. In each table, the results of the proposed method are contrasted with the results achieved by other methods that were listed on the Middlebury 3.0 benchmark's website at the time of evaluation (October 7, 2014). Ranks are given in small font next to error rates; the lowest error rates are highlighted in bold font. Note that the tables containing the average error rates are sorted alphabetically by the method name, whereas the tables containing the results on individual datasets are sorted by the average error rate in ascending order.

Results on the training set demonstrate high accuracy of the proposed method. In the case of dense results in non-occluded regions, IDR is the top-performing method when percentages of incorrectly assigned pixels, i.e., the `bad*` metrics, are taken into account. Choosing the error thresholds of 0.5, 1.0, 2.0, and 4.0 pixels, the corresponding average percentages of pixels in error are 48.2, 27.6, 18.1, and 11.8%, respectively. Examination of the `bad*` metrics computed for individual datasets reveals that IDR ranks first in exactly 1/3 of the cases, simulteneously ranking among the top five, and in most cases among the top three methods, on nearly all of the remaining datasets. The proposed method achieves an average `avgerr` (absolute disparity error) of 5.33, and an average `rms` (root-mean-square error) of 17.1 pixels, ranking third in both cases. The two methods with the lowest values of the `avgerr` and `rms` metrics are the half-size and the quarter-size variants of the semi-global matching (SGM) [118], which outperform IDR by 0.04 and 0.5 pixels in the case of `avgerr`, and 0.01 and 1.2 pixels in the case of `rms`. While the percentages of incorrectly assigned disparities are favorable to IDR, it must be that the disparity errors at incorrectly assigned pixels have, on average, higher magnitudes when compared to SGM. This hypothesis is supported

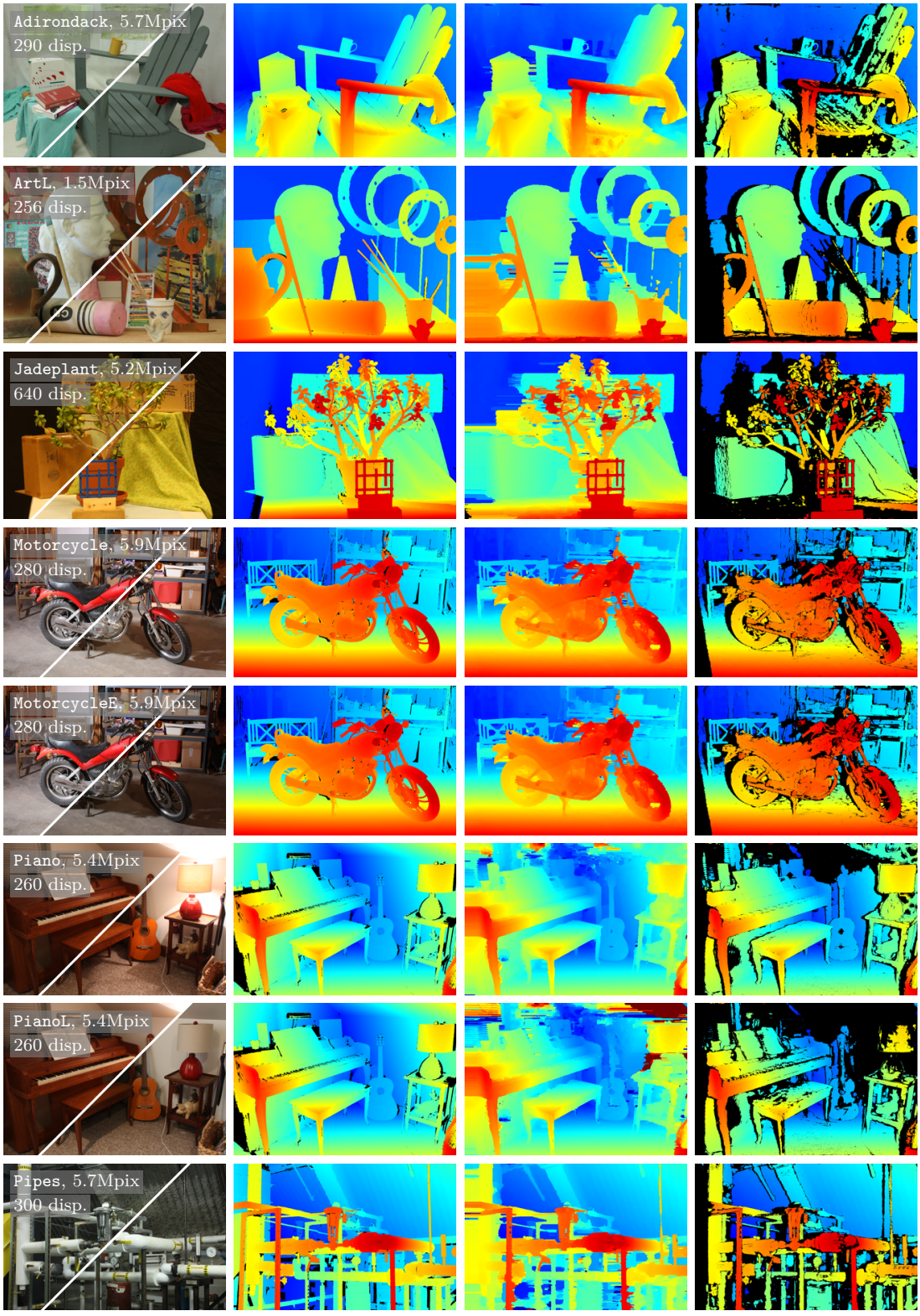


Figure 5.8: Disparity maps computed for the Middlebury 3.0 training set (1 of 2)

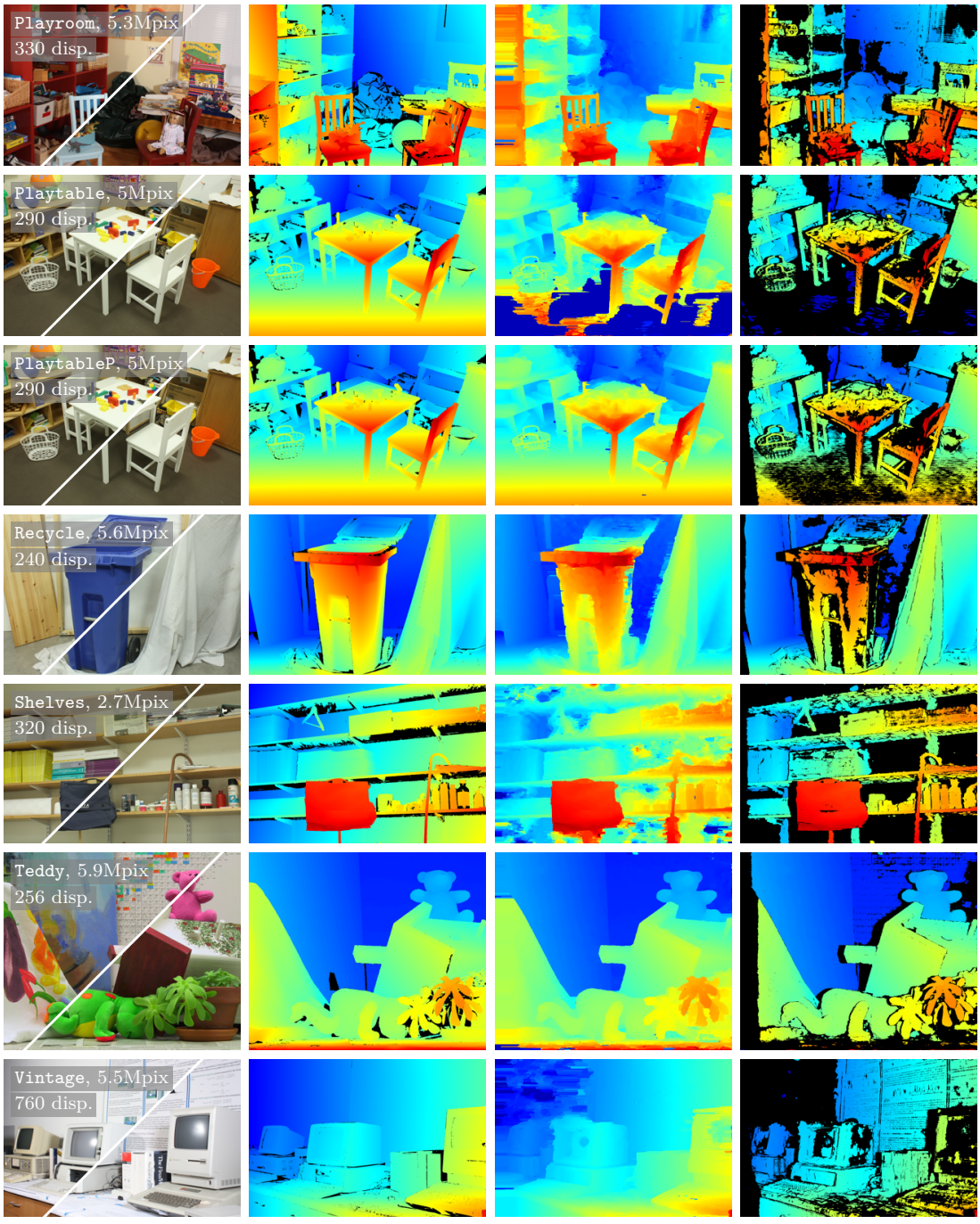


Figure 5.9: Disparity maps computed for the Middlebury 3.0 training set (2 of 2)

Table 5.6: Dense results on the Middlebury 3.0 training set in non-occluded regions.

Method[1]	bad0.5	bad1.0	bad2.0	bad4.0	avgerr	rms	A50	A90	A95	A99
BSM [159], Q	82.2 16	61.9 16	37.1 16	23.4 14	13.4 14	35.8 14	1.61 16	38.8 15	79.5 14	171 16
Cens5 [160], H	55.4 7	34.2 8	23.3 9	17.2 7	7.25 7	21.5 6	0.85 7	17.5 7	44.7 7	108 7
ELAS [161], F	64.7 12	43.9 13	32.6 15	25.4 16	12.0 12	27.1 11	1.26 13	37.2 13	60.8 11	109 8
ELAS [161], H	72.5 15	47.9 15	31.9 14	23.6 15	12.2 13	27.1 12	1.30 15	40.2 16	61.2 12	103 6
IDR, H	**49.6** 1	**26.8** 1	**16.8** 1	**12.0** 1	5.33 3	17.1 3	**0.59** 1	12.8 3	23.8 2	74.1 2
LAMC [162], H	51.8 4	29.6 3	19.8 3	15.0 4	6.31 5	22.0 7	0.67 3	13.0 5	35.8 5	110 9
LPS [163], F	52.3 5	33.7 6	26.2 11	21.0 12	57.0 15	119 16	1.15 12	28.9 12	115 16	162 15
LPS [163], H	57.9 10	33.9 7	23.2 7	18.1 11	85.3 16	97.7 15	1.29 14	25.1 11	108 15	156 14
SGBM1 [164], F	56.2 8	36.5 10	27.4 13	22.0 13	11.3 11	29.7 13	0.89 9	37.9 14	63.7 13	125 12
SGBM1 [164], H	57.8 9	34.3 9	23.3 8	17.7 10	8.45 10	25.5 10	0.79 6	22.2 10	51.9 10	118 10
SGBM1 [164], Q	69.3 14	44.1 14	26.4 12	17.6 9	8.30 8	25.2 8	1.00 11	20.8 8	47.4 8	127 13
SGBM2 [164], Q	67.3 13	42.1 12	25.5 10	17.3 8	8.43 9	25.5 9	0.96 10	21.9 9	49.1 9	123 11
SGM [118], F	52.3 6	31.7 5	22.1 6	16.5 6	5.82 4	17.8 4	0.77 5	12.8 4	32.5 3	84.3 4
SGM [118], H	51.8 3	28.2 2	17.6 2	12.1 2	**4.83** 1	**15.9** 1	0.7 4	10.8 2	**22.6** 1	**71.5** 1
SGM [118], Q	64.6 11	37.3 11	21.0 5	12.9 3	5.29 2	17.1 2	0.86 8	**10.6** 1	32.9 4	80.6 3
SNCC [165], H	50.8 2	29.8 4	20.4 4	15.2 5	6.96 6	20.6 5	0.64 2	16.7 6	40.0 6	91.7 5

[1] Resolutions at which the methods were evaluated follow their acronyms: Q - quarter-size, H - half-size, F - full-size.

Table 5.7: Dense results on the Middlebury 3.0 training set using the `bad0.5` metric in non-occluded regions.

Method	Avg. `bad0.5`	1 `Adiron`	1 `ArtL`	1 `Jadepl`	1 `Motor`	1 `MotorE`	1 `Piano`	0.5 `PianoL`	1 `Pipes`	0.5 `Playrm`	0.5 `Playt`	1 `PlaytP`	1 `Recyc`	0.5 `Shelvs`	1 `Teddy`	0.5 `Vintge`
IDR, `H`	**49.6** 1	**46.2** 1	51.9 7	**43.6** 1	49.0 3	**45.4** 1	47.6 2	59.5 3	37.2 2	60.4 2	76.2 9	**35.6** 1	**44.8** 1	75.1 8	49.5 2	67.3 4
SNCC [165], `H`	50.8 2	48.4 3	49.1 2	47.6 4	50.6 5	46.8 2	**46.9** 1	**58.7** 1	39.1 4	63.0 5	78.8 12	37.3 3	48.6 4	74.5 6	50.1 3	66.3 3
SGM [118], `H`	51.8 3	50.3 4	51.2 5	45.1 3	50.7 6	48.1 4	49.2 4	62.5 4	41.0 8	61.5 3	78.2 10	37.6 4	47.9 3	75.2 9	53.2 8	68.2 6
LAMC [162], `H`	51.8 4	51.3 5	51.8 6	48.5 5	52.2 8	48.8 5	48.4 3	59.0 2	40.7 7	63.7 6	66.2 3	41.6 8	47.8 2	74.6 7	50.6 5	67.6 5
LPS [163], `F`	52.3 5	47.4 2	58.6 9	48.8 6	**34.9** 1	86.9 15	53.8 6	72.0 7	**30.7** 1	**60.0** 1	**46.2** 1	41.2 7	51.0 5	**67.4** 1	**48.5** 1	**57.8** 1
SGM [118], `F`	52.3 6	57.8 8	**42.8** 1	44.1 2	51.1 7	46.9 3	50.4 5	65.6 5	40.3 5	64.5 8	80.5 14	37.0 2	55.2 8	73.2 4	50.4 4	71.3 9
Cens5 [160], `H`	55.4 7	55.4 7	54.3 8	51.5 8	55.7 10	54.4 6	54.5 7	66.2 6	40.5 6	66.7 10	80.0 13	41.0 6	52.6 6	76.3 10	51.5 6	73.9 11
SGBM1 [164], `F`	56.2 8	61.7 11	49.5 4	49.6 7	49.1 4	69.2 9	55.8 8	77.0 11	38.6 3	65.5 9	70.0 5	40.1 5	55.0 7	71.4 3	56.4 9	70.3 7
SGBM1 [164], `H`	57.8 9	60.6 9	60.7 10	52.9 9	52.3 9	61.6 8	57.6 9	74.0 9	41.6 9	63.9 7	68.5 4	44.7 9	56.3 9	73.2 5	59.0 10	70.4 8
LPS [163], `H`	57.9 10	53.2 6	66.1 12	56.0 10	38.7 2	87.4 16	60.3 10	76.2 10	43.6 10	62.5 4	47.0 2	46.7 10	61.3 11	71.2 2	52.4 7	58.7 2
SGM [118], `Q`	64.6 11	61.5 10	65.7 11	63.9 12	62.8 12	60.8 7	64.2 12	73.2 8	60.3 13	70.3 12	78.2 11	55.5 12	59.8 10	80.4 14	64.3 12	74.8 12
ELAS [161], `F`	64.7 12	64.1 12	49.3 3	63.1 11	59.2 11	77.9 12	70.0 13	84.7 14	58.8 11	75.3 14	75.8 8	48.6 11	64.8 13	76.7 11	61.0 11	71.4 10
SGBM2 [164], `Q`	67.3 13	64.5 13	73.3 15	68.3 13	64.0 13	70.8 10	64.1 11	81.8 13	59.3 12	69.8 11	71.1 7	58.8 13	62.4 12	79.4 13	67.0 14	75.7 13
SGBM1 [164], `Q`	69.3 14	71.0 15	72.4 14	69.2 14	66.6 14	71.3 11	71.1 14	81.6 12	61.8 14	71.2 13	70.5 6	60.5 14	67.3 14	78.3 12	65.9 13	76.7 15
ELAS [161], `H`	72.5 15	70.0 14	66.8 13	76.4 15	67.3 15	79.6 14	75.1 15	85.9 15	67.2 15	78.4 15	82.8 15	63.8 15	70.9 15	80.5 15	67.1 15	76.0 14
BSM [159], `Q`	82.2 16	80.6 16	79.8 16	82.9 16	79.4 16	79.1 13	82.3 16	88.6 16	80.4 16	84.5 16	87.5 16	81.9 16	81.5 16	88.0 16	81.0 16	87.6 16

[1] Resolutions at which the methods were evaluated follow their acronyms: `Q` - quarter-size, `H` - half-size, `F` - full-size.

Table 5.8: Dense results on the Middlebury 3.0 training set using the `bad1.0` metric in non-occluded regions.

Method	Avg. `bad1.0`	1 `Adiron`	1 `ArtL`	1 `Jadepl`	1 `Motor`	1 `MotorE`	1 `Piano`	0.5 `PianoL`	1 `Pipes`	0.5 `Playrm`	0.5 `Playt`	1 `PlaytP`	1 `Recyc`	0.5 `Shelvs`	1 `Teddy`	0.5 `Vintge`
IDR, `H`	**26.8** 1	**24.9** 1	14.1 3	**27.5** 1	24.2 3	**21.1** 1	**24.6** 1	**37.4** 1	19.4 3	**37.5** 1	62.0 10	**22.1** 1	**25.2** 1	59.8 8	**8.67** 1	48.9 3
SGM [118], `H`	28.2 2	29.1 4	11.5 2	28.1 2	25.5 4	22.5 2	26.1 2	42.1 4	20.5 5	38.3 2	64.7 11	24.7 2	27.0 2	59.0 5	10.1 2	51.3 6
LAMC [162], `H`	29.6 3	31.0 6	16.9 6	34.0 6	27.9 8	24.1 4	27.8 4	38.2 2	23.1 7	43.7 6	45.6 3	27.5 6	28.7 4	59.5 7	10.8 3	50.6 4
SNCC [165], `H`	29.8 4	28.4 3	15.6 4	33.5 4	26.9 7	23.3 3	27.0 3	40.1 3	23.2 8	43.6 5	65.4 12	25.3 3	28.4 3	60.3 9	11.4 6	50.8 5
SGM [118], `F`	31.7 5	40.9 11	**8.60** 1	29.1 3	28.0 9	24.2 5	29.3 7	46.9 5	23.8 10	45.0 7	68.9 15	26.4 4	35.7 11	59.3 6	10.9 4	59.4 13
LPS [163], `F`	33.7 6	29.9 5	32.7 12	34.7 7	**12.3** 1	78.4 16	28.3 5	59.6 9	**15.8** 1	41.4 3	33.4 2	29.4 11	33.6 8	**51.5** 1	10.9 5	45.8 2
LPS [163], `H`	33.9 7	27.2 2	37.4 15	33.8 5	14.5 2	78.0 15	28.9 6	60.1 10	18.4 2	41.7 4	**31.4** 1	27.2 5	33.1 6	52.5 2	12.7 8	**38.8** 1
Cens5 [160], `H`	34.2 8	37.0 8	17.5 7	35.9 8	32.5 10	28.9 6	33.6 8	47.4 6	23.6 9	47.4 9	67.9 13	29.1 9	33.6 7	62.1 11	12.3 7	61.2 15
SGBM1 [164], `H`	34.3 9	39.5 10	19.0 8	36.0 9	25.9 5	36.9 8	36.6 10	58.6 8	20.4 4	45.0 7	52.4 6	29.2 10	33.0 5	57.2 3	18.1 11	55.0 7
SGBM1 [164], `F`	36.5 10	45.9 14	16.6 5	36.3 10	26.2 6	51.0 11	35.6 9	64.2 11	21.5 6	49.1 12	55.6 7	28.3 8	35.0 10	58.3 4	17.8 10	57.9 12
SGM [118], `Q`	37.3 11	34.4 7	23.5 10	41.0 11	35.6 11	33.8 7	39.5 11	53.6 7	32.6 11	47.8 10	59.3 8	33.9 12	34.2 9	64.3 14	17.3 9	56.8 9
SGBM2 [164], `Q`	42.1 12	39.0 9	35.1 14	48.3 12	37.4 13	50.1 10	41.8 12	68.1 13	32.8 12	48.4 11	48.5 5	34.9 13	37.0 12	62.5 12	27.2 15	57.6 11
ELAS [161], `F`	43.9 13	42.7 12	20.1 9	49.5 14	36.9 12	63.2 13	48.1 13	74.4 15	37.8 14	56.7 14	60.3 9	28.0 7	45.7 14	64.2 13	20.4 12	56.7 8
SGBM1 [164], `Q`	44.1 14	47.5 15	33.2 13	49.0 13	40.6 14	47.0 9	49.9 14	66.9 12	34.6 13	50.3 13	48.2 4	38.2 15	43.0 13	62.0 10	25.3 14	59.7 14
ELAS [161], `H`	47.9 15	45.1 13	25.2 11	57.3 15	41.3 15	63.5 14	55.5 15	74.0 14	42.1 15	60.7 15	68.5 14	35.6 14	46.8 15	65.3 15	23.8 13	57.4 10
BSM [159], `Q`	61.9 16	62.2 16	43.6 16	66.3 16	60.4 16	59.8 12	65.4 16	77.9 16	61.1 16	69.9 16	75.7 16	65.0 16	63.4 16	76.5 16	39.3 16	75.7 16

[1] Resolutions at which the methods were evaluated follow their acronyms: `Q` - quarter-size, `H` - half-size, `F` - full-size.

