THESE

pour obtenir le grade de
DOCTEUR DE L'ECOLE CENTRALE DE LYON
Spécialisté: Informatique

dans le cadre de l'Ecole Doctorale InfoMaths
présentée et soutenue

Contributions to Accurate and Efficient Cost Aggregation for Stereo Matching

---

Dongming CHEN
March 2015

Directeur de thèse: Liming CHEN
Co-directeur de thèse: Mohsen ARDABILIAN

JURY

Prof. Jean Ponce	École Normale Supérieure	Président
Prof. Peter Sturm	INRIA Grenoble	Rapporteur
Prof. Valérie Gouet-Brunet	IGN	Rapporteur
Prof. Liming Chen	École Centrale de Lyon	Directeur de thèse
Dr. Mohsen Ardabilian	École Centrale de Lyon	Co-directeur de thèse

Acknowledgments

First of all, I would like to express my gratitude to my supervisors, Professor Liming Chen and Associate Professor Mohsen Ardabilian, who afforded me the opportunity to work in their research group. They also gave me much freedom to pursue various research topics on 3D reconstruction that I am really interested in. I have learned much about 3D vision from their vast knowledge of the field.

I would like to thank my doctoral committee, Prof. Jean Ponce, Prof. Peter Sturm and Prof. Valérie Gouet-Brunet, for their interests on my work and their valuable comments. I would like to thank those colleagues who help me a lot on my research. For example, dr. Boyang Gao discussed with me and gave me many thoughtful suggestions, and he also help me to solve many practical programming problems; dr. Huibin Li helped me on my paper writing; Xiaofang Wang provided me many meaningful ideas and we had a nice cooperation together, *etc.*

Many thanks to my friends in the LIRIS Lab of Ecole Centrale de Lyon: Dr. Di Huang, Dr. Chao Zhu, Dr. Huibin Li, Dr. Boyang Gao, Dr. Ningning Liu, Dr. Yu Zhang, Dr. Kai Wang, Dr. Huiliang Jin, Dr. Bingxue Zhang, Yuxing Tang, Xiaofang Wang, Wei Chen, Wuming Zhang, Huaxiong Ding, Yinhang Tang, Ying Lv, Chen Wang, Fei Zheng, Xuchen Liu, *etc.* It is them who made my Ph.D. life colorful.

I would like to thank my country, China, and Chinese government, who fund me for three years.

Last but most importantly, great thanks to the support and understanding from my parents, my girlfriend and my family.

Abstract

3D-related applications are becoming more and more popular in our daily life, such as 3D movies, 3D printing, 3D maps, 3D object recognition, *etc.* Many applications require realistic 3D models and thus 3D reconstruction is a key technique behind them. In this thesis, we focus on a basic problem of 3D reconstruction, *i.e.* stereo matching, which searches for correspondences in a stereo pair or more images of a 3D scene. Although various stereo matching methods have been published in the past decades, it is still a challenging task since the high requirement of accuracy and efficiency in practical applications. For example, autonomous driving demands real-time stereo matching technique; while 3D object modeling demands high quality solution. This thesis is dedicated to develop efficient and accurate stereo matching method.

The well-known bilateral filter based adaptive support weight method represents the state-of-the-art local method, but it hardly sorts the ambiguity induced by nearby pixels at different disparities but with similar colors. Therefore, we proposed a novel trilateral filter based method that remedies such ambiguities by introducing a boundary strength term. As evaluated on the commonly accepted Middlebury benchmark, the proposed method is proved to be the most accurate local stereo matching method at the time of submission (April 2013).

The computational complexity of the trilateral filter based method is high and depends on the support window size. In order to enhance its computational efficiency, we proposed a recursive trilateral filter method, inspired by recursive filter. The raw costs are aggregated on a grid graph by four one-dimensional aggregations and its computational complexity proves to be $O(N)$, which is independent of the support window size. The practical runtime of the proposed recursive trilateral filter based method processing $375 * 450$ resolution image is roughly $260ms$ on a PC with a 3.4 GHz Inter Core $i7$ CPU, which is hundreds times faster than the original

trilateral filter based method.

The trilateral filter based method introduced a boundary strength term, which is computed from color edges, to handle the ambiguity induced by nearby pixels at different disparities but with similar colors. The color edges consist of two types of edges, *i.e.* depth edges and texture edges. Actually, only depth edges are useful for the boundary strength term. Therefore, we presented a depth edge detection method, aiming to pick out depth edges and proposed a depth edge trilateral filter based method. Evaluation on Middlebury benchmark proves the effectiveness of the proposed depth edge trilateral filter method, which is more accurate than the original trilateral filter method and other local stereo matching methods.