Table 5.9: Dense results on the Middlebury 3.0 training set using the `bad2.0` metric in non-occluded regions.

Method	Avg. `bad2.0`	1 `Adiron`	1 `ArtL`	1 `Jadepl`	1 `Motor`	1 `MotorE`	1 `Piano`	0.5 `PianoL`	1 `Pipes`	0.5 `Playrm`	0.5 `Playt`	1 `PlaytP`	1 `Recyc`	0.5 `Shelvs`	1 `Teddy`	0.5 `Vintge`
IDR, `H`	**16.8** 1	12.0 2	10.2 3	18.7 2	10.4 3	**8.53** 1	16.6 2	**26.0** 1	11.8 5	22.4 2	49.7 10	**13.4** 1	**12.9** 1	49.0 8	**5.54** 1	34.0 2
SGM [118], `H`	17.6 2	15.3 4	7.69 2	**18.1** 1	10.9 4	8.90 2	**16.4** 1	29.1 3	11.5 4	**21.7** 1	52.5 12	15.8 2	14.6 2	46.4 4	6.52 2	39.3 5
LAMC [162], `H`	19.8 3	17.6 6	13.3 5	25.5 7	12.4 7	10.4 4	19.0 4	27.7 2	15.4 8	31.1 9	31.3 5	19.7 8	17.5 6	49.7 11	7.93 6	37.7 3
SNCC [165], `H`	20.4 4	15.6 5	12.5 4	24.5 5	12.2 6	10.2 3	20.1 6	30.0 4	15.6 12	30.3 8	54.0 13	16.4 3	16.9 5	49.2 9	8.66 8	40.7 6
SGM [118], `Q`	21.0 5	14.9 3	15.0 9	26.4 9	14.3 10	13.2 7	22.7 8	36.6 7	15.6 11	26.3 4	38.8 6	20.0 11	16.9 4	47.7 6	8.00 7	41.1 7
SGM [118], `F`	22.1 6	28.4 14	**6.52** 1	20.1 3	13.9 9	11.7 5	19.7 5	33.2 5	15.5 10	30.0 7	58.3 16	18.5 6	23.8 13	49.5 10	7.38 4	49.9 14
LPS [163], `H`	23.2 7	**11.4** 1	32.3 16	21.7 4	**5.72** 1	68.5 15	19.0 3	50.2 9	**8.89** 1	25.9 3	**20.9** 1	18.0 4	16.5 3	**39.7** 1	6.96 3	**26.6** 1
SGBM1 [164], `H`	23.3 8	25.2 11	13.7 7	26.1 8	11.8 5	21.3 8	25.6 10	48.5 8	11.4 3	31.4 10	40.2 7	19.9 9	17.7 7	47.3 5	11.8 10	45.1 10
Cens5 [160], `H`	23.3 9	23.5 10	13.6 6	25.2 6	15.0 11	12.7 6	23.2 9	35.3 6	15.2 7	33.2 12	55.9 14	19.5 7	20.4 9	51.2 13	9.48 9	51.3 15
SGBM2 [164], `Q`	25.5 10	18.6 8	23.3 13	34.2 12	15.2 12	32.1 11	26.9 12	54.4 12	15.4 9	28.4 5	26.8 3	21.0 13	17.8 8	45.8 3	15.9 16	42.0 9
LPS [163], `F`	26.2 11	18.6 7	29.3 15	26.6 10	6.52 2	70.0 16	21.9 7	52.0 11	9.95 2	29.1 6	24.3 2	21.2 14	22.2 12	43.3 2	7.82 5	38.1 4
SGBM1 [164], `Q`	26.4 12	25.5 12	21.3 12	34.5 13	16.1 13	22.3 9	31.5 13	51.6 10	16.2 13	31.7 11	28.8 4	24.0 15	21.2 10	48.0 7	14.8 14	45.4 11
SGBM1 [164], `F`	27.4 13	35.4 16	14.1 8	27.9 11	13.9 8	38.5 12	26.4 11	56.0 13	13.7 6	37.4 13	44.8 8	20.2 12	21.9 11	50.2 12	12.1 11	48.6 13
ELAS [161], `H`	31.9 14	22.4 9	17.7 11	46.4 16	21.0 14	47.3 13	35.5 15	60.9 14	24.6 14	40.3 15	52.2 11	19.9 10	27.5 14	51.8 14	13.5 12	41.5 8
ELAS [161], `F`	32.6 15	26.0 13	16.3 10	43.1 15	21.6 15	50.5 14	33.4 14	65.3 16	26.4 15	38.9 14	46.9 9	18.4 5	31.3 15	55.4 15	14.4 13	46.7 12
BSM [159], `Q`	37.1 16	33.7 15	27.6 14	41.5 14	31.0 16	29.7 10	39.2 16	61.2 15	30.3 16	46.7 16	57.1 15	40.8 16	34.4 16	58.3 16	15.7 15	56.4 16

[1] Resolutions at which the methods were evaluated follow their acronyms: `Q` - quarter-size, `H` - half-size, `F` - full-size.

Table 5.10: Dense results on the Middlebury 3.0 training set using the `bad4.0` metric in non-occluded regions.

Method	Avg. `bad4.0`	1 `Adiron`	1 `ArtL`	1 `Jadepl`	1 `Motor`	1 `MotorE`	1 `Piano`	0.5 `PianoL`	1 `Pipes`	0.5 `Playrm`	0.5 `Playt`	1 `PlaytP`	1 `Recyc`	0.5 `Shelvs`	1 `Teddy`	0.5 `Vintge`
IDR, `H`	**12.0** 1	6.57 3	8.13 3	12.7 2	6.02 4	**4.86** 1	13.2 2	**21.2** 1	8.85 5	14.2 2	41.1 11	**8.11** 1	**6.94** 1	40.9 10	**3.87** 1	24.8 2
SGM [118], `H`	12.1 2	7.95 4	5.92 2	**12.3** 1	5.91 3	4.87 2	**11.7** 1	22.1 2	7.87 3	**12.1** 1	42.4 13	10.3 2	8.42 3	35.4 3	4.70 3	31.4 6
SGM [118], `Q`	12.9 3	**5.43** 1	10.6 5	18.1 5	6.32 5	5.71 3	14.5 3	26.6 6	10.1 7	14.6 3	25.8 6	11.4 4	8.34 2	35.0 2	5.00 4	30.3 4
LAMC [162], `H`	15.0 4	10.8 8	10.7 6	20.3 9	7.40 8	6.63 5	15.6 5	23.8 3	12.4 13	23.5 12	23.2 5	14.9 13	11.4 9	42.2 13	6.33 7	29.1 3
SNCC [165], `H`	15.2 5	9.37 5	9.97 4	19.0 7	7.00 7	6.16 4	16.8 7	24.5 5	12.1 12	22.0 9	45.5 14	11.2 3	10.6 8	38.1 6	6.48 8	32.4 9
SGM [118], `F`	16.5 6	20.5 15	**4.92** 1	15.5 3	8.67 11	7.06 6	15.2 4	24.1 4	11.5 10	19.1 7	49.8 16	13.8 10	17.3 14	39.5 9	5.51 5	41.4 15
Cens5 [160], `H`	17.2 7	15.0 11	10.9 7	18.3 6	8.70 12	7.46 7	18.9 9	28.7 7	11.7 11	23.3 11	46.5 15	13.1 7	12.3 10	41.2 11	7.54 9	42.4 16
SGBM2 [164], `Q`	17.3 8	9.40 6	17.3 13	25.3 12	7.42 9	21.5 11	19.8 10	45.6 11	9.93 6	16.8 4	15.9 2	13.2 9	9.08 4	37.0 4	10.3 15	32.0 7
SGBM1 [164], `Q`	17.6 9	13.0 10	15.9 12	25.9 13	7.44 10	12.3 8	23.0 13	42.5 8	10.5 9	19.0 6	18.0 4	15.6 15	10.2 6	38.4 7	9.96 14	35.9 10
SGBM1 [164], `H`	17.7 10	17.7 14	11.9 8	19.9 8	6.50 6	16.0 10	21.2 11	42.8 9	8.29 4	22.9 10	31.5 7	13.8 11	10.2 7	39.4 8	8.60 10	36.8 11
LPS [163], `H`	18.1 11	6.25 2	28.5 16	16.5 4	**3.26** 1	59.4 15	15.6 6	45.0 10	**6.10** 1	17.4 5	**13.9** 1	13.1 7	9.26 5	**32.5** 1	4.69 2	**18.5** 1
LPS [163], `F`	21.0 12	12.0 9	26.4 15	22.0 10	4.10 2	59.8 16	18.7 8	46.7 12	7.20 2	21.6 8	17.0 3	15.4 14	14.5 11	37.1 5	5.91 6	30.8 5
SGBM1 [164], `F`	22.0 13	28.8 16	12.7 9	22.4 11	8.82 13	33.0 12	22.4 12	50.9 15	10.3 8	29.7 15	36.1 8	14.2 12	14.7 12	43.1 14	8.79 11	39.4 13
BSM [159], `Q`	23.4 14	16.3 12	20.7 14	27.4 14	13.5 16	12.8 9	24.4 14	49.8 13	16.8 14	30.8 16	42.3 12	28.2 16	19.1 15	44.8 15	9.26 12	41.4 14
ELAS [161], `H`	23.6 15	10.6 7	13.0 10	41.0 16	12.1 14	37.8 13	26.2 16	50.8 14	18.3 15	26.8 13	41.1 10	12.3 5	17.2 13	41.8 12	9.61 13	32.2 8
ELAS [161], `F`	25.4 16	16.4 13	13.7 11	39.0 15	13.2 15	41.2 14	25.9 15	57.6 16	20.0 16	27.1 14	36.7 9	12.3 6	22.2 16	46.1 16	10.9 16	37.8 12

[1] Resolutions at which the methods were evaluated follow their acronyms: `Q` - quarter-size, `H` - half-size, `F` - full-size.

Table 5.11: Dense results on the Middlebury 3.0 training set using the `avgerr` metric in non-occluded regions.

Method	Avg. `avgerr`	1 `Adiron`	1 `ArtL`	1 `Jadepl`	1 `Motor`	1 `MotorE`	1 `Piano`	0.5 `PianoL`	1 `Pipes`	0.5 `Playrm`	0.5 `Playt`	1 `PlaytP`	1 `Recyc`	0.5 `Shelvs`	1 `Teddy`	0.5 `Vintge`
SGM [118], `H`	**4.83** 1	2.06 3	2.70 2	9.17 2	2.33 4	2.11 2	**2.82** 1	5.70 2	4.13 3	**3.11** 1	25.6 12	2.60 3	2.39 3	8.31 3	**1.32** 1	14.7 12
SGM [118], `Q`	5.29 2	1.85 2	4.25 5	14.6 4	2.76 6	2.59 5	3.61 5	7.22 4	5.63 8	4.24 3	15.7 8	2.67 4	2.31 2	7.78 2	1.50 3	13.9 11
IDR, `H`	5.33 3	**1.83** 1	3.07 3	**8.89** 1	2.08 3	**1.84** 1	3.72 6	10.3 7	4.89 4	3.43 2	38.9 15	**1.86** 1	**1.80** 1	9.87 7	1.37 2	8.00 2
SGM [118], `F`	5.82 4	4.62 10	**2.34** 1	13.2 3	2.97 8	2.49 3	3.24 2	**5.18** 1	5.35 6	4.40 4	23.9 10	3.61 11	4.08 14	8.83 4	1.52 4	16.2 14
LAMC [162], `H`	6.31 5	3.54 7	5.09 7	18.2 10	3.26 9	3.18 7	4.87 10	8.77 6	6.89 12	7.08 11	9.38 5	4.65 15	2.73 7	11.7 13	2.64 11	10.5 4
SNCC [165], `H`	6.96 6	2.89 5	4.01 4	18.1 9	2.68 5	2.52 4	3.52 4	7.08 3	6.14 9	5.64 6	45.4 16	3.13 6	2.90 9	**7.59** 1	1.54 5	13.5 10
Cens5 [160], `H`	7.25 7	4.15 9	4.84 6	15.9 8	3.46 10	3.13 6	4.75 9	8.72 5	7.14 13	6.45 8	33.7 14	4.24 13	3.78 13	10.0 8	1.71 6	16.1 13
SGBM1 [164], `Q`	8.30 8	4.75 11	7.55 12	22.0 13	3.69 14	7.45 9	6.67 15	30.3 13	6.76 11	7.02 10	6.38 4	4.62 14	2.85 8	11.3 12	4.60 15	10.7 6
SGBM2 [164], `Q`	8.43 9	4.00 8	9.28 13	21.4 12	3.53 12	13.6 11	4.64 8	29.9 12	6.59 10	6.61 9	6.17 3	3.95 12	2.60 6	10.8 10	3.84 14	10.5 3
SGBM1 [164], `H`	8.45 10	9.86 15	6.80 10	15.2 5	2.89 7	11.2 10	6.07 13	32.2 15	5.14 5	9.31 14	13.3 6	3.27 9	2.52 5	12.5 14	3.73 13	10.7 5
SGBM1 [164], `F`	11.3 11	18.4 16	7.45 11	15.7 7	3.48 11	29.3 14	6.51 14	39.1 16	5.37 7	12.8 16	13.5 7	3.24 8	3.44 12	15.1 15	3.00 12	11.1 7
ELAS [161], `F`	12.0 12	5.25 12	5.41 9	57.9 15	3.99 15	17.7 13	5.34 12	18.0 9	8.01 14	7.46 13	16.6 9	2.57 2	5.83 15	10.5 9	2.39 9	18.0 15
ELAS [161], `H`	12.2 13	2.96 6	5.21 8	67.5 16	3.63 13	15.8 12	5.31 11	14.3 8	8.06 15	7.21 12	26.8 13	2.95 5	3.39 11	9.35 5	2.49 10	12.1 9
BSM [159], `Q`	13.4 14	7.27 13	11.4 14	30.5 14	6.67 16	6.52 8	10.8 16	32.1 14	10.5 16	12.5 15	24.4 11	12.8 16	7.42 16	16.4 16	4.88 16	32.8 16
LPS [163], `F`	57.0 15	9.35 14	11.5 15	20.1 11	1.58 2	628 15	4.18 7	18.0 10	4.03 2	5.76 7	3.60 2	3.37 10	3.00 10	11.1 11	1.87 8	11.6 8
LPS [163], `H`	85.3 16	2.15 4	12.6 16	15.4 6	**1.45** 1	999 16	3.40 3	20.1 11	**3.63** 1	4.60 5	**3.24** 1	3.15 7	2.44 4	9.71 6	1.78 7	**4.76** 1

[1] Resolutions at which the methods were evaluated follow their acronyms: `Q` - quarter-size, `H` - half-size, `F` - full-size.

Table 5.12: Dense results on the Middlebury 3.0 training set using the `rms` metric in non-occluded regions.

Method	Avg. `rms`	1 `Adiron`	1 `ArtL`	1 `Jadepl`	1 `Motor`	1 `MotorE`	1 `Piano`	0.5 `PianoL`	1 `Pipes`	0.5 `Playrm`	0.5 `Playt`	1 `PlaytP`	1 `Recyc`	0.5 `Shelvs`	1 `Teddy`	0.5 `Vintge`
SGM [118], `H`	**15.9** 1	8.95 3	10.8 2	**39.9** 1	12.0 4	11.6 2	9.19 3	14.6 2	**17.5** 1	11.2 2	54.9 12	9.17 4	10.1 8	18.0 4	4.69 2	31.7 9
SGM [118], `Q`	17.1 2	**8.35** 1	13.7 4	48.4 3	13.2 6	12.7 4	11.6 7	18.3 4	20.9 6	13.8 4	45.4 9	8.71 3	8.80 4	16.6 2	**4.16** 1	31.8 10
IDR, `H`	17.1 3	8.40 2	11.6 3	40.9 2	10.2 3	**9.93** 1	12.3 10	29.8 8	20.5 5	**11.1** 1	74.8 15	**7.03** 1	**7.11** 1	19.4 7	7.89 10	20.3 2
SGM [118], `F`	17.8 4	15.8 9	**10.3** 1	48.8 4	13.1 5	12.0 3	9.21 4	**12.6** 1	20.1 4	14.8 5	48.6 10	11.2 9	13.3 13	18.7 5	5.38 5	32.2 11
SNCC [165], `H`	20.6 5	13.3 6	13.9 5	61.1 9	13.4 7	13.0 5	9.11 2	17.4 3	21.8 9	17.3 7	82.5 16	13.2 11	11.5 11	**15.2** 1	5.15 3	31.1 8
Cens5 [160], `H`	21.5 6	15.0 7	15.9 7	56.8 8	15.2 11	14.6 6	12.3 9	20.3 5	26.6 15	17.6 8	65.8 14	14.3 12	14.1 14	20.7 8	5.37 4	32.9 13
LAMC [162], `H`	22.0 7	15.0 8	18.1 9	62.2 11	15.3 12	15.3 7	16.7 13	23.5 6	24.8 14	19.6 9	30.6 5	17.8 15	10.8 10	24.1 11	13.9 12	32.5 12
SGBM1 [164], `Q`	25.2 8	17.1 11	21.7 10	62.3 13	16.8 15	28.2 9	17.5 14	66.4 15	24.5 11	24.5 13	22.6 3	16.4 14	9.65 7	23.9 10	20.0 16	25.6 5
SGBM2 [164], `Q`	25.5 9	16.1 10	27.1 15	61.7 10	16.2 14	39.7 13	12.9 11	63.2 13	24.8 13	24.2 12	23.3 4	15.3 13	9.56 6	22.0 9	15.0 13	28.1 7
SGBM1 [164], `H`	25.5 10	32.0 14	23.7 11	50.0 5	14.2 9	35.4 10	16.6 12	64.6 14	21.8 8	27.9 14	34.1 7	11.3 10	8.74 3	25.4 12	17.2 14	25.0 4
ELAS [161], `F`	27.1 11	17.2 12	16.1 8	110 15	15.0 10	36.8 12	11.8 8	31.8 9	23.9 10	21.4 11	41.1 8	8.24 2	16.4 15	19.2 6	6.48 6	39.2 14
ELAS [161], `H`	27.1 12	11.0 5	15.4 6	122 16	14.0 8	36.3 11	11.5 6	29.7 7	24.7 12	20.2 10	60.9 13	9.31 5	10.3 9	17.3 3	6.98 7	25.9 6
SGBM1 [164], `F`	29.7 13	45.2 15	25.3 13	52.0 6	15.8 13	63.1 14	17.5 15	69.3 16	21.8 7	34.0 16	30.8 6	11.0 8	11.8 12	29.0 13	13.3 11	24.9 3
BSM [159], `Q`	35.8 14	24.4 13	29.9 16	87.2 14	23.6 16	23.6 8	27.6 16	56.1 12	32.3 16	32.7 15	51.4 11	33.4 16	23.0 16	31.9 14	18.1 15	75.6 16
LPS [163], `H`	97.7 15	10.0 4	25.8 14	54.6 7	**8.83** 1	999 16	**8.59** 1	37.6 11	18.0 2	13.7 3	10.3 2	9.45 6	8.03 2	67.4 15	7.58 9	**14.3** 1
LPS [163], `F`	119 16	223 16	25.0 12	62.2 12	9.32 2	999 15	9.96 5	33.3 10	19.1 3	16.4 6	**9.98** 1	9.98 7	9.08 5	102 16	7.35 8	55.9 15

[1] Resolutions at which the methods were evaluated follow their acronyms: `Q` - quarter-size, `H` - half-size, `F` - full-size.

Table 5.13: Dense results on the Middlebury 3.0 training set on all pixels with known ground truth.