Résumé

Les applications basées sur 3D tels que les films 3D, l'impression 3D, la cartographie 3D, la reconnaissance 3D, sont de plus en plus présentes dans notre vie quotidienne; elles exigent une reconstruction 3D qui apparaît alors comme une technique clé. Dans cette thèse, nous nous intéressons à l'appariement stéréo qui est au cœur de l'acquisition 3D. Malgré les nombreuses publications traitant de l'appariement stéréo, il demeure un défi en raison des contraintes de précision et de temps de calcul: la conduite autonome requiert le temps réel; la modélisation d'objets 3D exige une précision et une résolution élevées.

La méthode de pondération adaptative des pixels de support (adaptaive-support-weight), basée sur le bien connu filtre bilatéral, est une méthode de l'état de l'art, de catégorie locale, qui en dépit de ses potentiels atouts peine à lever l'ambiguïté induite par des pixels voisins, de disparités différentes mais avec des couleurs similaires. Notre première contribution, à base de filtre trilatéral, est une solution pertinente qui tout en conservant les avantages du filtre bilatéral permet de lever l'ambiguïté mentionnée. Evaluée sur le corpus de référence, communément acceptée, Middlebury, elle se positionne comme étant la plus précise au moment où nous écrivons ces lignes.

Malgré ces performances, la complexité de notre première contribution est élevée. Elle dépend en effet de la taille de la fenêtre support. Nous avons proposé alors une implémentation récursive du filtre trilatérale, inspirée par les filtres récursifs. Ici, les coûts bruts en chaque pixel sont agrégés à travers une grille organisée en graphe. Quatre passages à une dimension permettent d'atteindre une complexité en O(N), indépendante cette fois de la taille de la fenêtre support. C'est-à-dire des centaines de fois plus rapide que la méthode originale.

Pour le calcul des pondérations des pixels du support, notre méthode basée sur le filtre trilatéral introduit un nouveau terme, qui est une fonction d'amplitude

du gradient. Celui-ci est remarquable aux bords des objets, mais aussi en cas de changement de couleurs et de texture au sein des objets. Or, le premier cas est déterminant dans l'estimation de la profondeur. La dernière contribution de cette thèse vise alors à distinguer les contours des objets de ceux issus du changement de couleur au sein de l'objet. Les évaluations, sur Middlebury, prouvent l'efficacité de la méthode proposée. Elle est en effet plus précise que la méthode basée sur le filtre trilatéral d'origine, mais aussi d'autres méthodes locales.

Contents

Chapter 1

Introduction

Contents

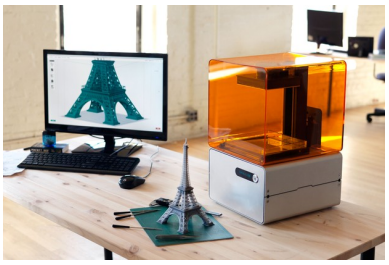
1.1 Research Topic

In recent years, 3D-related applications are becoming more and more popular in our daily life. As illustrated in Figure 1.1, 3D entertainment trends into the explosive spread of the entire field, *i.e.* 3D movies and 3D games; the use of 3D printing has significantly increased in the industrial manufacturing and has also made inroads into various households for personal use; the 3D map helps us get around more conveniently, *etc.* Realistic 3D models, more specifically the 3D reconstruction as the key technique to achieve them, are requisite in a plethora of applications including the above.

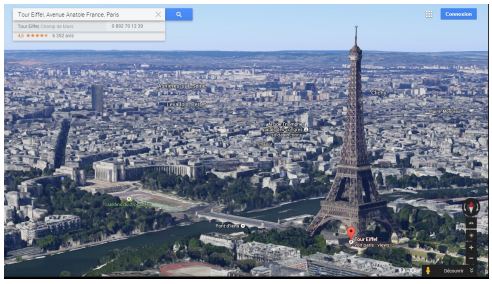
Over past decades, a variety of 3D reconstruction methods have been developed, which can be categorized into two classes: active methods and passive methods. Active methods actively interfere with the reconstructed object, either mechanically or radiometrically. A simple example of a mechanical method would use a depth gauge to measure a distance to a rotating object put on a turntable. More applicable radiometric methods emit radiance towards the object and then measure its reflected part. Examples range from moving light sources, colored visible light, time-of-

(a) 3D Game

(b) 3D Movie



(c) 3D Printer



(d) 3D Map

Figure 1.1: 3D-related applications need 3D models.

flight lasers to microwaves or ultrasound. On the other hand, passive methods do not interfere with the reconstructed object, they only use a sensor to measure the radiance reflected or emitted by the object's surface to infer its 3D structure. Typically, the sensor is an image sensor in a camera sensitive to visible light and the input to these methods is a set of digital images or video, the output is a 3D model.