Method[1]	bad0.5	bad1.0	bad2.0	bad4.0	avgerr	rms	A50	A90	A95	A99
BSM [159], Q	84.4 16	66.7 16	44.8 16	32.6 16	23.5 14	52.2 14	2.10 16	85.0 16	138 14	204 16
Cens5 [160], H	60.0 7	40.5 6	29.7 7	22.5 6	10.6 6	27.0 6	1.02 8	37.7 6	62.1 6	120 8
ELAS [161], F	68.4 12	49.2 13	37.8 14	30.1 15	14.9 8	30.8 7	1.75 14	46.6 9	71.8 8	117 7
ELAS [161], H	75.3 15	53.1 15	37.9 15	28.9 13	15.3 9	31.1 8	1.79 15	50.5 11	69.8 7	116 6
IDR, H	**54.3** 1	**33.0** 1	**22.8** 1	**17.3** 1	8.57 4	23.8 4	**0.75** 1	29.8 4	47.9 4	107 4
LAMC [162], H	57.8 6	38.2 5	29.2 6	24.5 7	14.6 7	38.5 9	0.81 3	46.0 8	96.2 9	172 9
LPS [163], F	57.6 5	40.7 8	33.3 10	27.9 12	61.1 15	150 16	1.46 12	47.7 10	145 16	200 15
LPS [163], H	62.3 9	40.7 7	30.6 8	25.2 8	90.4 16	116 15	1.49 13	45.1 7	141 15	191 13
SGBM1 [164], F	61.0 8	43.2 10	34.5 13	29.1 14	18.9 13	45.8 13	1.24 11	65.4 15	112 13	192 14
SGBM1 [164], H	62.5 10	41.4 9	30.9 9	25.3 9	16.3 10	42.2 12	0.96 5	54.6 14	103 11	185 12
SGBM1 [164], Q	72.7 14	50.3 14	34.1 12	25.6 11	16.3 11	42.1 11	1.21 10	51.3 12	105 12	183 11
SGBM2 [164], Q	71.0 13	48.6 12	33.4 11	25.5 10	16.4 12	41.4 10	1.16 9	52.8 13	103 10	180 10
SGM [118], F	56.7 4	37.5 4	27.6 4	21.4 5	8.53 3	22.4 2	0.99 7	29.2 3	47.6 3	104 2
SGM [118], H	56.6 3	34.8 2	24.0 2	17.6 2	**7.62** 1	**21.2** 1	0.81 2	**27.3** 1	**44.7** 1	**98.5** 1
SGM [118], Q	68.2 11	43.6 11	28.0 5	19.4 3	8.51 2	22.7 3	0.97 6	29.2 2	47.3 2	106 3
SNCC [165], H	55.4 2	35.9 3	26.4 3	20.6 4	10.4 5	26.3 5	0.85 4	35.8 5	57.9 5	113 5

[1] Resolutions at which the methods were evaluated follow their acronyms: Q - quarter-size, H - half-size, F - full-size.

Table 5.14: Sparse results on the Middlebury 3.0 training set in non-occluded regions.

Method[1]	bad0.5	bad1.0	bad2.0	bad4.0	avgerr	rms	A50	A90	A95	A99
BSM [159], Q	82.2 16	61.9 16	37.1 16	23.4 16	13.4 14	35.8 14	1.61 16	38.8 16	79.5 14	171 16
Cens5 [160], H	35.7 5	16.6 5	8.27 5	4.96 5	2.25 5	11.2 6	0.48 6	6.75 6	11.2 6	61.5 6
ELAS [161], F	45.8 7	27.1 8	17.7 10	12.6 11	5.12 11	16.3 9	0.8 10	22.8 12	37.4 12	83.3 8
ELAS [161], H	58.6 14	35.6 14	20.9 13	14.0 12	5.97 12	17.6 11	1.02 13	25.5 14	38.6 13	78.8 7
IDR, H	31.9 3	11.7 2	4.58 2	**2.20** 1	**0.95** 1	**5.47** 1	0.34 3	**1.64** 1	**3.61** 1	**18.4** 1
LAMC [162], H	51.8 9	29.6 10	19.8 11	15.0 13	6.31 13	22.0 13	0.67 8	13.0 10	35.8 11	110 13
LPS [163], F	52.3 10	33.7 12	26.2 15	21.0 15	57.0 15	119 16	1.15 14	28.9 15	115 16	162 15
LPS [163], H	57.9 13	33.9 13	23.2 14	18.1 14	85.3 16	97.7 15	1.29 15	25.1 13	108 15	156 14
SGBM1 [164], F	34.2 4	17.0 6	10.2 6	7.25 7	3.27 7	15.0 7	0.47 5	13.3 11	29.5 10	85.1 9
SGBM1 [164], H	45.8 8	23.5 7	13.6 7	9.20 8	3.87 8	16.2 8	0.63 7	8.52 8	22.6 7	88.1 10
SGBM1 [164], Q	61.8 15	37.2 15	20.1 12	12.0 10	4.81 10	17.7 12	0.88 12	9.20 9	29.2 9	97.3 11
SGBM2 [164], Q	57.6 12	33.0 11	17.3 9	10.2 9	4.28 9	17.0 10	0.8 11	7.32 7	24.7 8	97.5 12
SGM [118], F	**26.4** 1	**9.54** 1	**3.92** 1	2.22 2	1.02 2	6.29 2	**0.32** 1	3.03 2	4.56 2	20.8 2
SGM [118], H	38.5 6	16.4 4	7.52 4	3.92 4	1.64 4	8.27 3	0.47 4	3.43 4	8.77 4	36.5 3
SGM [118], Q	55.6 11	29.1 9	13.7 8	6.77 6	2.43 6	10.6 5	0.76 9	3.66 5	8.38 3	52.2 5
SNCC [165], H	29.8 2	12.4 3	6.03 3	3.54 3	1.56 3	9.15 4	0.34 2	3.42 3	9.69 5	47.0 4

[1] Resolutions at which the methods were evaluated follow their acronyms: Q - quarter-size, H - half-size, F - full-size.

by the error percentiles: the lowest 50th percentile `A50` is that of IDR, whereas the lowest values of `A90`, `A95`, and `A99` come from SGM. Dense results evaluated using all pixels with known ground truth reflect the results in the non-occluded regions.

When sparse results are evaluated, the percentages of incorrectly assigned disparities drop to 31.9, 11.7, 4.58, and 2.2%. The lowest error rates using the `bad0.5`, `bad1.0`, and `bad2.0` metrics are achieved by the full-resolution variant of SGM, while IDR attains the lowest rates when the `bad4.0` metric is considered, simultaneously ranking third on `bad0.5`, and second on both `bad1.0` and `bad2.0`. SNCC [165], a block-based two-stage stereo matching method based on normalized cross-correlation, splits SGM and IDR in the `bad0.5` ranking. The advantage in accuracy of the full-resolution SGM over IDR is expected, since IDR operates on half-size images which, due to downsampling, carry less details than the full-size ones. At the same time, IDR outperforms all other methods in terms of the average values of the `avgerr` and `rms` metrics, which suggests that pixels with high-magnitude disparity errors are rejected by the confidence thresholding operation. In addition, the fill rate, i.e., the percentage of pixels with a disparity assignment, of IDR averages 74.28%, whereas the fill rates of the full-resolution SGM and SNCC are 65.75% and 69.35%.

Furthermore, error rates on the `MotorcycleE` dataset (acquired with camera exposure changed between the views), `ArtL`, and `PianoL` datasets (both acquired under changing illumination) prove that the formulation of the per-pixel matching cost combining census and gradient terms allows for robust matching in the presence of radiometric distortions. In all three cases, the proposed method recovers highly accurate disparity maps, performing particularly well on the `MotorcycleE` and `PianoL`

datasets. Together with the `Playtable` and `Shelves` datasets, the `PianoL` is at the same time among the datasets for which the highest error rates were recorded. One of the views in the `PianoL` dataset contains an overexposure resulting in saturated RGB triplets that cannot be reliably matched; high-magnitude disparity errors appear in the overexposed area around the lamp. Results on the `Playtable` dataset, where significant vertical offsets occur between the matching pixels, show IDR's limited capability to cope with imperfect rectification. Like most methods, IDR was designed under the assumption that the matching pixels are confined within the corresponding image scanlines. This assumption is most severely violated by the `Playtable` dataset. If matching is performed on the perfectly rectified version of the `Playtable`, i.e., the `PlaytableP` dataset, IDR is the top performing method with respect to every single error metric. The scene shown in the `Shelves` dataset, on the other hand, contains a uniformly colored, untextured wall that is prone to mismatches. Note that the matches in the image regions corresponding to the wall, among which a significant number are incorrect, are assigned low confidence values and are thus filtered out in the sparse disparity maps of `Shelves`.

Disparity maps generated for the stereo pairs in the test set are shown in Figures 5.8 and 5.9. Low-resolution ground truth images are provided for reference that were downloaded form the benchmark's website. Note that the ground truth data for the test set in the PFM image format, necessary to generate color-coded ground truth images of the matching resolution, are not publicly available. Also note that the color-coded previews of the ground truth disparity maps were generated based on the true range of disparities, while the disparity maps obtained using IDR were color-coded

based on the computed minimum and maximum disparities, which may cause the two to appear different in shade, even in the correctly recovered areas.

The corresponding quantitative results in tabular form are given in the order that matches the order of the results on the training datasets. First, average values of the error metrics computed for the dense disparity maps in non-occluded regions are given in Table 5.15. Per-dataset results using the `bad*`, `avgerr`, and `rms` metrics are then provided in Tables 5.16 through 5.21. Error metrics evaluated for the dense maps over using all pixels for which the true disparities are known are given in Table 5.23. Table 5.22 contains error rates computed for the sparse maps in non-occluded regions.

IDR's performance on the test set closely reflects the results achieved on the training set. The average percentages of incorrectly assigned disparities achieved by the proposed method using subsequent error thresholds of 0.5, 1.0, 2.0, and 4.0 pixels are 52.1, 29.7, 18.1, and 12.7%. Correspondingly, IDR ranks first when the dafault `bad2.0` metric is considered, second using the `bad0.5` and `bad1.0` metrics (with the full-resolution LPS [163] being the top-ranked method in both cases), and third using the `bad4.0` metric (right after quarter-size and half-size variants of SGM). If the error percentages are evaluated over all pixels with known ground truth, individual ranks of IDR remain largely the same with the exception of the `bad4.0` metric, where IDR ranks second, separating half-size and quarter-size SGM.

Exceptional performance of LPS on the test set comes from its two-stage operation that combines local and global matching enhanced with plane fitting, and the ability to handle imperfect rectification. In the first stage, sparse feature matching is performed that allows for non-zero vertical disparities. The vertical disparities are then refined

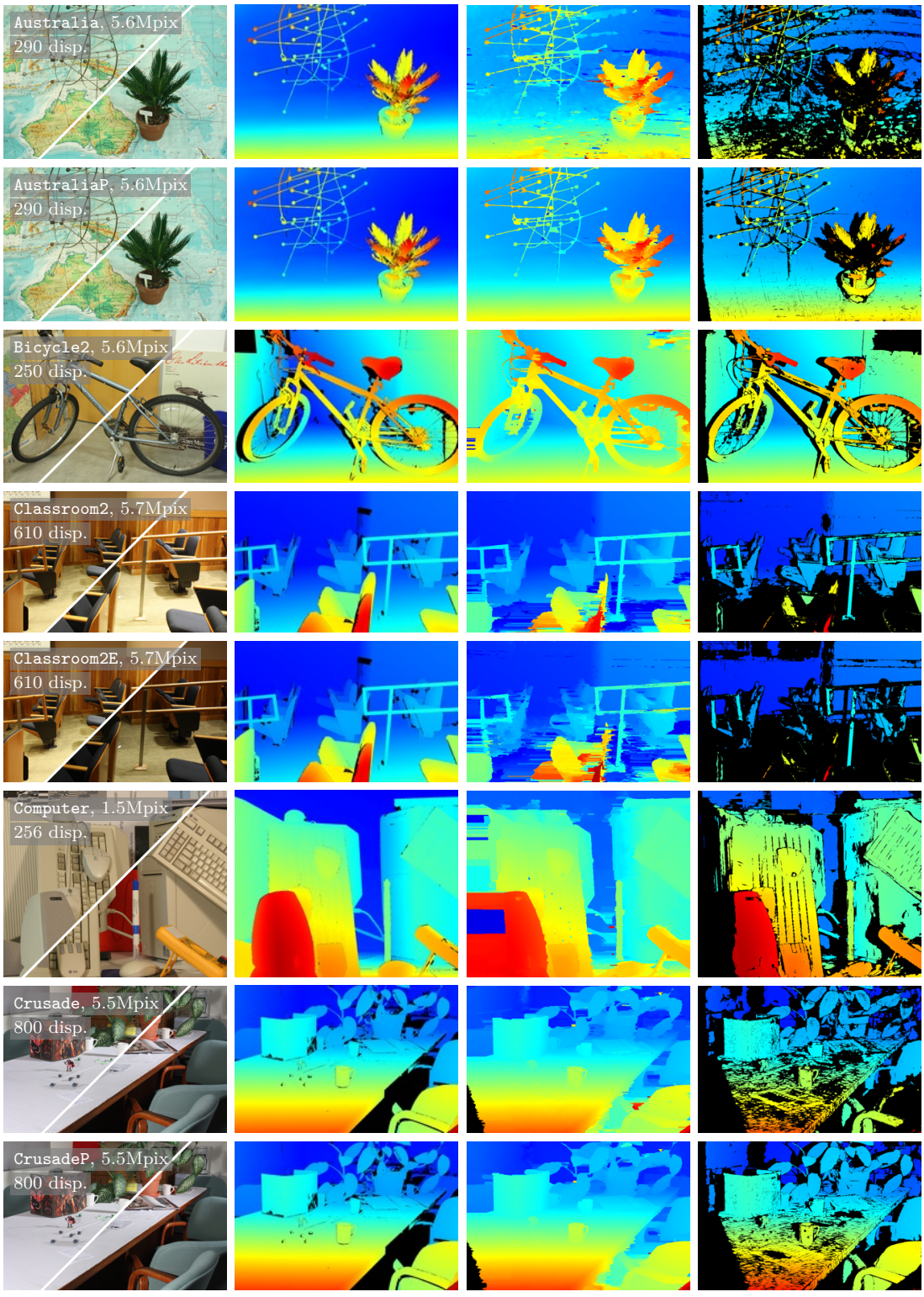


Figure 5.10: Disparity maps computed for the Middlebury 3.0 test set (1 of 2)

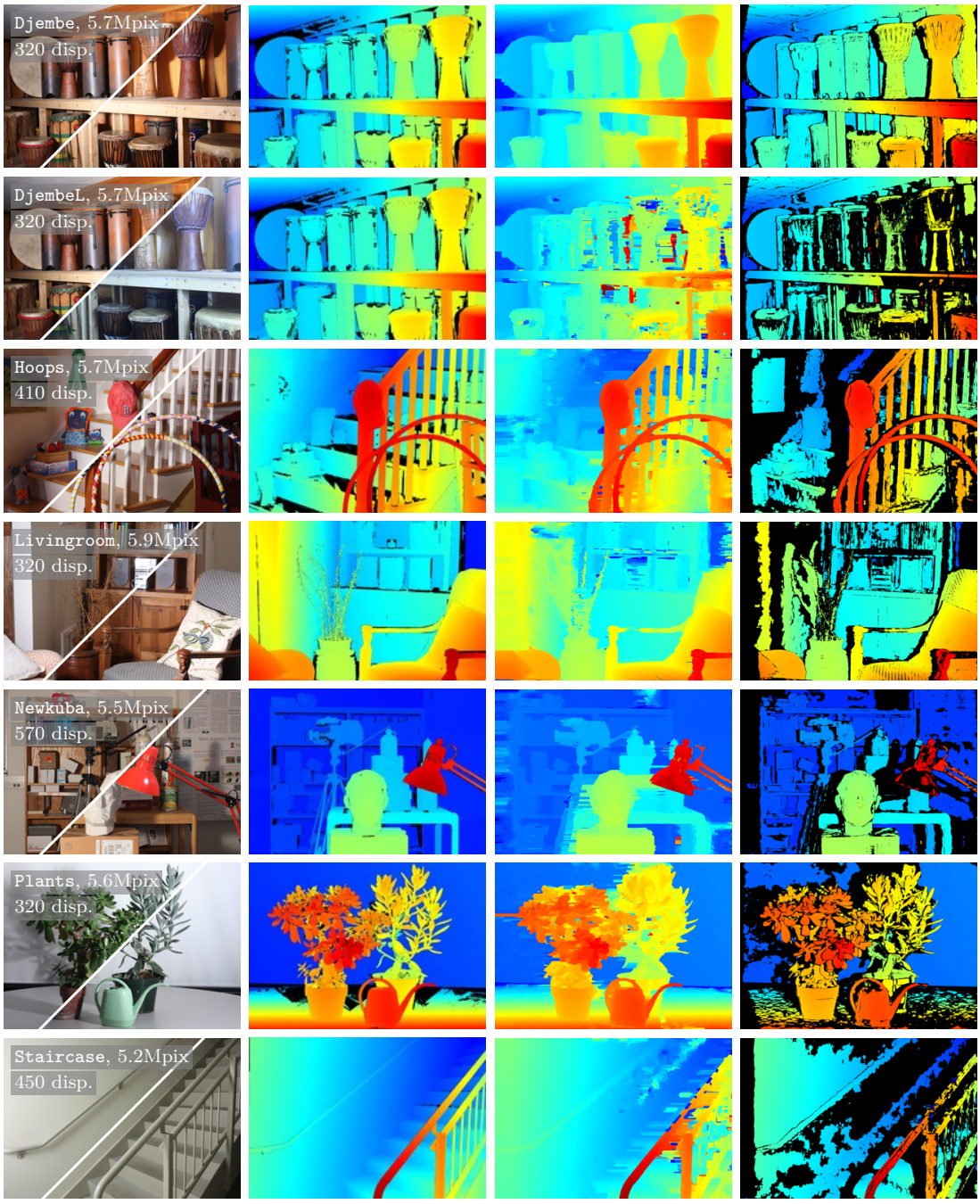


Figure 5.11: Disparity maps computed for the Middlebury 3.0 test set (2 of 2)

Table 5.15: Dense results on the Middlebury 3.0 test set in non-occluded regions.

Method[1]	bad0.5	bad1.0	bad2.0	bad4.0	avgerr	rms	A50	A90	A95	A99
BSM [159], Q	83.4 16	66.3 16	41.5 16	26.9 16	19.9 13	54.0 13	2.23 9	55.1 13	115 15	263 16
Cens5 [160], H	59.0 10	39.0 10	26.6 11	18.6 9	9.34 7	29.4 7	0.82 5	22.5 7	58.7 8	156 9
ELAS [161], F	66.1 11	45.5 13	33.1 15	25.5 15	15.0 10	33.9 9	4.50 13	38.9 12	67.4 9	137 5
ELAS [161], H	71.3 15	47.5 15	29.3 14	21.0 13	10.5 8	28.1 6	2.84 10	23.4 8	52.0 6	131 4
IDR, H	52.1 2	29.7 2	**18.1** 1	12.7 3	6.35 3	21.8 3	**0.62** 1	17.3 3	33.3 3	**102** 1
LAMC [162], H	56.9 8	36.8 8	26.0 9	19.2 12	8.98 6	31.3 8	0.79 4	21.5 6	53.6 7	141 7
LPS [163], F	**48.2** 1	**27.6** 1	20.3 4	16.6 6	84.6 15	115 15	4.99 14	85.9 16	100 14	173 11
LPS [163], H	56.1 6	30.9 3	19.2 3	15.5 4	85.8 16	156 16	6.75 16	85.5 15	92.8 13	171 10
SGBM1 [164], F	56.8 7	38.0 9	28.4 13	23.4 14	22.8 14	55.7 14	5.27 15	62.7 14	136 16	229 15
SGBM1 [164], H	58.8 9	36.0 7	23.8 7	18.5 8	16.0 11	43.3 11	3.96 12	38.2 11	77.3 10	201 13
SGBM1 [164], Q	71.3 14	47.3 14	27.9 12	18.7 10	14.3 9	41.8 10	1.27 8	34.4 9	77.7 11	198 12
SGBM2 [164], Q	68.2 13	44.2 12	26.4 10	18.0 7	17.1 12	48.4 12	3.63 11	37.0 10	90.3 12	218 14
SGM [118], F	55.0 5	35.8 6	25.3 8	19.0 11	7.97 5	25.6 4	0.83 6	18.0 4	41.3 5	141 8
SGM [118], H	54.4 4	31.1 4	18.4 2	12.2 2	**5.32** 1	**20.0** 1	0.66 2	9.87 2	23.8 2	109 2
SGM [118], Q	66.6 12	40.4 11	20.8 5	**11.8** 1	5.38 2	21.0 2	0.86 7	**6.78** 1	**22.5** 1	112 3
SNCC [165], H	53.3 3	32.9 5	21.9 6	15.8 5	7.82 4	26.1 5	0.7 3	18.1 5	40.8 4	139 6

[1] Resolutions at which the methods were evaluated follow their acronyms: Q - quarter-size, H - half-size, F - full-size.

Table 5.16: Dense results on the Middlebury 3.0 test set using the `bad0.5` metric in non-occluded regions.