Passive methods are widely applied in 3D reconstruction due to the less complicated and lower cost system. It requires one, two or multiple digital still or video cameras to obtain a sequence of images taken from different viewpoints. Different applications demand the systems with different camera amounts. In the case of a single camera based system, the camera captures consecutive images by moving the camera around the object and a 3D model is reconstructed from the resulting consecutive images. An challenge in this method is that the camera path must be estimated since precise calibration of freely moving cameras is extremely difficult. This problem of recovering scene geometry from the motion of points in the image

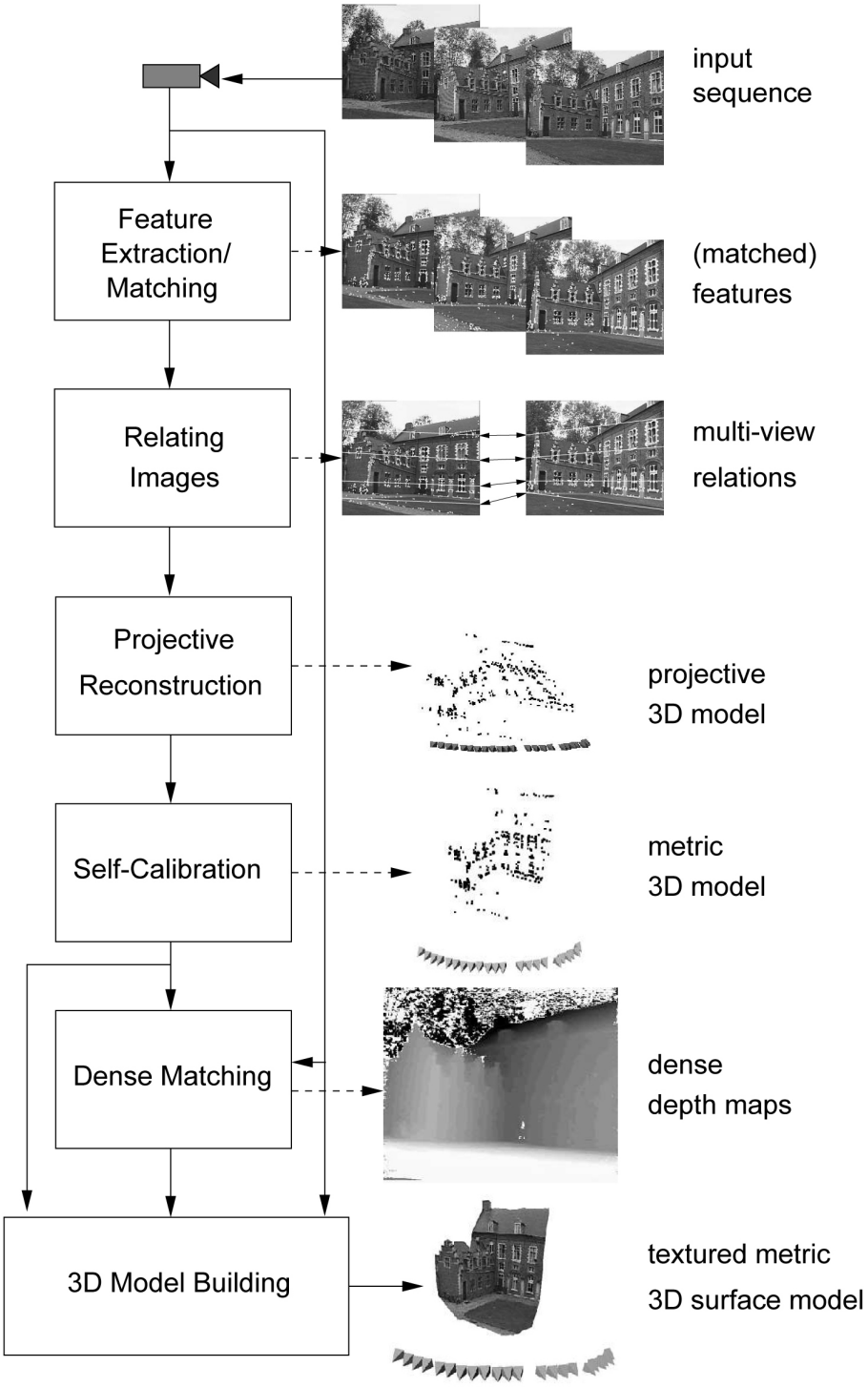


Figure 1.2: Image based 3D reconstructed system presented in [1].

plane of a moving camera is called structure from motion. It usually applies to reconstruct static objects, *e.g.* buildings. A structure from motion system [1] is presented in Figure 1.2. This system can be divided into three parts, (1) camera calibration, including *feature extraction and matching*, *relating images*, *projective reconstruction* and *self-calibration*, which estimates the camera parameters of each view; (2) stereo matching, *i.e. dense matching*, which calculates dense depth maps from related views; and (3) 3D modeling, *i.e. 3D model building*, which reconstructs a 3D model from depth maps.