Method	Avg. `bad0.5`	0.5 `Austr`	1 `AustrP`	1 `Bicyc2`	1 `Class`	0.5 `ClassE`	1 `Compu`	1 `Crusa`	1 `CrusaP`	1 `Djemb`	0.5 `DjembL`	0.5 `Hoops`	1 `Livgrm`	1 `Nkuba`	1 `Plants`	0.5 `Stairs`
LPS [163], `F`	**48.2** 1	**25.2** 1	**28.4** 1	28.1 3	**29.8** 1	98.3 13	58.8 2	**50.7** 1	56.9 2	28.3 2	96.5 15	60.5 2	**41.7** 1	60.3 8	**54.9** 1	49.1 2
IDR, `H`	52.1 2	73.3 4	29.2 2	**27.8** 1	49.5 7	76.3 3	64.3 9	71.6 4	**56.5** 1	32.1 6	61.0 4	**59.7** 1	45.4 2	57.5 3	59.0 4	**47.3** 1
SNCC [165], `H`	53.3 3	79.2 12	30.8 4	28.4 4	49.0 6	76.4 4	61.1 6	77.2 10	60.3 5	31.7 5	**56.0** 1	62.5 4	46.3 3	57.4 2	61.6 5	52.0 3
SGM [118], `H`	54.4 4	75.9 7	33.1 5	31.8 5	52.8 9	76.1 2	62.4 7	72.3 5	57.1 3	36.0 8	62.7 5	61.7 3	48.3 4	58.6 5	58.5 3	62.7 5
SGM [118], `F`	55.0 5	78.8 10	29.4 3	27.9 2	54.5 10	77.1 5	59.3 3	80.8 14	59.2 4	29.3 3	59.5 3	63.4 6	50.8 7	**56.2** 1	62.8 6	76.0 12
LPS [163], `H`	56.1 6	43.4 2	44.5 9	37.7 9	35.6 2	98.7 15	62.9 8	59.9 2	60.3 6	40.6 10	97.6 16	63.8 7	57.8 11	60.1 7	58.3 2	63.5 6
SGBM1 [164], `F`	56.8 7	76.4 8	41.1 7	36.9 8	48.0 5	91.9 12	**56.8** 1	77.4 11	65.5 8	**26.3** 1	76.8 9	66.0 9	48.3 5	58.4 4	62.9 7	67.0 8
LAMC [162], `H`	56.9 8	84.3 15	39.9 6	34.5 6	47.0 4	78.4 7	65.5 10	79.0 12	66.9 10	30.4 4	58.3 2	63.0 5	50.2 6	61.1 10	65.5 10	59.4 4
SGBM1 [164], `H`	58.8 9	71.8 3	49.2 10	43.9 10	46.5 3	88.6 10	61.0 5	70.5 3	65.1 7	35.2 7	76.6 8	65.2 8	51.3 8	60.0 6	64.3 8	74.7 11
Cens5 [160], `H`	59.0 10	79.0 11	41.1 7	34.9 7	52.5 8	77.2 6	68.3 11	79.4 13	68.7 12	36.2 9	65.5 6	66.6 10	52.0 9	60.5 9	65.1 9	68.7 9
ELAS [161], `F`	66.1 11	82.4 14	49.8 11	47.1 11	69.4 15	99.5 16	60.6 4	82.3 15	68.5 11	50.1 11	76.3 7	69.7 11	57.0 10	61.6 11	83.3 15	65.0 7
SGM [118], `Q`	66.6 12	77.1 9	54.3 13	54.2 12	63.6 12	**74.7** 1	70.3 14	73.6 6	66.3 9	63.0 14	77.1 10	72.7 12	62.4 12	67.9 12	68.6 11	74.5 10
SGBM2 [164], `Q`	68.2 13	73.6 5	52.4 12	56.2 13	61.3 11	90.8 11	70.5 15	74.8 7	69.7 13	59.6 12	87.6 13	73.2 13	64.0 13	69.3 13	72.2 12	79.6 14
SGBM1 [164], `Q`	71.3 14	74.3 6	66.0 15	64.6 15	64.2 13	85.3 8	68.4 12	75.7 8	74.2 15	63.4 15	85.5 12	74.8 14	65.8 14	69.4 15	75.9 14	86.5 15
ELAS [161], `H`	71.3 15	79.7 13	60.7 14	62.7 14	69.4 14	98.7 14	68.7 13	77.1 9	71.0 14	61.7 13	81.2 11	76.5 15	69.2 15	69.3 14	74.8 13	76.6 13
BSM [159], `Q`	83.4 16	89.0 16	78.9 16	79.8 16	81.9 16	88.2 9	81.8 16	84.7 16	82.0 16	79.9 16	89.2 14	86.3 16	83.5 16	84.6 16	84.2 16	90.2 16

[1] Resolutions at which the methods were evaluated follow their acronyms: `Q` - quarter-size, `H` - half-size, `F` - full-size.

Table 5.17: Dense results on the Middlebury 3.0 test set using the `bad1.0` metric in non-occluded regions.

Method	Avg. `bad1.0`	0.5 `Austr`	1 `AustrP`	1 `Bicyc2`	1 `Class`	0.5 `ClassE`	1 `Compu`	1 `Crusa`	1 `CrusaP`	1 `Djemb`	0.5 `DjembL`	0.5 `Hoops`	1 `Livgrm`	1 `Nkuba`	1 `Plants`	0.5 `Stairs`
LPS [163], `F`	**27.6** 1	**10.1** 1	**8.09** 1	14.9 3	**15.6** 1	96.6 13	20.9 2	**23.3** 1	**23.3** 1	**10.9** 1	93.6 15	46.7 4	**27.3** 1	33.1 4	**27.7** 1	33.1 2
IDR, `H`	29.7 2	54.9 6	8.72 2	**13.0** 1	30.2 5	**52.2** 1	22.1 8	48.9 5	29.3 3	13.7 4	44.2 5	**41.2** 1	30.6 2	**31.7** 1	31.7 3	**30.1** 1
LPS [163], `H`	30.9 3	15.3 2	15.5 6	16.5 6	17.0 2	97.6 15	21.2 4	28.5 2	29.7 4	14.9 9	95.5 16	42.3 2	35.1 6	35.2 8	27.9 2	39.8 4
SGM [118], `H`	31.1 4	58.1 8	10.2 4	15.5 5	30.4 6	54.6 2	21.2 5	48.8 4	27.9 2	13.9 5	44.1 4	44.2 3	33.1 3	32.4 3	31.9 4	44.7 6
SNCC [165], `H`	32.9 5	64.2 13	12.2 5	15.3 4	32.1 7	56.3 4	21.3 7	57.4 10	34.8 6	14.8 8	**36.0** 1	47.4 6	33.7 4	32.2 2	37.2 7	37.1 3
SGM [118], `F`	35.8 6	62.3 10	9.47 3	13.7 2	39.0 13	59.6 7	20.9 3	64.4 14	34.3 5	14.7 7	42.4 3	49.3 7	39.1 11	33.4 6	40.8 8	63.4 14
SGBM1 [164], `H`	36.0 7	51.8 3	24.1 11	21.4 10	24.4 3	78.5 10	**20.8** 1	47.8 3	41.9 8	13.4 3	63.5 9	49.8 8	34.6 5	34.6 7	35.4 5	59.2 12
LAMC [162], `H`	36.8 8	71.6 15	19.4 9	19.5 9	26.7 4	57.0 5	27.2 10	60.9 12	45.1 11	13.9 6	41.4 2	47.4 5	37.3 8	36.1 9	43.8 10	44.1 5
SGBM1 [164], `F`	38.0 9	59.9 9	19.3 8	19.5 8	32.9 8	86.2 12	21.2 6	59.6 11	44.2 10	11.4 2	67.0 11	54.5 12	36.0 7	33.3 5	37.1 6	53.4 8
Cens5 [160], `H`	39.0 10	63.2 12	19.0 7	19.2 7	33.7 11	57.3 6	30.0 14	61.3 13	47.7 14	19.2 10	49.7 6	51.2 9	39.5 12	36.4 10	43.0 9	55.0 10
SGM [118], `Q`	40.4 11	58.0 7	24.1 12	28.0 12	35.8 12	54.9 3	31.4 15	50.9 6	40.1 7	31.7 13	58.4 7	52.6 10	37.8 10	42.7 12	43.9 11	54.2 9
SGBM2 [164], `Q`	44.2 12	52.0 4	24.8 13	31.3 13	33.1 9	83.3 11	28.3 11	54.2 7	47.4 12	30.7 12	77.2 13	54.5 13	41.1 13	46.6 14	50.2 12	63.4 13
ELAS [161], `F`	45.5 13	67.3 14	22.0 10	24.5 11	47.4 15	99.0 16	24.0 9	66.4 15	43.5 9	26.0 11	61.5 8	53.5 11	37.4 9	40.9 11	71.8 16	47.3 7
SGBM1 [164], `Q`	47.3 14	52.7 5	38.4 15	37.9 15	33.5 10	73.0 8	28.6 12	55.8 8	53.4 15	34.6 15	73.1 12	55.8 14	42.0 14	47.6 15	55.0 14	74.8 15
ELAS [161], `H`	47.5 15	62.8 11	30.6 14	33.9 14	46.8 14	97.5 14	29.7 13	57.4 9	47.6 13	33.3 14	65.8 10	57.9 15	45.2 15	46.5 13	52.4 13	57.3 11
BSM [159], `Q`	66.3 16	78.4 16	58.1 16	60.2 16	64.6 16	77.1 9	50.2 16	70.4 16	64.8 16	60.9 16	79.9 14	73.1 16	67.4 16	70.0 16	67.9 15	80.9 16

[1] Resolutions at which the methods were evaluated follow their acronyms: `Q` - quarter-size, `H` - half-size, `F` - full-size.

Table 5.18: Dense results on the Middlebury 3.0 test set using the `bad2.0` metric in non-occluded regions.

Method	Avg. `bad2.0`	0.5 `Austr`	1 `AustrP`	1 `Bicyc2`	1 `Class`	0.5 `ClassE`	1 `Compu`	1 `Crusa`	1 `CrusaP`	1 `Djemb`	0.5 `DjembL`	0.5 `Hoops`	1 `Livgrm`	1 `Nkuba`	1 `Plants`	0.5 `Stairs`
IDR, `H`	**18.1** 1	37.5 7	**4.08** 1	7.49 2	23.3 9	40.6 3	12.8 5	24.5 3	11.3 4	5.46 3	33.1 5	**26.0** 1	21.5 2	21.7 3	15.3 3	**21.2** 1
SGM [118], `H`	18.4 2	40.3 8	4.54 3	8.03 3	22.9 8	40.5 2	**11.4** 1	24.7 4	10.1 2	5.40 2	29.6 3	28.5 3	23.9 4	**20.0** 1	14.2 2	30.9 5
LPS [163], `H`	19.2 3	**6.14** 1	5.34 4	9.24 4	**7.53** 1	96.0 14	12.3 2	**9.61** 1	**9.40** 1	**5.18** 1	92.4 16	27.4 2	24.3 5	23.0 6	**10.0** 1	25.6 3
LPS [163], `F`	20.3 4	6.72 2	6.06 5	9.72 5	9.87 2	94.3 13	14.1 8	11.2 2	11.2 3	5.88 5	89.3 15	36.0 7	**20.5** 1	23.8 7	16.0 4	25.4 2
SGM [118], `Q`	20.8 5	35.5 6	9.57 10	13.8 10	16.5 6	**32.1** 1	19.0 13	25.8 5	16.7 6	8.95 10	39.8 7	31.1 4	22.6 3	20.7 2	21.3 8	32.2 6
SNCC [165], `H`	21.9 6	48.6 13	6.98 6	9.79 6	25.7 11	46.0 6	12.4 3	36.8 10	16.6 5	7.25 8	**23.1** 1	34.2 5	26.7 9	21.8 4	19.9 6	28.4 4
SGBM1 [164], `H`	23.8 7	32.9 5	10.8 11	13.6 8	16.2 5	71.2 10	12.6 4	26.6 6	23.0 9	5.83 4	53.8 10	39.2 12	25.6 6	22.8 5	18.8 5	47.4 13
SGM [118], `F`	25.3 8	45.1 11	4.33 2	**6.87** 1	32.2 14	50.0 7	13.0 6	48.1 16	18.3 7	7.66 9	29.6 2	36.1 8	31.2 15	24.2 8	24.5 9	50.2 14
LAMC [162], `H`	26.0 9	55.8 15	11.9 13	14.3 11	18.3 7	44.0 4	18.3 12	39.9 11	29.5 13	6.67 7	31.1 4	34.5 6	28.8 12	26.3 13	30.1 12	35.7 8
SGBM2 [164], `Q`	26.4 10	27.9 3	12.1 14	17.8 14	13.7 3	74.5 11	14.0 7	30.3 7	26.3 11	11.0 12	64.4 13	37.9 9	25.8 7	25.3 11	29.3 11	43.7 10
Cens5 [160], `H`	26.6 11	47.1 12	8.74 7	11.9 7	25.6 10	45.3 5	19.5 14	40.6 12	29.0 12	9.93 11	36.5 6	38.6 10	31.0 14	25.0 10	25.6 10	44.6 11
SGBM1 [164], `Q`	27.9 12	28.3 4	17.2 15	19.0 15	14.5 4	57.9 8	15.6 10	31.8 9	31.4 15	13.2 13	58.6 11	38.6 11	27.0 10	25.9 12	31.4 13	59.7 15
SGBM1 [164], `F`	28.4 13	43.5 10	9.09 9	13.6 9	25.9 12	82.0 12	14.4 9	43.4 13	30.3 14	5.98 6	59.3 12	45.8 15	28.5 11	24.9 9	20.1 7	45.9 12
ELAS [161], `H`	29.3 14	41.8 9	11.1 12	16.6 13	27.9 13	96.0 15	19.8 15	31.6 8	21.4 8	14.1 14	50.8 9	40.2 14	29.2 13	29.7 14	32.3 14	37.3 9
ELAS [161], `F`	33.1 15	49.3 14	8.96 8	14.5 12	35.1 15	98.0 16	16.4 11	47.4 15	24.2 10	14.9 15	49.7 8	39.7 13	26.2 8	30.1 15	60.8 16	34.6 7
BSM [159], `Q`	41.5 16	59.8 16	25.8 16	27.9 16	38.9 16	60.6 9	33.3 16	46.9 14	37.3 16	26.3 16	64.8 14	51.5 16	42.6 16	45.2 16	42.8 15	66.6 16

[1] Resolutions at which the methods were evaluated follow their acronyms: `Q` - quarter-size, `H` - half-size, `F` - full-size.

Table 5.19: Dense results on the Middlebury 3.0 test set using the `bad4.0` metric in non-occluded regions.

Method	Avg. `bad4.0`	0.5 `Austr`	1 `AustrP`	1 `Bicyc2`	1 `Class`	0.5 `ClassE`	1 `Compu`	1 `Crusa`	1 `CrusaP`	1 `Djemb`	0.5 `DjembL`	0.5 `Hoops`	1 `Livgrm`	1 `Nkuba`	1 `Plants`	0.5 `Stairs`
SGM [118], `Q`	**11.8** 1	20.1 5	6.25 8	7.99 7	9.98 5	**21.9** 1	12.0 12	9.53 3	5.64 4	3.73 4	27.5 6	18.2 3	**14.6** 1	**13.7** 1	10.9 4	18.2 2
SGM [118], `H`	12.2 2	26.7 8	3.56 3	5.02 2	20.0 8	34.4 2	**6.61** 1	9.90 4	**3.27** 1	**2.70** 1	19.8 2	17.2 2	17.8 4	15.0 2	8.47 2	21.2 6
IDR, `H`	12.7 3	24.9 7	**3.28** 1	5.63 3	20.9 9	35.0 3	7.96 2	10.4 5	4.35 2	3.16 2	27.9 7	**16.5** 1	16.0 2	17.0 4	9.60 3	**16.0** 1
LPS [163], `H`	15.5 4	**4.75** 1	4.38 4	6.24 4	**4.95** 1	93.9 15	8.06 3	**4.89** 1	4.97 3	3.27 3	87.8 16	19.5 4	18.8 5	18.6 5	**6.69** 1	19.8 3
SNCC [165], `H`	15.8 5	34.7 14	6.19 7	7.65 6	22.4 12	41.1 6	8.34 5	20.2 10	7.12 6	4.74 9	**17.4** 1	25.1 6	23.0 11	16.2 3	11.6 5	21.1 5
LPS [163], `F`	16.6 6	5.68 2	5.23 5	7.03 5	7.30 2	90.6 13	9.32 9	6.47 2	6.87 5	4.25 7	82.6 15	27.8 10	16.3 3	19.1 7	11.8 6	20.6 4
SGBM2 [164], `Q`	18.0 7	13.7 3	8.67 13	12.3 13	7.33 3	69.5 11	8.63 7	15.7 6	14.5 12	4.74 9	54.5 14	27.8 9	18.8 6	18.6 6	20.2 11	26.0 9
SGBM1 [164], `H`	18.5 8	21.2 6	7.39 12	11.1 11	13.1 6	67.7 10	8.36 6	16.2 9	13.0 10	3.85 6	47.1 10	33.4 14	20.0 8	19.6 8	13.8 7	40.5 14
Cens5 [160], `H`	18.6 9	34.6 13	6.46 10	8.60 8	22.2 11	38.7 5	13.9 14	21.7 11	10.9 9	5.66 12	27.5 5	30.8 13	25.5 14	19.9 10	14.0 8	35.7 11
SGBM1 [164], `Q`	18.7 10	14.2 4	10.5 15	13.7 15	8.22 4	50.5 9	9.66 10	16.1 8	16.1 14	5.94 13	49.3 11	29.0 12	19.5 7	19.8 9	20.2 12	45.3 15
SGM [118], `F`	19.0 11	31.8 11	3.50 2	**4.40** 1	27.9 15	44.7 7	8.17 4	35.1 16	9.11 7	4.99 11	20.2 3	25.1 5	25.8 15	20.7 11	18.5 10	36.4 12
LAMC [162], `H`	19.2 12	39.1 15	10.2 14	12.9 14	14.7 7	35.5 4	13.5 13	23.4 12	15.3 13	4.48 8	26.6 4	25.9 7	23.5 12	21.4 12	21.9 14	30.9 10
ELAS [161], `H`	21.0 13	26.9 9	7.11 11	10.8 10	21.1 10	93.8 14	14.2 15	15.9 7	9.66 8	8.59 14	42.3 9	28.1 11	22.0 10	23.2 14	21.2 13	25.2 8
SGBM1 [164], `F`	23.4 14	30.6 10	6.32 9	11.2 12	22.4 12	79.2 12	9.10 8	34.3 15	23.3 16	3.79 5	53.3 12	40.1 16	23.6 13	21.8 13	15.1 9	39.8 13
ELAS [161], `F`	25.5 15	32.4 12	5.50 6	10.3 9	29.4 16	96.4 16	11.0 11	32.7 14	14.2 11	9.81 15	41.4 8	26.9 8	20.2 9	23.4 15	52.2 16	22.6 7
BSM [159], `Q`	26.9 16	42.3 16	12.0 16	15.9 16	26.6 14	48.7 8	22.6 16	25.3 13	17.0 15	11.3 16	54.2 13	35.4 15	30.8 16	31.5 16	24.9 15	55.5 16

[1] Resolutions at which the methods were evaluated follow their acronyms: `Q` - quarter-size, `H` - half-size, `F` - full-size.

Table 5.20: Dense results on the Middlebury 3.0 test set using the `avgerr` metric in non-occluded regions.

Method	Avg. `avgerr`	0.5 `Austr`	1 `AustrP`	1 `Bicyc2`	1 `Class`	0.5 `ClassE`	1 `Compu`	1 `Crusa`	1 `CrusaP`	1 `Djemb`	0.5 `DjembL`	0.5 `Hoops`	1 `Livgrm`	1 `Nkuba`	1 `Plants`	0.5 `Stairs`
SGM [118], `H`	**5.32** 1	7.93 5	3.16 3	2.20 2	10.5 8	16.2 3	2.45 2	3.50 5	**2.09** 1	**0.98** 1	7.60 3	5.44 2	5.70 3	5.76 2	6.32 2	10.6 6
SGM [118], `Q`	5.38 2	8.06 6	5.29 9	4.20 6	5.22 5	**9.65** 1	3.69 8	3.33 3	2.77 3	1.45 9	10.2 5	6.63 3	4.33 2	**5.67** 1	7.83 4	12.4 10
IDR, `H`	6.35 3	7.58 3	**2.54** 1	2.21 3	16.8 13	24.1 5	3.28 7	3.34 4	2.15 2	0.99 2	13.5 7	**5.13** 1	5.76 4	6.81 3	6.35 3	8.02 2
SNCC [165], `H`	7.82 4	12.1 13	5.16 8	3.54 4	17.4 14	27.1 7	2.95 3	5.92 7	3.51 7	1.25 6	**4.92** 1	9.15 5	7.60 11	11.2 12	8.21 5	8.76 5
SGM [118], `F`	7.97 5	8.64 7	2.90 2	**2.01** 1	14.7 11	23.5 4	**2.38** 1	10.2 10	2.95 4	1.40 8	7.20 2	8.44 4	8.47 12	13.2 14	11.7 10	12.0 9
LAMC [162], `H`	8.98 6	12.5 14	7.89 14	7.57 13	6.36 6	12.9 2	5.29 14	7.82 9	6.93 11	1.45 9	10.1 4	11.5 8	8.68 13	11.0 10	16.1 14	19.4 15
Cens5 [160], `H`	9.34 7	13.5 15	5.70 11	3.88 5	17.5 15	24.6 6	5.78 15	6.32 8	4.83 9	1.53 11	10.6 6	12.7 9	9.13 15	14.0 15	10.3 8	14.1 11
ELAS [161], `H`	10.5 8	9.61 10	5.69 10	5.36 9	12.1 10	82.5 9	3.93 11	5.64 6	4.32 8	2.01 14	17.3 8	12.9 10	5.93 6	8.13 6	12.9 11	8.38 3
SGBM1 [164], `Q`	14.3 9	8.69 8	8.00 15	7.08 12	3.86 4	102 10	5.28 13	15.4 12	15.8 13	2.19 15	33.2 10	18.3 12	6.08 8	10.3 9	14.6 12	18.3 14
ELAS [161], `F`	15.0 10	9.62 11	4.11 6	5.34 8	14.7 12	141 12	3.78 9	10.7 11	6.42 10	2.00 13	18.0 9	13.5 11	6.18 9	8.57 7	28.9 16	10.7 7
SGBM1 [164], `H`	16.0 11	8.84 9	5.70 11	5.50 10	6.94 7	150 13	3.84 10	18.0 13	15.8 13	1.37 7	33.8 11	22.4 14	7.07 10	10.0 8	10.2 7	16.5 12
SGBM2 [164], `Q`	17.1 12	7.70 4	6.57 13	6.33 11	3.42 2	133 11	4.92 12	26.5 15	27.0 15	1.87 12	37.1 12	19.4 13	5.77 5	11.1 11	15.6 13	11.5 8
BSM [159], `Q`	19.9 13	23.4 16	9.58 16	9.29 14	26.7 16	52.0 8	9.81 16	21.6 14	14.9 12	6.31 16	40.5 13	23.9 15	17.8 16	22.6 16	16.8 15	48.0 16
SGBM1 [164], `F`	22.8 14	10.3 12	4.76 7	5.18 7	10.7 9	183 14	3.17 6	46.0 16	40.4 16	1.15 4	42.5 14	29.9 16	9.12 14	12.8 13	10.6 9	17.7 13
LPS [163], `F`	84.6 15	3.65 2	3.37 4	9.41 15	3.65 3	999 15	3.12 5	3.06 2	3.15 6	1.18 5	999 15	10.8 7	**3.92** 1	7.16 4	8.56 6	8.75 4
LPS [163], `H`	85.8 16	**3.37** 1	3.55 5	26.7 16	**2.48** 1	999 16	3.00 4	**2.90** 1	3.05 5	1.07 3	999 16	10.3 6	5.95 7	7.82 5	**6.23** 1	**7.60** 1

[1] Resolutions at which the methods were evaluated follow their acronyms: `Q` - quarter-size, `H` - half-size, `F` - full-size.

Table 5.21: Dense results on the Middlebury 3.0 test set using the `rms` metric in non-occluded regions.

Method	Avg. `rms`	0.5 `Austr`	1 `AustrP`	1 `Bicyc2`	1 `Class`	0.5 `ClassE`	1 `Compu`	1 `Crusa`	1 `CrusaP`	1 `Djemb`	0.5 `DjembL`	0.5 `Hoops`	1 `Livgrm`	1 `Nkuba`	1 `Plants`	0.5 `Stairs`
SGM [118], `H`	**20.0** 1	22.6 4	17.8 3	10.9 3	34.6 9	40.5 2	8.67 2	16.3 4	**12.5** 1	**4.01** 1	22.0 3	23.6 2	17.8 8	27.5 2	26.0 2	40.1 9
SGM [118], `Q`	21.0 2	25.5 6	22.8 8	16.4 6	26.4 5	**31.0** 1	10.7 4	15.2 2	14.4 3	4.61 4	27.1 5	26.8 3	14.3 2	29.7 4	29.6 4	45.9 13
IDR, `H`	21.8 3	22.1 3	**15.6** 1	**10.3** 1	48.3 13	55.1 5	12.8 8	**14.9** 1	13.1 2	4.04 2	35.5 8	**20.6** 1	17.9 9	**27.1** 1	**25.7** 1	31.2 3
SGM [118], `F`	25.6 4	22.9 5	17.0 2	10.4 2	40.4 12	50.4 4	**8.24** 1	29.7 9	15.4 4	6.05 10	21.7 2	29.5 4	21.9 12	53.9 14	36.2 9	37.6 7
SNCC [165], `H`	26.1 5	30.2 13	22.8 9	15.1 4	50.3 15	58.0 7	10.1 3	20.8 6	16.9 5	5.00 6	**15.7** 1	31.2 5	19.1 10	51.7 12	30.9 6	32.7 5
ELAS [161], `H`	28.1 6	27.3 10	24.1 11	18.0 7	38.8 10	122 9	11.2 6	21.6 7	19.5 8	5.92 8	35.3 7	38.6 9	14.8 3	31.7 5	38.3 11	30.3 2
Cens5 [160], `H`	29.4 7	33.1 15	24.5 12	15.8 5	49.0 14	55.7 6	17.6 12	21.7 8	21.0 9	5.55 7	29.7 6	38.1 8	22.5 13	55.0 15	36.3 10	41.5 10
LAMC [162], `H`	31.3 8	31.0 14	28.1 15	23.2 13	29.0 7	41.1 3	17.9 13	37.9 11	40.9 11	7.53 13	26.4 4	36.2 7	24.1 14	44.5 11	45.6 14	50.8 15
ELAS [161], `F`	33.9 9	25.6 7	19.8 6	18.4 8	39.8 11	197 11	12.9 9	30.7 10	24.9 10	6.05 10	38.1 9	39.7 10	16.5 4	32.6 6	54.7 16	33.8 6
SGBM1 [164], `Q`	41.8 10	28.2 12	28.0 14	22.0 12	18.0 3	185 10	19.7 15	72.1 12	72.6 13	7.59 14	61.8 10	46.8 11	17.1 5	39.2 8	40.7 12	50.0 14
SGBM1 [164], `H`	43.3 11	26.9 9	23.6 10	19.1 10	26.6 6	232 13	15.9 11	78.0 13	73.9 14	6.68 12	63.4 11	51.0 13	20.8 11	35.6 7	34.0 7	41.8 11
SGBM2 [164], `Q`	48.4 12	26.5 8	25.6 13	20.6 11	16.7 2	214 12	18.2 14	105 15	109 15	9.36 15	66.3 12	49.7 12	17.2 6	43.4 10	43.1 13	37.7 8
BSM [159], `Q`	54.0 13	50.6 16	32.3 16	26.5 14	76.3 16	112 8	28.0 16	79.9 14	66.1 12	22.8 16	67.9 13	56.8 14	41.0 16	67.9 16	46.1 15	88.6 16
SGBM1 [164], `F`	55.7 14	27.9 11	22.0 7	18.9 9	32.3 8	260 14	13.3 10	137 16	131 16	5.95 9	75.8 14	61.6 15	25.4 15	42.2 9	34.6 8	43.1 12
LPS [163], `F`	115 15	19.4 2	18.0 4	233 15	20.9 4	999 15	12.0 7	16.8 5	17.4 7	4.92 5	999 15	33.0 6	**11.6** 1	28.1 3	30.8 5	31.8 4
LPS [163], `H`	156 16	**17.1** 1	18.4 5	675 16	**15.8** 1	999 16	10.9 5	16.2 3	17.3 6	4.17 3	999 16	147 16	17.2 6	52.9 13	27.0 3	**26.4** 1

[1] Resolutions at which the methods were evaluated follow their acronyms: `Q` - quarter-size, `H` - half-size, `F` - full-size.

Table 5.22: Sparse results on the Middlebury 3.0 test set in non-occluded regions.

Method[1]	bad0.5	bad1.0	bad2.0	bad4.0	avgerr	rms	A50	A90	A95	A99
BSM [159], Q	83.4 16	66.3 16	41.5 16	26.9 16	19.9 14	54.0 14	2.23 11	55.1 14	115 16	263 16
Cens5 [160], H	34.9 5	18.1 6	9.31 5	4.95 5	2.54 5	14.2 6	0.58 5	3.63 6	9.63 6	91.7 6
ELAS [161], F	44.1 7	25.9 8	16.1 9	11.1 11	6.01 11	20.6 9	10.4 16	31.7 13	44.1 10	106 8
ELAS [161], H	55.7 10	33.7 13	17.4 11	10.7 10	4.75 8	17.6 7	2.92 12	15.1 7	29.1 7	93.9 7
IDR, H	30.4 3	11.8 2	4.37 2	1.84 2	**0.95** 1	**6.70** 1	0.48 2	**1.60** 1	**2.72** 1	**18.6** 1
LAMC [162], H	56.9 14	36.8 14	26.0 15	19.2 15	8.98 13	31.3 13	0.79 8	21.5 8	53.6 12	141 13
LPS [163], F	48.2 9	27.6 9	20.3 14	16.6 14	84.6 15	115 15	4.99 14	85.9 16	100 15	173 15
LPS [163], H	56.1 11	30.9 10	19.2 12	15.5 13	85.8 16	156 16	6.75 15	85.5 15	92.8 14	171 14
SGBM1 [164], F	33.4 4	16.7 4	9.44 6	6.52 7	4.11 7	19.5 8	4.84 13	29.3 12	42.7 9	110 9
SGBM1 [164], H	44.7 8	23.0 7	12.3 7	8.20 8	4.99 9	20.8 10	0.64 6	26.0 10	42.6 8	112 10
SGBM1 [164], Q	61.2 15	38.1 15	19.7 13	11.5 12	6.51 12	23.5 12	0.97 10	22.3 9	55.6 13	126 12
SGBM2 [164], Q	56.4 12	33.3 12	16.6 10	9.41 9	5.87 10	23.4 11	0.87 9	26.8 11	50.7 11	120 11
SGM [118], F	**23.0** 1	**8.77** 1	**3.33** 1	**1.73** 1	0.99 2	8.01 2	**0.46** 1	1.82 2	3.60 2	41.5 2
SGM [118], H	38.4 6	17.2 5	7.08 4	3.33 4	1.70 4	10.4 3	0.54 4	2.15 3	4.09 3	56.5 3
SGM [118], Q	56.8 13	31.8 11	13.8 8	6.36 6	2.83 6	13.7 5	0.77 7	3.22 5	7.31 5	68.5 5
SNCC [165], H	27.6 2	12.7 3	6.11 3	3.26 3	1.68 3	11.2 4	0.52 3	2.70 4	6.70 4	64.1 4

[1] Resolutions at which the methods were evaluated follow their acronyms: Q - quarter-size, H - half-size, F - full-size.

Table 5.23: Dense results on the Middlebury 3.0 test set on all pixels with known ground truth.

Method[1]	bad0.5	bad1.0	bad2.0	bad4.0	avgerr	rms	A50	A90	A95	A99
BSM [159], Q	85.6 16	70.9 16	49.3 16	36.6 16	37.1 14	81.1 14	3.51 10	141 15	212 16	314 16
Cens5 [160], H	63.2 9	45.1 10	33.2 9	24.4 6	13.0 6	35.7 7	1.04 5	41.2 7	87.0 7	166 8
ELAS [161], F	69.6 11	51.0 12	39.1 15	31.1 14	18.7 8	40.2 8	4.78 13	56.4 8	96.0 8	153 6
ELAS [161], H	74.4 14	53.1 14	36.3 14	27.4 11	14.2 7	34.6 6	3.12 9	40.7 6	82.5 6	144 3
IDR, H	56.9 2	36.4 2	**25.0** 1	18.7 2	10.1 3	30.1 3	**0.73** 1	27.4 3	**59.7** 1	145 4
LAMC [162], H	62.4 8	45.0 9	35.4 11	29.3 13	23.1 9	60.1 9	1.06 6	79.5 9	156 9	265 10
LPS [163], F	**54.2** 1	**36.0** 1	28.8 5	24.7 8	95.0 16	175 15	4.83 14	121 11	183 11	266 11
LPS [163], H	61.1 6	38.8 4	27.6 3	23.3 5	92.9 15	223 16	6.00 16	108 10	156 10	216 9
SGBM1 [164], F	61.7 7	44.9 8	36.0 12	31.1 15	35.9 13	78.0 13	5.84 15	141 16	193 15	280 13
SGBM1 [164], H	63.6 10	43.4 7	32.3 8	27.0 10	30.2 11	71.3 11	4.59 12	127 13	183 13	279 12
SGBM1 [164], Q	74.7 15	53.6 15	36.2 13	27.6 12	28.9 10	70.6 10	1.68 8	125 12	183 12	290 15
SGBM2 [164], Q	72.0 13	51.0 13	35.0 10	26.9 9	30.9 12	73.7 12	4.06 11	138 14	188 14	284 14
SGM [118], F	59.4 5	41.8 6	31.2 7	24.4 7	11.6 5	32.2 4	1.09 7	34.8 5	69.3 3	159 7
SGM [118], H	59.2 4	38.1 3	25.7 2	**18.6** 1	**9.27** 1	**28.1** 1	0.79 2	**21.8** 1	61.9 2	**134** 1
SGM [118], Q	70.3 12	47.0 11	28.8 6	19.4 3	9.68 2	29.6 2	1.00 4	22.7 2	70.4 4	135 2
SNCC [165], H	57.7 3	38.8 5	27.8 4	20.9 4	11.3 4	32.4 5	0.83 3	34.5 4	74.4 5	153 5

[1] Resolutions at which the methods were evaluated follow their acronyms: Q - quarter-size, H - half-size, F - full-size.

and local disparity plane hypotheses are generated. Finally, disparities are assigned through local plane sweeps around the hypothesized planes using SGM with narrow disparity range and normalized cross-correlation as a cost metric. In the second stage, the image is divided into tiles and plane information is propagated to adjacent tiles, again using SGM. This approach enables LPS to outperform IDR with respect to the most stringent metrics, i.e., `bad0.5` and `bad1.0`, on the datasets with significant rectification errors, or those that contain well-textured planar surfaces, such as the `Australia`, `Classrooom`, and `Crusade` datasets.

The proposed method, on the other hand, exhibits unmatched accuracy in the cases when the views contain fine scene structures, e.g., on the `AustraliaP` and `Bicycle2` datasets, as well as on the `ClassroomE` dataset, which is one of the two datasets in the test set that contain radiometric distortions. The `DjembeL` dataset, which is the remaining one, illustrates a failure case where the cost metric of IDR does not allow for robust matching. The views of the `DjembeL` scene were acquired under different lighting conditions, in that an additional light source was used during acquisition of the right view. As a result, reflections and shadows (absent in the left view) appear in the right view that affect the census vectors and local gradient estimates, preventing correct disparity assignment. Note that such a change in illumination, i.e., the appearance of an extra light source that only affects one of the views, is unlikely to occur in applications of stereo matching, where the views are typically captured using a pair of synchronized cameras.

At the same time, IDR achieves the lowest error rates on the `Hoops` and `Staircase` datasets. It may be argued that these are the most challenging datasets in the test

set, since they contain weakly textured surfaces where transitions of pixel intensities, if any, are barely noticeable. The census and gradient terms in the cost metric of IDR capture even the slightest texture details and intensity transitions, reducing the uncertainty when selecting matches, which ultimately enabled IDR to surpass other methods.

Similarly to the evaluation on the training datasets, absolute and root-mean-square disparity errors associated with incorrectly assigned pixels of the test images are relatively high. With an average `avgerr` of 6.35 and `rms` of 21.8 pixels, the proposed method ranks third despite having the lowest 50th percentile error. Again, the thresholding of confidence values allows for high-magnitude errors to be excluded from the sparse disparity maps, resulting in the top-ranking `avgerr` of 0.95 and `rms` of 2.1 pixels, and the lowest error percentiles among the evaluated methods. The fill rate of IDR is 69.75%, while the full-size SGM and the half-size SNCC, both of which are IDR's competitors in the sparse rankings using the `avgerr` and `rms` metrics, achieve the fill rates of 57.28% and 62.03%. Higher fill rates are only achieved by some of the quarter-size methods. This is expected since operating on lower-resolution datasets, which inherently contain less disparities, is more likely to produce disparities that fall within the consistency threshold.

Matching times recorded for individual datasets in the training and test sets are given in Tables 5.24 and 5.25, respectively. Achieving an average matching time of 0.34s on the training set and 0.49s on the test set, the proposed method is the fastest among those operating on half-resolution images and in the global ranking is outperformed only by the SGBM1 and SGBM2 methods (OpenCV's [164] single-pass

and two-pass reimplementations of Hirschmuller's SGM), both of which operate on quarter-size images. While IDR evaluates 800% of the number of disparity hypotheses evaluated by SGBM1 and SGMB2 (doubling the image dimensions results in 4 times as many pixels and twice as many disparities), the two methods attain efficiency gains of only 19% and 51% over IDR.

Increased execution times were observed during matching of the `Jadeplant` and `Vintage` images in the training set, and the `Classroom*`, `Crusade*`, and `Newkuba` images in the test set, all of which contain broad spectra of disparities. Since cost aggregation using the two-pass approximation of the bilateral filter cannot be performed in-place, as there exists no inter-block synchronization mechanism in CUDA, the proposed method requires two cost volumes to be allocated in the global memory of the graphics hardware. One of these volumes serves as an auxiliary buffer and is used solely to store intermediate filtering results, i.e., the results of the vertical cost aggregation step. The two-pass bilateral filter is efficiently evaluated by sweeping through the layers of the volumes using two kernel calls (first reading from the primary volume and writing to the auxiliary one, then reading from the auxiliary volume and writing back to the primary one), each time loading the guidance data necessary to compute the filter weights. In case of the said datasets, allocation of the auxiliary cost volume was not possible, due to the size of allocation exceeding the amount of available memory, or due to the fragmentation of the global memory space.

To overcome memory limitations when matching images with high number of disparities, the auxiliary cost volume is replaced with an auxiliary image sized to store a single layer of the filtered cost. In this event, the cost is aggregated in a layer-by-

Table 5.24: Matching times (in seconds) recorded on the Middlebury 3.0 training set.

Method	Avg. time	1 Adiron	1 ArtL	1 Jadepl	1 Motor	1 MotorE	1 Piano	0.5 PianoL	1 Pipes	0.5 Playrm	0.5 Playt	1 PlaytP	1 Recyc	0.5 Shelvs	1 Teddy	0.5 Vintge
SGBM1 [164], Q	**0.18** 1	**0.22** 1	**0.04** 1	**0.29** 1	**0.19** 1	**0.19** 1	**0.17** 1	**0.17** 1	**0.18** 1	**0.20** 1	**0.16** 1	**0.16** 1	**0.18** 1	**0.17** 1	**0.08** 1	**0.36** 1
SGBM2 [164], Q	0.29 2	0.3 2	0.07 2	0.51 2	0.3 2	0.3 2	0.28 2	0.28 2	0.29 2	0.34 2	0.27 2	0.26 2	0.29 2	0.27 2	0.12 2	0.63 2
IDR, H	0.34 3	0.37 3	0.08 3	0.68 3	0.33 3	0.33 3	0.29 3	0.29 3	0.34 3	0.35 3	0.3 3	0.3 3	0.3 3	0.29 3	0.14 3	0.92 4
ELAS [161], H	0.72 4	0.83 4	0.20 4	0.84 4	0.88 4	0.77 4	0.74 4	0.71 4	0.77 4	0.77 4	0.81 4	0.81 4	0.80 4	0.83 4	0.38 4	0.91 3
SNCC [165], H	0.97 5	1.03 5	0.24 5	1.67 6	1.05 6	1.09 6	0.82 5	0.82 5	1.07 5	1.08 5	0.90 5	0.91 5	0.95 5	0.94 5	0.42 5	2.06 6
SGM [118], Q	0.99 6	1.08 6	0.25 6	1.63 5	1.04 5	1.04 5	0.94 6	0.97 6	1.09 6	1.08 6	0.97 6	0.96 6	0.98 6	0.95 6	0.42 6	1.92 5
Cens5 [160], H	1.34 7	1.44 7	0.31 8	2.36 7	1.37 8	1.37 8	1.26 8	1.26 8	1.44 7	1.44 7	1.28 7	1.27 7	1.30 8	1.25 8	0.58 8	2.97 7
SGBM1 [164], H	1.48 8	1.60 8	0.29 7	3.42 8	1.34 7	1.34 7	1.21 7	1.23 7	1.54 8	1.53 8	1.32 8	1.31 8	1.25 7	1.17 7	0.5 7	4.25 8
ELAS [161], F	3.56 9	4.12 9	0.89 9	4.22 9	4.37 9	3.93 9	3.64 9	3.28 9	3.91 9	3.79 9	4.08 9	3.79 9	4.23 9	3.98 9	1.62 9	4.54 9
SGM [118], H	6.48 10	6.86 10	1.64 10	11.3 11	6.76 10	6.76 10	5.97 10	6.12 10	6.89 10	6.96 10	6.23 10	6.10 10	6.22 10	6.30 10	2.82 10	13.9 11
LPS [163], H	9.35 11	10.2 11	2.69 11	10.0 10	7.97 11	8.42 11	13.6 12	10.4 11	9.17 11	12.0 11	10.4 11	10.1 11	12.4 11	12.6 12	3.43 11	12.3 10
SGBM1 [164], F	14.3 12	13.9 12	2.93 12	29.6 12	16.5 12	16.2 12	12.4 11	12.3 12	14.2 12	14.5 12	12.3 12	12.6 12	12.9 12	11.0 11	5.73 12	33.6 13
LPS [163], F	35.7 13	29.3 13	8.39 13	147 14	26.7 13	27.5 13	29.6 13	27.6 13	27.8 13	30.8 13	27.6 13	27.5 13	29.7 13	32.4 13	17.7 13	30.7 12
SGM [118], F	52.3 14	55.4 14	13.3 14	97.1 13	53.3 14	53.1 14	46.9 14	48.5 14	55.5 14	56.5 14	49.2 14	47.4 14	49.4 14	45.5 14	21.2 14	123 14
BSM [159], Q	196 15	214 15	52.7 15	312 15	211 15	213 15	187 15	188 15	214 15	210 15	192 15	192 15	195 15	199 15	93.2 15	345 15
LAMC [162], H	520 16	519 16	158 16	1030. 16	493 16	495 16	477 16	542 16	573 16	567 16	486 16	482 16	481 16	578 16	217 16	961 16

[1] Resolutions at which the methods were evaluated follow their acronyms: Q - quarter-size, H - half-size, F - full-size.

Table 5.25: Matching times (in seconds) recorded on the Middlebury 3.0 test set.

Method	Avg. time	0.5 Austr	1 AustrP	1 Bicyc2	1 Class	0.5 ClassE	1 Compu	1 Crusa	1 CrusaP	1 Djemb	0.5 DjembL	0.5 Hoops	1 Livgrm	1 Nkuba	1 Plants	0.5 Stairs
SGBM1 [164], Q	**0.23** 1	**0.18** 1	**0.18** 1	**0.16** 1	**0.32** 1	**0.32** 1	**0.04** 1	**0.36** 1	**0.38** 1	**0.18** 1	**0.18** 1	**0.26** 1	**0.19** 1	**0.27** 1	**0.18** 1	**0.24** 1
SGBM2 [164], Q	0.38 2	0.29 2	0.29 2	0.26 2	0.56 2	0.55 2	0.07 2	0.64 2	0.64 2	0.29 2	0.29 2	0.39 2	0.32 2	0.46 2	0.3 2	0.42 2
IDR, H	0.49 3	0.35 3	0.33 3	0.28 3	0.69 3	0.7 3	0.09 3	0.9 3	0.9 3	0.35 3	0.35 3	0.48 3	0.38 3	0.63 3	0.38 3	0.49 3
ELAS [161], H	0.79 4	0.99 4	0.82 4	0.80 4	0.87 4	0.71 4	0.20 4	0.93 4	0.95 4	0.82 4	0.80 4	0.85 4	0.79 4	0.83 4	0.88 4	0.74 4
SNCC [165], H	1.38 5	1.03 5	0.97 5	0.95 5	1.78 5	1.75 5	0.25 5	2.14 5	2.23 5	1.09 5	1.14 5	1.37 5	1.71 8	1.70 5	1.08 5	1.31 5
SGM [118], Q	1.48 6	1.23 6	1.21 6	1.09 6	2.00 6	2.03 6	0.25 6	2.32 6	2.29 6	1.21 6	1.27 6	1.52 6	1.32 5	1.82 6	1.26 6	1.49 6
Cens5 [160], H	1.82 7	1.45 7	1.42 7	1.19 8	2.60 7	2.62 7	0.3 8	3.02 7	3.04 7	1.43 7	1.45 7	1.79 7	1.51 6	2.24 7	1.42 7	1.83 7
SGBM1 [164], H	2.27 8	1.52 8	1.58 8	1.15 7	3.89 8	3.80 9	0.27 7	4.00 8	4.05 8	1.54 8	1.51 8	1.99 8	1.60 7	3.30 8	1.50 8	2.12 8
ELAS [161], F	3.94 9	4.53 9	4.52 9	4.11 9	4.37 9	3.72 8	0.88 9	4.55 9	4.60 9	4.11 9	3.93 9	4.16 9	4.05 9	4.07 9	3.99 9	3.70 9
LPS [163], H	9.52 10	11.3 11	8.31 11	8.84 11	11.3 10	11.6 10	3.37 12	9.54 10	9.25 10	8.61 11	11.0 11	14.3 11	9.33 11	8.82 10	11.7 11	11.7 11
SGM [118], H	9.90 11	7.93 10	7.71 10	6.69 10	13.6 11	13.9 11	1.77 10	15.8 11	15.8 11	8.06 10	8.47 10	10.4 10	8.57 10	12.2 11	8.15 10	10.0 10
SGBM1 [164], F	20.0 12	13.7 12	13.8 12	12.8 12	26.6 13	26.2 13	2.70 11	34.3 13	34.7 13	17.1 12	16.7 12	20.5 12	17.6 12	26.2 13	16.7 12	18.2 12
LPS [163], F	25.8 13	33.6 13	24.2 13	24.6 13	24.0 12	23.7 12	16.7 14	26.7 12	27.1 12	27.5 13	24.9 13	33.6 13	25.0 13	23.9 12	32.6 13	25.6 13
SGM [118], F	72.1 14	56.5 14	52.6 14	45.7 14	91.4 14	105 14	11.8 13	126 14	125 14	55.8 14	59.3 14	73.2 14	60.2 14	91.7 14	58.5 14	71.8 14
BSM [159], Q	244 15	213 15	217 15	190 15	318 15	315 15	49.6 15	353 15	357 15	221 15	221 15	253 15	227 15	288 15	209 15	236 15
LAMC [162], H	704 16	580 16	501 16	512 16	865 16	1010. 16	142 16	1200. 16	1190. 16	478 16	617 16	782 16	613 16	794 16	615 16	779 16

[1] Resolutions at which the methods were evaluated follow their acronyms: Q - quarter-size, H - half-size, F - full-size.

layer manner by alternating calls to the vertical and horizontal filtering kernels. Note, however, that multiple invocations of the filtering kernels result in increased number of global memory reads due repetitive reloading of image data necessary to guide the filter, which is the reason for the observed time overhead. Assuming that the amount of global memory on the graphics card was sufficient to perform cost aggregation in the usual way, i.e., using two cost volumes, the projected matching times are 0.35s for `Jadeplant`, 0.52s for `Vintage`, 0.64s for `Classroom*`, 0.54s for `Crusade*`, and 0.4s for `Newkuba`, which translates to an average time of 0.29s on the training set, and 0.43s on the test set. In applications where the computational efficiency is critical, additional improvements in matching times can be obtained by reducing the filter radius in the cost aggregation step from 32 to 16 pixels. This modification results in an average matching time of 0.21s per dataset (30% improvement in computational efficiency) and leads to an average `bad2.0` error of 17.4% (%3.57 decrease in accuracy with reference to the variant with larger filter radius), when evaluated on the half-size training set.

After the official submission of the results to the benchmark's website, a number of updates to the method were proposed that allowed for further reduction in error rates. To mitigate the detrimental effects of imperfect image rectification, a more robust cost metric was obtained by integrating the information from adjacent scanlines. In particular, improvements in accuracy of matching were observed in the case when the costs were computed by choosing a minimum per-pixel dissimilarity metric associated with every triplet of vertically connected pixels, each of which includes the pixel of interest and its two immediate neighbors located above and below. Note that, when

costs are computed by considering such triplets of pixels, the vertical component of the gradient difference present in the formulation of the per-pixel dissimilarity metric provides largely redundant information and, in some cases, may result in increased error rates. For this reason, it is advised that the gradient differences be computed using only the x-components of the gradients at the pixels under consideration.

It was later observed that enhancing the two-pass bilateral filter with disparity gradient information may, in certain cases, result in incorrect disparity estimates in the refinement stage. Specifically, if the disparity predictions produced by the horizontal pass contain erroneous values, so will the gradient estimates in the horizontal pass, which contributes to amplification of errors. A solution is to perform refinement in a hybrid way, i.e., using the disparity gradient information in the vertical pass and then simply adaptively averaging the estimates in the horizontal pass. In contrast to the refinement operation that does not employ gradient-based prediction, including center pixels when computing expected disparity values is advantageous, as these contain disparity evidence required to calculate the disparity gradients at the neighboring pixels. Using the previously described cost computation scheme in combination with 6 iterations of hybrid disparity refinement, the average default error metric (`bad2.0`) achieved on the training datasets dropped to 15.4%, which corresponds to an 8.33% increase in accuracy compared to the version published online. The most significant improvement was seen in the case of the `Playtable` dataset (an instance of imperfect rectification), where the error rate decreased from 49.7% to 39.3%.

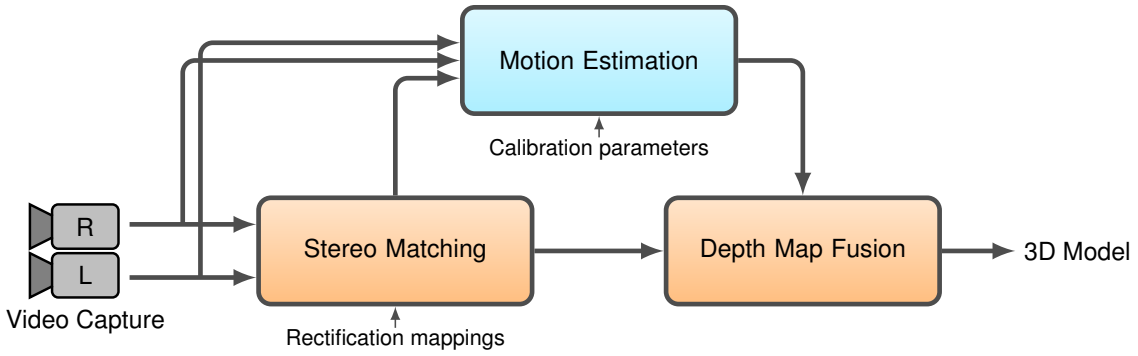


Figure 5.12: Structure reconstruction pipeline

5.4 Application in Structure Reconstruction

To demonstrate the applicability of IDR in the problem of structure reconstruction, a reconstruction process shown in Figure 5.12 is implemented that employs the proposed method. The reconstruction pipeline contains three functional blocks: the stereo matching block, the motion estimation block, and the depth map fusion block; successive frames acquired using a calibrated stereo rig are fed into the stereo matching and motion estimation blocks. In the following, the functional blocks of the reconstruction pipeline are discussed in detail and geometric models (in the form of point clouds) resulting from structure reconstruction are provided for visual assessment.

The stereo matching block contains the proposed method that was configured to reject inconsistent matches. While inconsistencies in the disparity maps, which are predominantly caused by object occlusions, can be overwritten using the disparity fill operation, it is not recommended to include disparities generated using the fill operation in the process of reconstruction, as these are only informed guesses regarding the true

disparity and are likely to introduce errors. The stereo matching block performs stereo image rectification by resampling the images according to a precomputed coordinate mapping, such that the epipolar lines become coincident with the scanlines in both images. As discussed in section 2.5, this mapping is constructed to simultaneously eliminate image distortion caused by the camera lenses.

The motion estimation block, which accepts a sequence of stereo frames and the corresponding disparity maps as inputs, implements a variant of the bundle adjustment process [166] designed to recover the motion parameters of the stereo rig traveling through the scene. The scene is assumed to be static. To obtain initial estimates of the motion parameters, feature detection and tracking is first performed on the incoming frames. Using the provided disparity maps and the disparity-to-depth mapping calculated at the time of calibration of the stereo rig, a set of correspondences between world points is calculated from the set of feature matches. The point registration technique given in [167] is then used to compute a sequence of rigid transformations characterizing the motion of the cameras between subsequent frames. In order to identify and reject the correspondences associated with mismatched features, the registration is iterated within the random sampling consensus framework [168].

Since depth information is available, the motion parameters, i.e., the rigid transformations obtained through point registration, are appropriately scaled. Note that, if no depth information was provided, estimation of motion parameters would still be possible from the fundamental matrices. In that event, however, the rigid transformation can only be recovered up to an unknown scale. Once known, the motion parameters of both cameras and the world points used in registration are jointly refined

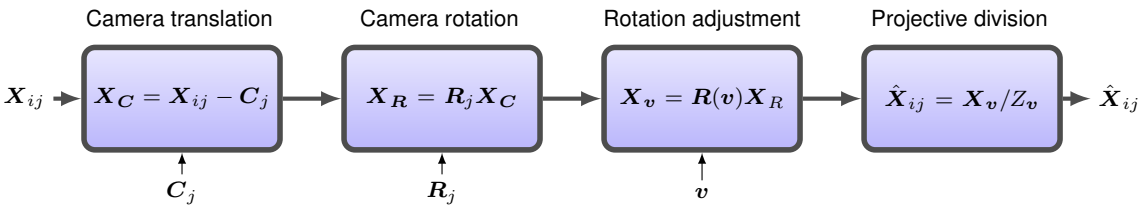


Figure 5.13: Decomposition of the camera projection process for motion estimation. The vector $\boldsymbol{X}_i$ denotes the i-th world point and $\hat{\boldsymbol{X}}_{ij}$ denotes its projection in the view of the j-th camera. The translation vector $\boldsymbol{C}_j$ and the rotation matrix $\boldsymbol{R}_j$ encode the position and the orientation of the camera; $\boldsymbol{v}$ and $\boldsymbol{R}(\boldsymbol{v})$ are the minimal rotation vector and the corresponding rotation matrix. Vectors $\boldsymbol{X}_C$, $\boldsymbol{X}_R$, $\boldsymbol{X}_v$ are the intermediate results of applying camera translation, camera rotation, and rotation adjustment operations to $\boldsymbol{X}_i$; Z_v is the z-ccordinate of $\boldsymbol{X}_v$.

by minimizing the reprojection errors in all frames using the Levenberg-Marquardt algorithm. The projective action of the camera on the world points is decomposed as shown in Figure 5.13, making it possible to adjust the orientation of the camera by applying an incremental rotation matrix in each iteration of the minimization. As previously discussed in section 2.3.2, it is convenient to parametrize the incremental rotation matrix by a minimal rotation vector. For small adjustments, the derivative of the reprojection error with respect to the minimal rotation vector takes a simple linear form (for details, see for example [169] p. 43).

Finally, the camera motion parameters are used in the depth map fusion block to merge individual disparity/depth maps. The fusion operation renders a sequence of disparity maps onto the image plane of a chosen reference view. This is done by converting image coordinates and disparity values to points in the world's coordinate

frame, transforming the resulting points into the coordinate system of the reference view, and then inverting the disparity-to-depth mapping to obtain projections of the world points in the reference view. While there exist sophisticated techniques that combine multiple depth hypotheses at each pixel of the reference view by considering the visibility of image features [170–172], a simple technique is used here that averages the depth hypotheses, simultaneously rejecting those that significantly deviate from the mean depth at each pixel.

An offline approach to structure reconstruction is taken, where the frames captured using the stereo camera setup are first written to video files and then processed in batch. Reconstruction is demonstrated using two distinct video sequences. The `Playroom` sequence [173], for which the ground truth disparities are known, depicts a synthetic, i.e., computer-generated, scene of a playroom. This scene was rendered into the views of a moving ideal stereo camera setup, resulting in distortionless video frames that can be matched without prior rectification. The artificial cameras operated at the resolution of 640×480 pixels, at which the common focal length was 761 pixels, corresponding to a 45.5° horizontal field of view. Displaced by 88.9mm, the cameras produced disparities of up to 60 pixels. The `Warrior` sequence, named after the statue of a Terracotta Army warrior present in the scene, was acquired at the laboratory of the Perceptual Systems Research Group at the UNL, in 2014, using a handheld stereo setup of PointGrey Research Flea3 FL3-U3-13S2C-CS cameras. At the resolution of 640×480 pixels, the focal length of the cameras after stereo image rectification was determined to be 425 pixels (a horizontal field of view of 85°). With a 61.86mm offset between the cameras, the disparities do not exceed 80 pixels. Ground truth disparity

maps are not available for the `Warrior` sequence.

Camera trajectories extracted from both video sequences are shown in Figure 5.14 along with approximate models of the underlying scene structure. Results of structure reconstruction on the `Playroom` sequence are given in Figure 5.15. Disparity estimates from frames 20 - 70 were integrated to obtain a merged disparity map and a corresponding model of the scene in the form of a dense point cloud. Frame 45 was chosen as a reference point; the ground truth disparity map at the reference frame and the point cloud computed using the ground truth data are provided for comparison. Point clouds recovered from selected frames of the `Warrior` sequence are presented in Figure 5.16. Each point cloud was generated by fusing disparities within a bundle of 11 frames centered in time at the frame of interest. In Figure 5.17, these models are placed in a common coordinate frame to obtain a global model of the scene captured in the `Warrior` sequence; flat-colored point clouds are additionally shown in the figure to illustrate the alignment of the partial models[1].

Whereas the evaluation using the Middlebury benchmark proved that the proposed stereo matching method achieves high accuracy on still frames, the experiments with structure reconstruction from video sequences reveal that the method benefits from integrating disparity information from multiple views. Paired with motion estimation and depth map fusion techniques, the proposed method is capable of acquiring coherent models of the scene that extend beyond the viewing frustum of a single frame. In particular, the depth map fusion operation provides occlusion handling capabilities,

[1]Motion estimation and depth map fusion software was provided by Dr. Eric Psota, Research Assistant Professor at the Department of Electrical Engineering, University of Nebraska-Lincoln.

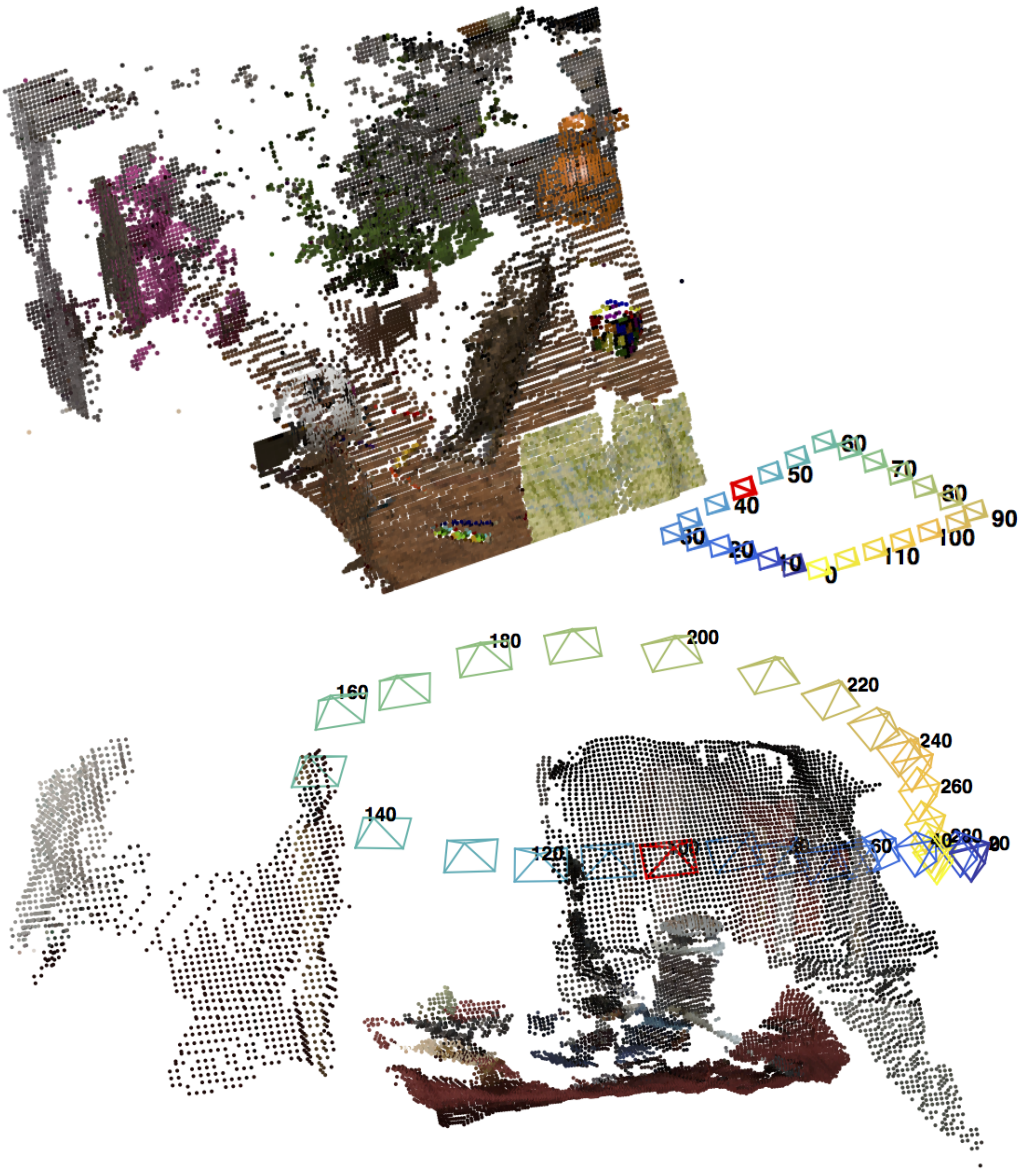
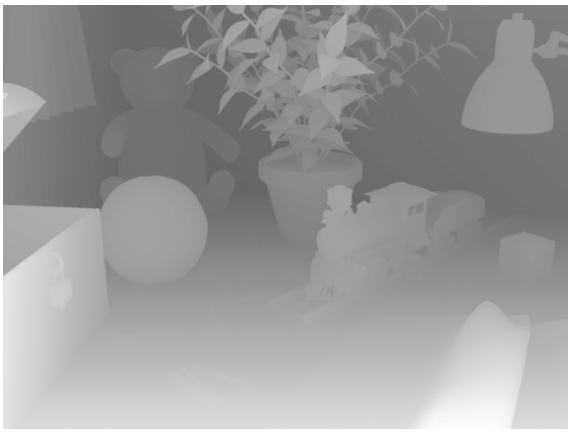
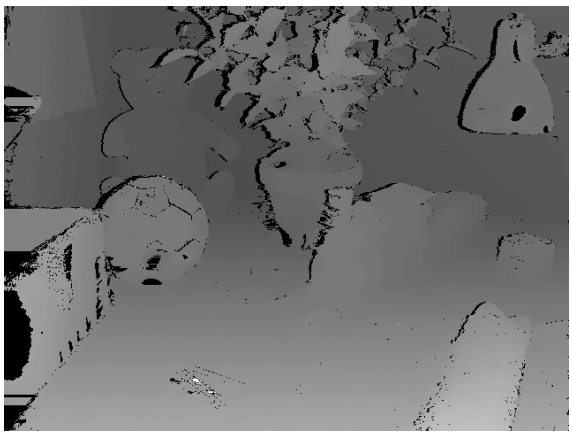


Figure 5.14: Camera trajectories extracted from the `Playroom` (top) and `Warrior` sequences. Views equally spaced in time are shown with the corresponding frame numbers. Approximate models of the scenes were generated by mapping the pixel coordinates and the disparities sampled at regularly spaced points within the reference views (highlighted in red) to world coordinates.

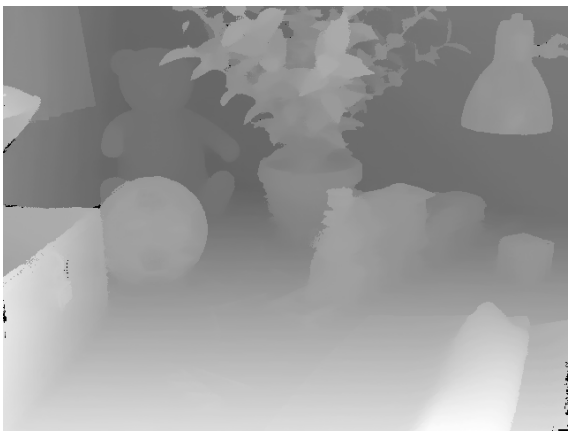
(a) Reference view (frame 45)



(b) True disparity



(c) Disparity at frame 45



(d) Merged disparity (frame 45 ± 25 frames)

(e) Point cloud using true disparity

(f) Point cloud using merged disparity

Figure 5.15: Results of structure reconstruction using the `Playroom` sequence

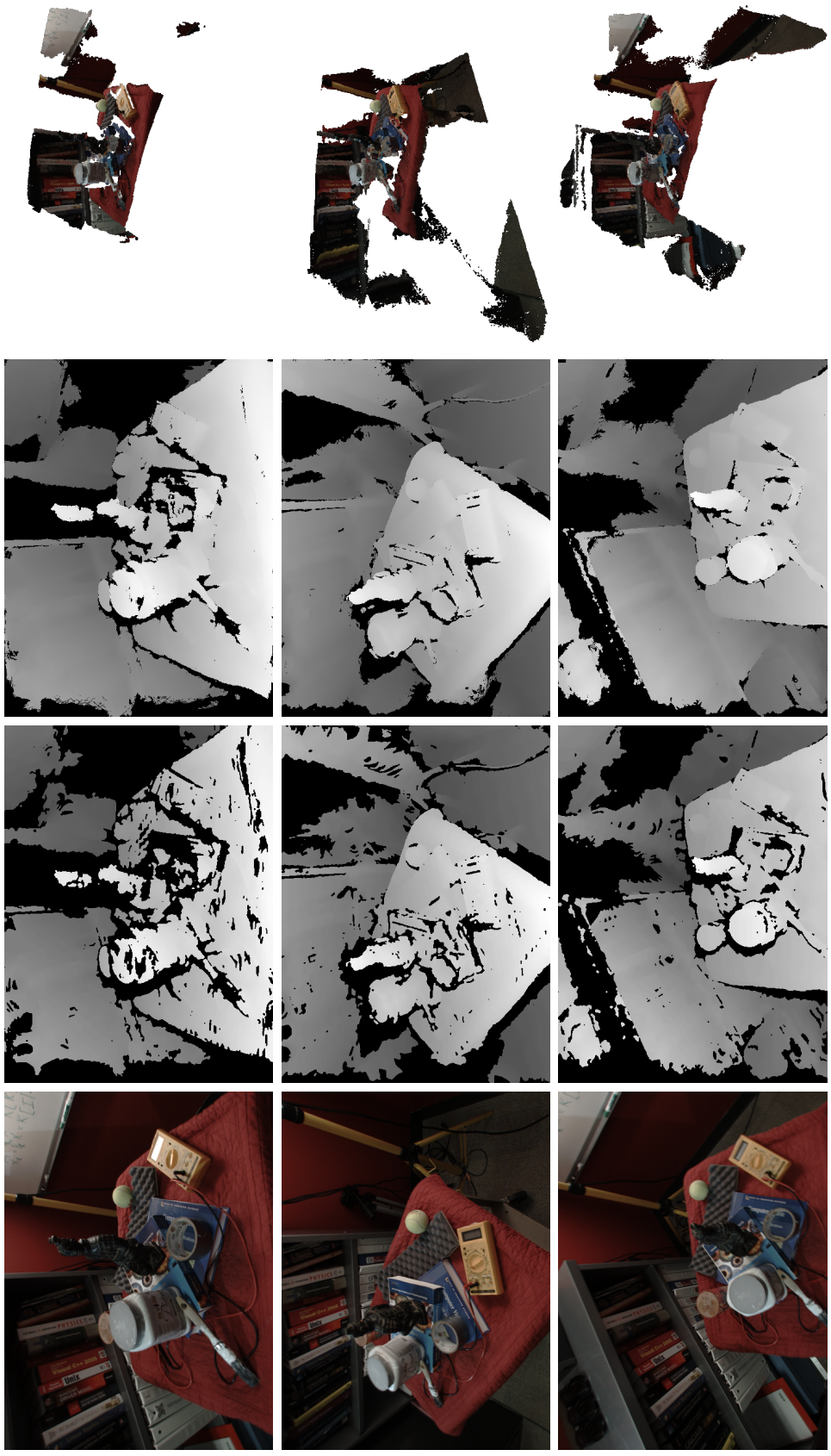


Figure 5.16: Disparity maps and point clouds generated using selected frames of the `Warrior` sequence.

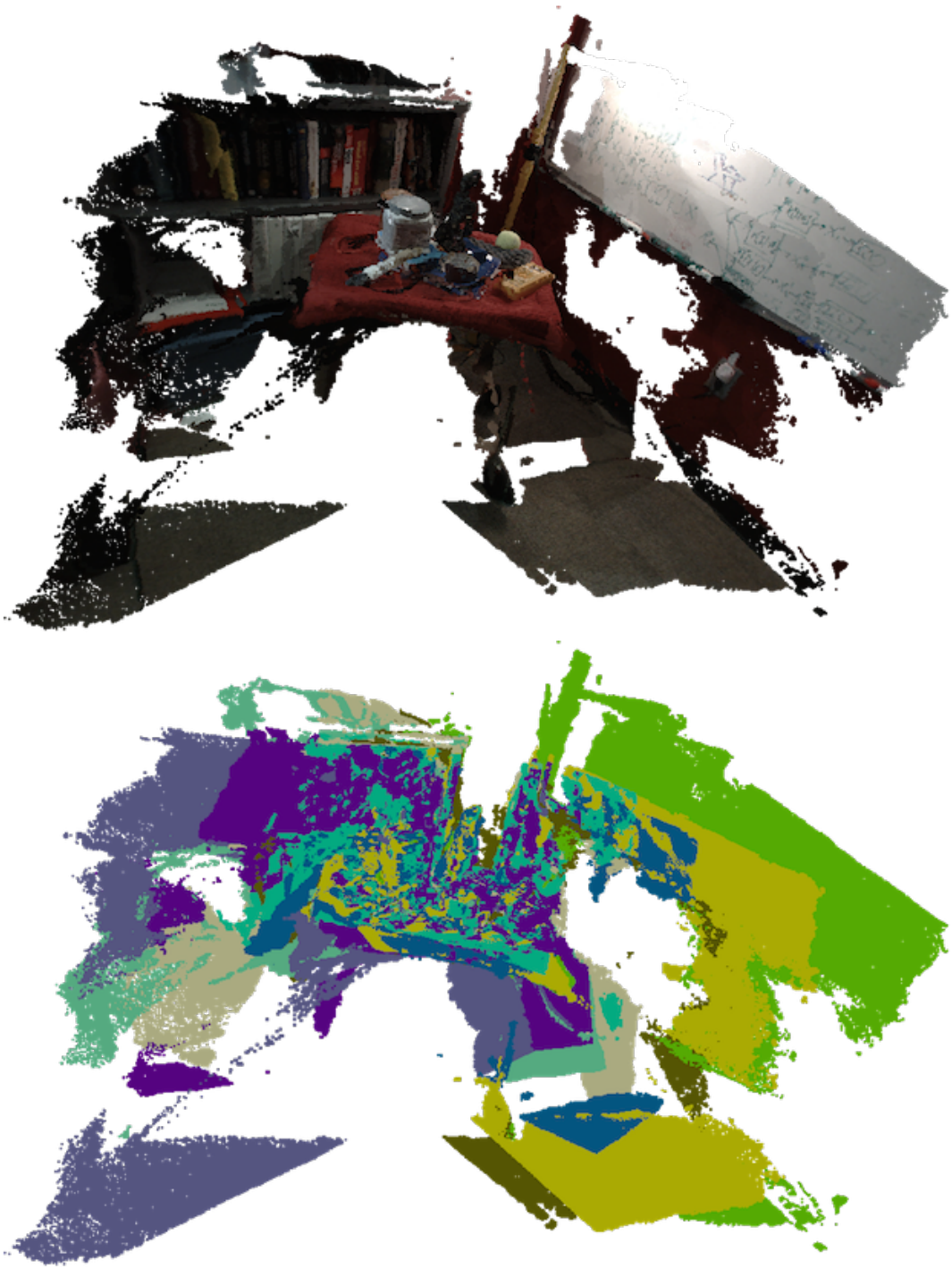


Figure 5.17: A global model of the scene in the `Warrior` sequence.

contributes to an increase in density of matches by generating valid disparity values for unfilled pixels, and opens a possibility to cross-check depth hypotheses using the visual evidence from multiple views. With the availability of real-time motion estimation techniques resulting from the recent advances in simultaneous localization and mapping (SLAM) [174], the proposed method is a suitable choice for applications that require efficient, accurate, and robust processing of stereo videos.

CHAPTER 6

Conclusion

Since the publication of Lucas and Kanade's influential paper [11], which marked the beginning of the field of visual correspondence, stereo matching remains one of the most actively studied problems in computer vision. A plethora of stereo matching methods have been proposed in the past three decades and this dynamism of developments in stereo matching is still seen today. However, very few of these methods are able to operate at frame rates required by practical applications due to the prohibitively high computational complexity of the matching process.

Recently, several methods were developed and implemented on graphics hardware normally used for computer gaming that claim near real-time or real-time operation [100, 102, 103, 105, 136, 143]. The use of graphics hardware allows for the matching to be performed in a massively parallel way, leading to drastic speedups when compared to sequential implementations on conventional general purpose processors. To fulfill memory access requirements of the graphics hardware and fully utilize its computational potential, these efficient methods often employ simplified operations,

trading accuracy for speed.

In the previous chapters, a new method was presented that enables real-time stereo matching with high accuracy. This method is constructed upon a cost filtering framework and uses a two-pass approximation of the well-known bilateral filter to perform adaptive aggregation of matching costs efficiently. Furthermore, the proposed method introduced a novel refinement technique that iteratively applies the said two-pass bilateral filter and probabilistic reasoning in order to identify and overwrite erroneous disparities. Through penalization of low-confidence matches and reassignment of disparities, the refinement technique significantly improves the accuracy of the initial solution. The refinement technique can be paired with any local method that maintains a cost volume to enhance disparity maps. Lastly, this method relies on a cost formulation that incorporates a weighted sum of pixel dissimilarity metrics computed using the gradients and census transforms at the pixels under consideration. Such a formulation of the cost metric guarantees robust matching, even if the input images are far from being photometrically consistent. The parameters of the proposed method have been adjusted through particle swarm optimization to achieve better accuracy.

Using the latest datasets and the evaluation methodology delivered by the widely accepted Middlebury stereo performance benchmark, the proposed method has been extensively evaluated and determined to provide high accuracy of matching. Specifically, the method was shown to be the top performer when evaluating the percentage of pixels with the disparity error greater than 2 (at the nominal image resolution). When other accuracy metrics were evaluated, including the percentages of incorrectly

assigned disparities with different error thresholds, the average absolute disparity error, and the root-mean-square error, the proposed method was among the top five and, more often, among the top three methods listed on the Middlebury benchmark at the time of evaluation. The use of the cost metric based on both gradient information and census transforms – in combination with the two-pass bilateral filter in cost aggregation – results in accurate initial solutions with well-defined object boundaries. The method is capable of capturing fine details in the disparity maps, even in areas with minimal texture details and minuscule color transitions. In addition, the results have shown that the two-component cost metric allows the method to maintain its accuracy in the presence of radiometric distortions, particularly under inconsistent camera exposures.

Subsequent accuracy improvements are achieved through iterative application of the refinement procedure to the initial solution. While the parameters of the bilateral filter used in cost aggregation enforce alignment of disparity discontinuities with the image edges, the filter in the refinement stage is configured to enforce local consistency of disparities. The use of larger filter radius in the refinement also aids in resolving ambiguous matches. Finally, incorporating gradient information into the refinement operation was shown to provide sub-pixel accurate disparities that transition smoothly along slanted surfaces, reducing the staircase effects in the disparity maps.

The proposed method has been designed to allow computations at individual pixels to be performed independent of other pixels whenever possible, simultaneously supporting sharing of data among neighboring pixels to avoid excessive reads from the memory of the graphics hardware. Profiling data indicate that the method

evaluates approximately 542 million disparity estimates per second when executed on the NVIDIA GeForce TITAN Black graphics card, which is equivalent to matching 640×480 images with a frame rate of 30FPS. Note, however, that this was possible by fixing the window size of the the bilateral filter in both the cost aggregation and disparity refinement stages at 33×33 and 67×67 pixels, respectively. Also note that processing high-resolution images may require larger windows.

The key weakness of the proposed method is its limited scalability. Even with fixed filter windows, both the computational complexity and memory requirements of the method are linear in the product of image dimensions and the number of disparities. Since the number of disparities is typically proportional to the dimensions of the images, doubling the images size will likely result in cubed memory requirements and execution times. In the view of the recent availability of ultra-high-definition image sensors and an ongoing trend towards increasing their resolution beyond today's standards, the issue of scalability remains an open challenge in stereo matching.

Addressing the issue of scalability is suggested as a direction for future work. One way to do so is to first perform superpixel segmentation on the input images, match the resulting superpixel representations, and then compute fine-scale updates to the initially obtained disparities to arrive at a full-resolution disparity map. By using superpixel segmentation, the size of the images can be reduced from millions of pixels to hundreds or thousands of superpixels. Matching of superpixels, however, will likely lack the computational regularity of matching arrays of ordinary pixels, which is quintessential for accelerating computations using graphics hardware. It is also unclear whether the existing superpixel segmentation algorithms, when applied

independently to both input images, are able to produce superpixel representations that can be reliably matched. Further investigation is needed to answer if superpixel segmentation and initial matching can be performed jointly, such that superpixels are constructed in a way that maximizes both the local similarity of intensities and photometric consistency between the two images. Integrating temporal disparity evidence into matching, yet without relying on motion estimation or assuming rigidity of the scene, is another possibility for future work. While temporal stereo matching has previously been approached, for instance, in [136, 175, 176], the body of work on this problem remains surprisingly small.

The proposed method expands stereo matching to reflect a more sophisticated model of human vision, and was shown to allow for highly accurate matching. Offering high accuracy of matching and real-time operation, the proposed method is expected to find applications in robotic navigation and 3D video surveillance, and could be a component of high-performance structure reconstruction algorithms. Remarkably, this high accuracy is achieved without requiring any contextual knowledge of the scene or performing plane fitting. The author believes, however, that the future of stereo matching lies in the integration of additional disparity evidence into the process of matching. Such disparity evidence can be derived from supplementary sensory data or through analysis of the scene using machine learning. It is noteworthy that, unlike many methods, the design of the proposed method allows for additional disparity evidence to be included into matching. Supported by auxiliary sensors and enhanced with machine learning, stereo matching has the potential to provide camera-equipped machines with the ability to fully comprehend the world around them.

Bibliography

[1] C. Wheatstone, "Contributions to the physiology of vision. part the first. on some remarkable, and hitherto unobserved, phenomena of binocular vision," *Philosophical Transactions of the Royal Society of London*, vol. 128, pp. 371–394, 1838.

[2] T. Shipley, "The first random-dot texture stereogram," *Vision Research*, vol. 11, no. 12, pp. 1491 – IN2, 1971.

[3] B. Julesz and J. E. Miller, "Automatic stereoscopic presentation of functions of two variables," *Bell System Technical Journal, The*, vol. 41, pp. 663–676, March 1962.

[4] B. Julesz, *Foundations of Cyclopean Perception*. University of Chicago Press, 1971.

[5] B. Julesz, "Towards the automation of binocular depth perception (automap)," in *Proceedings of the IFIPS Congres*, 1962.

[6] J. I. Nelson, "Globality and stereoscopic fusion in binocular vision," *Journal of Theoretical Biology*, vol. 49, no. 1, pp. 1 – 88, 1975.

[7] B. Julesz and J. Chang, "Interaction between pools of binocular disparity detectors tuned to different disparities," *Biological Cybernetics*, vol. 22, no. 2, pp. 107–119, 1976.

[8] D. Marr and T. Poggio, "Cooperative computation of stereo disparity," tech. rep., Cambridge, MA, USA, 1976.

[9] D. Marr and T. Poggio, "A computational theory of human stereo vision," *Proceedings of the Royal Society of London. Series B, Biological Sciences*, vol. 204, no. 1156, pp. 301–328, 1979.

[10] W. E. L. Grimson, "A computer implementation of a theory of human stereo vision," *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, vol. 292, no. 1058, pp. 217–253, 1981.

[11] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, IJCAI'81, (San Francisco, CA, USA), pp. 674–679, Morgan Kaufmann Publishers Inc., 1981.

[12] D. Scharstein, R. Szeliski, and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," pp. 131 –140, 2001.

[13] M. Gong, R. Yang, L. Wang, and M. Gong, "A performance study on different cost aggregation approaches used in real-time stereo matching," *International Journal of Computer Vision*, vol. 75, pp. 283–296, 2007.

[14] F. Tombari, S. Mattoccia, L. Di Stefano, and E. Addimanda, "Classification and evaluation of cost aggregation methods for stereo correspondence," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, June 2008.

[15] C. Loop and Z. Zhang, "Computing rectifying homographies for stereo vision," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 1, pp. –131 Vol. 1, 1999.

[16] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for markov random fields

with smoothness-based priors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, pp. 1068–1080, June 2008.

[17] K.-J. Yoon and I.-S. Kweon, "Locally adaptive support-weight approach for visual correspondence search," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 924 – 931 vol. 2, Jun. 2005.

[18] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. New York, NY, USA: Cambridge University Press, 2 ed., 2003.

[19] A. Fitzgibbon, "Simultaneous linear estimation of multiple view geometry and lens distortion," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–125–I–132 vol.1, 2001.

[20] C. Brauer-Burchardt and K. Voss, "A new algorithm to correct fish-eye- and strong wide-angle-lens-distortion from single images," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, vol. 1, pp. 225–228 vol.1, 2001.

[21] J. Heikkila and O. Silven, "Calibration procedure for short focal length off-the-shelf ccd cameras," in *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, vol. 1, pp. 166–170 vol.1, Aug 1996.

[22] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pp. 1106–1112, Jun 1997.

[23] J. Heikkila, "Geometric camera calibration using circular control points," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 1066–1077, Oct 2000.

[24] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, pp. 666–673 vol.1, 1999.

[25] A. E. Conrady, "Decentered-lens systems," *Monthly notices of the Royal Astronomical Society*, vol. 79, pp. 384–390, 1919.

[26] D. C. Brown, "Decentering distortion of lenses," *Photogrammetric Engineering*, vol. 32, no. 3, pp. 444–462, 1966.

[27] Z. Zhang, "Camera calibration with one-dimensional objects," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, pp. 892–899, July 2004.

[28] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *Robotics and Automation, IEEE Journal of*, vol. 3, pp. 323–344, August 1987.

[29] P. Sturm and S. Maybank, "On plane-based camera calibration: A general algorithm, singularities, applications," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 1, pp. –437 Vol. 1, 1999.

[30] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 1330–1334, Nov 2000.

[31] S. J. Maybank and O. D. Faugeras, "A theory of self-calibration of a moving camera," *Int. J. Comput. Vision*, vol. 8, pp. 123–151, Aug. 1992.

[32] R. Hartley, "An algorithm for self calibration from several views," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pp. 908–912, Jun 1994.

[33] Q.-T. Luong and O. D. Faugeras, "Self-calibration of a moving camera from pointcorrespondences and fundamental matrices," *Int. J. Comput. Vision*, vol. 22, pp. 261–289, Mar. 1997.

[34] G. Medioni and S. B. Kang, *Emerging Topics in Computer Vision.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.

[35] G. Xu and Z. Zhang, *Epipolar Geometry in Stereo, Motion, and Object Recognition: A Unified Approach*. Norwell, MA, USA: Kluwer Academic Publishers, 1996.

[36] R. I. Hartley, "Theory and practice of projective rectification," *Int. J. Comput. Vision*, vol. 35, pp. 115–127, Nov. 1999.

[37] M. Pollefeys, R. Koch, and L. Van Gool, "A simple and efficient rectification method for general motion," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, pp. 496–501 vol.1, 1999.

[38] A. Fusiello, E. Trucco, and A. Verri, "A compact algorithm for rectification of stereo pairs," *Mach. Vision Appl.*, vol. 12, pp. 16–22, July 2000.

[39] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, pp. 346–359, June 2008.

[40] P. Anandan, "A computational framework and an algorithm for the measurement of visual motion," *International Journal of Computer Vision*, vol. 2, no. 3, pp. 283–310, 1989.

[41] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pp. 593–600, Jun 1994.

[42] C. Harris and M. Stephens, "A combined corner and edge detector," in *In Proc. of Fourth Alvey Vision Conference*, pp. 147–151, 1988.

[43] B. Triggs, "Detecting keypoints with stable position, orientation, and scale under illumination changes," in *Computer Vision - ECCV 2004*, vol. 3024 of *Lecture Notes in Computer Science*, pp. 100–113, Springer Berlin Heidelberg, 2004.

[44] M. Brown, R. Szeliski, and S. Winder, "Multi-image matching using multi-scale oriented patches," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 510–517 vol. 1, June 2005.

[45] A. Guiducci, "Corner characterization by differential geometry techniques," *Pattern Recogn. Lett.*, vol. 8, pp. 311–318, Dec. 1988.

[46] S. M. Smith and J. M. Brady, "Susan: A new approach to low level image processing," *Int. J. Comput. Vision*, vol. 23, pp. 45–78, May 1997.

[47] M. Trajkovic and M. Hedley, "Fast corner detection," *Image and Vision Computing*, vol. 16, no. 2, pp. 75 – 87, 1998.

[48] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, vol. 1, pp. 430–443, May 2006.

[49] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, pp. 105–119, 2010.

[50] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *Int. J. Comput. Vision*, vol. 60, pp. 63–86, Oct. 2004.

[51] A. P. Witkin, "Scale-space filtering," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'83, (San Francisco, CA, USA), pp. 1019–1022, Morgan Kaufmann Publishers Inc., 1983.

[52] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, pp. 91–110, Nov. 2004.

[53] T. Lindeberg, "Feature detection with automatic scale selection," *Int. J. Comput. Vision*, vol. 30, pp. 79–116, Nov. 1998.

[54] P. Viola and M. Jones, "Robust real-time object detection," in *International Journal of Computer Vision*, 2001.

[55] M. Brown and D. Lowe, "Invariant features from interest point groups," in *In British Machine Vision Conference*, pp. 656–665, 2002.

[56] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, pp. 1615–1630, Oct 2005.

[57] J. J. Koenderink and A. J. van Doom, "Representation of local geometry in the visual system," *Biol. Cybern.*, vol. 55, pp. 367–375, Mar. 1987.

[58] W. Freeman and E. Adelson, "The design and use of steerable filters," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 13, pp. 891–906, Sep 1991.

[59] L. J. V. Gool, T. Moons, and D. Ungureanu, "Affine/ photometric invariants for planar intensity patterns," in *Proceedings of the 4th European Conference on Computer Vision-Volume I - Volume I*, ECCV '96, (London, UK, UK), pp. 642–651, Springer-Verlag, 1996.

[60] F. Schaffalitzky and A. Zisserman, "Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?"," in *Proceedings of the 7th European Conference on Computer Vision-Part I*, ECCV '02, (London, UK, UK), pp. 414–431, Springer-Verlag, 2002.

[61] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 509–522, Apr 2002.

[62] S. Lazebnik, C. Schmid, and J. Ponce, "A sparse texture representation using affine-invariant regions," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2, pp. II–319–II–324 vol.2, June 2003.

[63] A. Baumberg, "Reliable feature matching across widely separated views," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 1, pp. 774–781 vol.1, 2000.

[64] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool, "A comparison of affine region detectors," *Int. J. Comput. Vision*, vol. 65, pp. 43–72, Nov. 2005.

[65] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Found. Trends. Comput. Graph. Vis.*, vol. 3, pp. 177–280, July 2008.

[66] V. Lepetit and P. Fua, "Keypoint recognition using randomized trees," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, pp. 1465–1479, Sept 2006.

[67] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, pp. 448–461, March 2010.

[68] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, (Berlin, Heidelberg), pp. 778–792, Springer-Verlag, 2010.

[69] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2564–2571, Nov 2011.

[70] S. Leutenegger, M. Chli, and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2548–2555, Nov 2011.

[71] A. Alahi, R. Ortiz, and P. Vandergheynst, "Freak: Fast retina keypoint," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 510–517, June 2012.

[72] Y. Ke and R. Sukthankar, "Pca-sift: a more distinctive representation for local image descriptors," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2, pp. II–506–II–513 Vol.2, June 2004.

[73] G. Hua, M. Brown, and S. Winder, "Discriminant embedding for local image descriptors," in *International Conference on Computer Vision*, October 2007.

[74] H. Hirschmuller and D. Scharstein, "Evaluation of stereo matching costs on images with radiometric differences," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, pp. 1582–1599, Sept 2009.

[75] D. Terzopoulos, "Regularization of inverse visual problems involving discontinuities," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, pp. 413–424, July 1986.

[76] M. J. Black and A. Rangarajan, "On the unification of line processes, outlier rejection, and robust statistics with applications in early vision," *Int. J. Comput. Vision*, vol. 19, pp. 57–91, July 1996.

[77] P. Fua, "A parallel stereo algorithm that produces dense depth maps and preserves image features," *Machine Vision and Applications*, vol. 6, no. 1, pp. 35–49, 1993.

[78] A. F. Bobick and S. S. Intille, "Large occlusion stereo," *Int. J. Comput. Vision*, vol. 33, pp. 181–200, Sept. 1999.

[79] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, pp. 1222 –1239, nov 2001.

[80] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-6, pp. 721–741, Nov 1984.

[81] J. Besag, "On the Statistical Analysis of Dirty Pictures," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 48, no. 3, pp. 259–302, 1986.

[82] J. Sun, N.-N. Zheng, and H.-Y. Shum, "Stereo matching using belief propagation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, pp. 787 – 800, July 2003.

[83] M. Wainwright, T. Jaakkola, and A. Willsky, "Map estimation via agreement on trees: message-passing and linear programming," *Information Theory, IEEE Transactions on*, vol. 51, pp. 3697–3717, Nov 2005.

[84] V. Lempitsky, C. Rother, and A. Blake, "Logcut - efficient graph cut optimization for markov random fields," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–8, Oct 2007.

[85] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," *Int. J. Comput. Vision*, vol. 70, pp. 41–54, Oct. 2006.

[86] Q. Yang, L. Wang, R. Yang, H. Stewenius, and D. Nister, "Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, pp. 492–504, March 2009.

[87] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, pp. 1568–1583, Oct 2006.

[88] A. Klaus, M. Sormann, and K. Karner, "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3, pp. 15–18, 2006.

[89] Q. Yang, C. Engels, and A. Akbarzadeh, "Near real-time stereo for weakly-textured scenes.," in *British Machine Vision '08*, 2008.

[90] L. Wang and R. Yang, "Global stereo matching leveraged by sparse ground control points," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 3033 –3040, june 2011.

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

[92] Y. Boykov, O. Veksler, and R. Zabih, "A variable window approach to early vision," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, pp. 1283 –1294, dec 1998.

[93] O. Veksler, "Stereo correspondence with compact windows via minimum ratio cycle," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 1654 – 1660, dec 2002.

[94] O. Veksler, "Fast variable window for stereo correspondence using integral images," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1, pp. I–556 – I–561 vol.1, june 2003.

[95] A. Fusiello, V. Roberto, and E. Trucco, "Efficient stereo with multiple windowing," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 858 –863, June 1997.

[96] K. Prazdny, "Detection of binocular disparities," *Biological Cybernetics*, vol. 52, no. 2, pp. 93–99, 1985.

[97] Y. Xu, D. Wang, T. Feng, and H.-Y. Shum, "Stereo computation using radial adaptive windows," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 3, pp. 595–598 vol.3, 2002.

[98] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister, "High-quality real-time stereo using adaptive cost aggregation and dynamic programming," in *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, pp. 798 –805, Jun. 2006.

[99] S. Mattoccia, S. Giardino, and A. Gambini, "Accurate and efficient cost aggregation strategy for stereo correspondence based on approximated joint bilateral filtering.," in *Asian Conference on Computer Vision (ACCV2009)*, pp. 371–380, September 2009.

[100] W. Yu, T. Chen, F. Franchetti, and J. Hoe, "High performance stereo vision designed for massively data parallel platforms," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, pp. 1509 –1519, nov. 2010.

[101] D. Sun, S. Roth, and M. Black, "Secrets of optical flow estimation and their principles," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2432 –2439, june 2010.

[102] K. Zhang, J. Lu, Q. Yang, G. Lafruit, R. Lauwereins, and L. Van Gool, "Real-time and accurate stereo: A scalable approach with bitwise fast voting on cuda," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2011.

[103] S. Mattoccia, M. Viti, and F. Ries, "Near real-time fast bilateral stereo on the gpu," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pp. 136 –143, June 2011.

[104] Q. Yang, "Recursive bilateral filtering," in *ECCV*, p. to appear, 2012.

[105] Q. Yang, "Hardware-efficient bilateral filtering for stereo matching," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2013.

[106] D. Scharstein and R. Szeliski, "Stereo matching with nonlinear diffusion," *International Journal of Computer Vision*, vol. 28, pp. 155–174, June 1998.

[107] C. L. Zitnick and T. Kanade, "A cooperative algorithm for stereo matching and occlusion detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 675–684, July 2000.

[108] R. Brockers, "Cooperative stereo matching with color-based adaptive local support," in *Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns*, CAIP '09, (Berlin, Heidelberg), pp. 1019–1027, Springer-Verlag, 2009.

[109] Y. Ohta and T. Kanade, "Stereo by intra- and inter-scanline search using dynamic programming," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-7, pp. 139 –154, march 1985.

[110] P. Belhumeur, "A binocular stereo algorithm for reconstructing sloping, creased, and broken

surfaces in the presence of half-occlusion," in *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pp. 431–438, May 1993.

[111] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs, "A maximum likelihood stereo algorithm," *Comput. Vis. Image Underst.*, vol. 63, pp. 542–567, May 1996.

[112] S. Birchfield and C. Tomasi, "A pixel dissimilarity measure that is insensitive to image sampling," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, pp. 401–406, Apr 1998.

[113] O. Veksler, "Stereo correspondence by dynamic programming on a tree," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 384–390 vol. 2, June 2005.

[114] J. C. Kim, K.-M. Lee, B.-T. Choi, and S.-U. Lee, "A dense stereo matching using two-pass dynamic programming with generalized ground control points," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 1075–1082 vol. 2, June 2005.

[115] M. Gong and Y.-H. Yang, "Near real-time reliable stereo matching using programmable graphics hardware," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 924–931 vol. 1, June 2005.

[116] X. Huang, "Cooperative optimization for energy minimization in computer vision: A case study of stereo matching," in *Pattern Recognition, 26th DAGM Symposium, Proceedings*, pp. 302–309, 2004.

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

[118] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, pp. 328–341, Feb 2008.

[119] Q. Yang, "A non-local cost aggregation method for stereo matching," in *CVPR*, pp. 1402–1409, 2012.

[120] J. Lu, H. Yang, D. Min, and M. Do, "Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 1854–1861, June 2013.

[121] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *ACM Trans. Graph.*, vol. 28, pp. 24:1–24:11, July 2009.

[122] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision*, vol. 92, no. 1, pp. 1–31, 2011.

[123] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani, "Hierarchical model-based motion estimation," in *Proceedings of the Second European Conference on Computer Vision*, ECCV '92, (London, UK, UK), pp. 237–252, Springer-Verlag, 1992.

[124] R. Szeliski and J. Coughlan, "Spline-based image registration," *Int. J. Comput. Vision*, vol. 22, pp. 199–218, Mar. 1997.

[125] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1-3, pp. 185 – 203, 1981.

[126] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l1 optical flow," in *Proceedings of the 29th DAGM conference on Pattern recognition*, (Berlin, Heidelberg), pp. 214–223, Springer-Verlag, 2007.

[127] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D*, vol. 60, pp. 259–268, Nov. 1992.

[128] A. Chambolle, "Total variation minimization and a class of binary mrf models," in *Proceedings of the 5th international conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, EMMCVPR'05, (Berlin, Heidelberg), pp. 136–152, Springer-Verlag, 2005.

[129] J. Rannacher, "Realtime 3d motion estimation on graphics hardware," *Undergraduate Thesis, Heidelberg University*, 2009.

[130] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand, "A gentle introduction to bilateral filtering and its applications," in *ACM SIGGRAPH 2008 Classes*, SIGGRAPH '08, (New York, NY, USA), pp. 1:1–1:50, ACM, 2008.

[131] S. B. Kang, R. Szeliski, and J. Chai, "Handling occlusions in dense multi-view stereo," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–103 – I–110 vol.1, 2001.

[132] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions using graph cuts," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2, pp. 508 –515 vol.2, 2001.

[133] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Computer Vision, 1998. Sixth International Conference on*, pp. 839 –846, jan 1998.

[134] J. Chen, S. Paris, and F. Durand, "Real-time edge-aware image processing with the bilateral grid," *ACM Trans. Graph.*, vol. 26, July 2007.

[135] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," *Int. J. Comput. Vision*, vol. 81, pp. 24–52, Jan. 2009.

[136] C. Richardt, D. Orr, I. Davies, A. Criminisi, and N. A. Dodgson, "Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid," in *Proceedings of the 11th Europen Conference on Computer Vision. ECCV 2010*, pp. 510–523, 2010.

[137] F. Porikli, "Constant time o(1) bilateral filtering," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1 –8, june 2008.

[138] E. Elboher and M. Werman, "Cosine integral images for fast spatial and range filtering," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pp. 89 –92, sept. 2011.

[139] K. Chaudhury, D. Sage, and M. Unser, "Fast o(1) bilateral filtering using trigonometric range kernels," *Image Processing, IEEE Transactions on*, vol. 20, pp. 3376 –3382, dec. 2011.

[140] K. Chaudhury, "Constant-time filtering using shiftable kernels," *Signal Processing Letters, IEEE*, vol. 18, pp. 651 –654, nov. 2011.

[141] S. Sengupta, M. Harris, and M. Garland, "Efficient parallel scan algorithms for gpus," tech. rep., NVIDIA Corporation, Dec 2008.

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

[143] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 3017 –3024, june 2011.

[144] R. Deriche, *Recursively Implementing the Gaussian and its Derivatives*, pp. 263–267. No. 1893, 1992.

[145] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, pp. 1942–1948 vol.4, Nov 1995.

[146] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pp. 69–73, May 1998.

[147] X. Mei, X. Sun, M. Zhou, shaohui Jiao, H. Wang, and X. Zhang, "On building an accurate stereo matching system on graphics hardware," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 467–474, Nov 2011.

[148] X. Sun, X. Mei, S. Jiao, M. Zhou, and H. Wang, "Stereo matching with reliable disparity propagation," in *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2011 International Conference on*, pp. 132–139, may 2011.

[149] X. Zhou and P. Boulanger, "New eye contact correction using radial basis function for wide baseline videoconference system," in *Proceedings of the 13th Pacific-Rim Conference on Advances in Multimedia Information Processing*, PCM'12, (Berlin, Heidelberg), pp. 68–79, Springer-Verlag, 2012.

[150] L. Xu and J. Jia, "Stereo matching: An outlier confidence approach," in *ECCV*, pp. 775–787, Springer, 2008.

[151] Y. Mizukami, K. Okada, A. Nomura, S. Nakanishi, and K. Tadamura, "Sub-pixel disparity search for binocular stereo vision," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, pp. 364–367, Nov 2012.

[152] Q. Yang, R. Yang, J. Davis, and D. Nister, "Spatial-depth super resolution for range images," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1–8, June 2007.

[153] C. C. Pham and J. W. Jeon, "Domain transformation-based efficient cost aggregation for local stereo matching," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 23, pp. 1119–1130, July 2013.

[154] L. De-Maeztu, S. Mattoccia, A. Villanueva, and R. Cabeza, "Efficient aggregation via iterative block-based adapting support-weights," in *3D Imaging (IC3D), 2011 International Conference on*, pp. 1–5, Dec 2011.

[155] D. Scharstein and C. Pal, "Learning conditional random fields for stereo," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1–8, June 2007.

[156] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1–8, June 2007.

[157] D. Scharstein, H. Hirschmüller, G. Kitajima, York an Krathwohl, N. Nesic, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *36th German Conference on Pattern Recognition (GCPR)*, Sep. 2014.

[158] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Proceedings of the Third European Conference on Computer Vision (Vol. II)*, ECCV '94, (Secaucus, NJ, USA), pp. 151–158, Springer-Verlag New York, Inc., 1994.

[159] K. Zhang, J. Li, Y. Li, W. Hu, L. Sun, and S. Yang, "Binary stereo matching," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, pp. 356–359, Nov 2012.

[160] H. Hirschmüller, P. R. Innocent, and J. Garibaldi, "Real-time correlation-based stereo vision with reduced border errors," *Int. J. Comput. Vision*, vol. 47, pp. 229–246, Apr. 2002.

[161] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Proceedings of the 10th Asian Conference on Computer Vision - Volume Part I*, ACCV'10, (Berlin, Heidelberg), pp. 25–38, Springer-Verlag, 2011.

[162] C. Stentoumis, L. Grammatikopoulos, I. Kalisperakis, and G. Karras, "On accurate dense stereo-matching using a local adaptive multi-cost approach," *{ISPRS} Journal of Photogrammetry and Remote Sensing*, vol. 91, no. 0, pp. 29 – 49, 2014.

[163] S. Sinha, D. Scharstein, and R. Szeliski, "Efficient high-resolution stereo matching using local plane sweeps," in *CVPR*, 2014.

[164] "Open source computer vision library." http://opencv.org. Accessed: 2014/11/02.

[165] N. Einecke and J. Eggert, "A two-stage correlation method for stereoscopic depth estimation," in *Digital Image Computing: Techniques and Applications (DICTA), 2010 International Conference on*, pp. 227–234, Dec 2010.

[166] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, (London, UK, UK), pp. 298–372, Springer-Verlag, 2000.

[167] K. Arun, T. Huang, and S. Blostein, "Least-squares fitting of two 3-d point sets," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-9, pp. 698–700, Sept 1987.

[168] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, June 1981.

[169] R. Szeliski, *Computer Vision: Algorithms and Applications*. New York, NY, USA: Springer-Verlag New York, Inc., 1st ed., 2010.

[170] R. Koch, M. Pollefeys, and L. J. V. Gool, "Multi viewpoint stereo from uncalibrated video sequences," in *Proceedings of the 5th European Conference on Computer Vision-Volume I - Volume I*, ECCV '98, (London, UK), pp. 55–71, Springer-Verlag, 1998.

[171] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, R. Yang, D. Nister, and M. Pollefeys, "Real-time visibility-based fusion of depth maps," in *ICCV*, pp. 1–8, 2007.

[172] C. Zach, "Fast and high quality fusion of depth maps," in *3D Data Processing, Visualization and Transmission (3DPVT'08), 4th International Symposium on*, 2008.

[173] J. Kowalczuk, E. Psota, and L. Perez, "Real-time temporal stereo matching using iterative adaptive support weights," in *Electro/Information Technology (EIT), 2013 IEEE International Conference on*, pp. 1–6, May 2013.

[174] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pp. 225–234, Nov 2007.

[175] L. Zhang, B. Curless, and S. M. Seitz, "Spacetime stereo: Shape recovery for dynamic scenes," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 367–374, June 2003.

[176] A. Hosni, C. Rhemann, M. Bleyer, and M. Gelautz, "Temporally consistent disparity and optical flow via efficient spatio-temporal filtering," in *Proceedings of the 5th Pacific Rim conference on Advances in Image and Video Technology - Volume Part I*, PSIVT'11, (Berlin, Heidelberg), pp. 165–177, Springer-Verlag, 2012.