In the case of two cameras based stereo vision system, when a point is imaged from two different viewpoints, its image projection undergoes a displacement from its position in the first image to that in the second image. The amount of displacement is inversely proportional to distance and may therefore be used to compute 3D geometry. Given a correspondence between imaged points from two known viewpoints, it is possible to compute depth by triangulation. The problem of establishing correspondence is a fundamental difficulty, which is the stereo matching problem. Different from the previous structure from motion system, this two cameras based stereo vision system can be used to obtain the disparity map of a dynamic scene.

In the case of multiple cameras based multi-view stereo vision system, it can apply to reconstruct dynamic objects, *e.g.* facial models with highly-transient expression, because multiple images can be simultaneously captured from multiple viewpoints. Different from the two view stereo vision system, the multi-view stereo vision system can obtain a completed 3D model of a dynamic object from more views. A multi-view stereo vision system is presented in [2] using seven cameras, which aims to reconstruct a realistic facial model. The reconstructed facial models with highly-transient expression are presented in Figure 1.3. The framework of this multi-view stereo vision system is similar to the previous structure from motion system [1] and also includes three parts, *i.e.* camera calibration, stereo matching and 3d modeling. After camera calibration, depth maps of different views are computed by stereo matching algorithm, and the depth maps are shown in Figure 1.4. Finally, a high quality 3D facial model, as shown in Figure 1.5, is reconstructed from these depth maps. Both camera calibration and stereo matching are two indispensable

Figure 1.3: Top: Images of a subject slapping himself and causing a shock-wave in the face. Bottom: the respective reconstructions obtained in [2]

Figure 1.4: pairwise stereo matching

and challenging parts of these systems [1] [2]. Many works on camera calibration can be found in literatures [3] [4] [5] [6] and this dissertation concentrates on the latter, *i.e.* stereo matching.

Shape from shading is a different kind of passive methods. The basic problem behind this method is that, given a single intensity image of a smooth curved object, then how to recovered the shape of the object. Given the intensity of a point in the image and a known directional light source, Lambert law yields a one-parameter family of solutions for the surface normal in literature [7]. Additional constraints are needed to make the problem well-posed. It is generally solved by assuming similarity of surface reflectance and orientation at nearby points.

Figure 1.5: reconstructed 3D model

1.2 Objective

We focus in this dissertation on a basic problem of passive methods, *i.e.* stereo matching, which remains one of the most active research topics in the computer vision. Stereo matching aims to find corresponding pairs in two images of the same scene captured in different views. The accuracy and efficiency are two metrics to evaluate a stereo matching method. Both of them are important as far as real world applications concerned. For example, navigation for autonomous vehicle needs real-time stereo vision techniques; while 3D object reconstruction needs high quality solutions. The objective of this dissertation is to research on the efficient and accurate stereo matching method.

Stereo matching methods can be broadly categorized into two classes: global methods and local methods. Global methods produce accurate results, but are quite time consuming due to iterative optimization process. Local methods are efficient but tend to be less accurate. However, in recent years, the accuracy of local methods is improving rapidly as the introduction of adaptive support weight based methods [8] [9] [10]. Therefore, we investigated popular adaptive support weight methods and proposed three methods, *i.e. trilateral filter based method*, *recursive trilateral filter based method*, and *depth edge based trilateral filter method*, in order to efficiently obtain an accurate depth map. The proposed methods, corresponding to our three contributions, are briefly introduced in the following section.

1.3 Overview of our Approaches and Contributions

In our first contribution, we proposed a novel trilateral filer based method. We analyzed the well-known bilateral filter based adaptive support weight method, which is the state-of-the-art local stereo matching method, and found it hardly sort the ambiguity induced by nearby pixels at different disparities but with similar colors. Therefore, we introduced a novel trilateral filter based method that remedies such ambiguities by considering the possible disparity discontinuities through color discontinuity boundaries, i.e. the boundary strength between two pixels, which is measured by a local energy model. As evaluated on the commonly accepted Middlebury benchmark, the proposed method is proved to be the most accurate local stereo matching method at the time of submission (April 2013).

In our second contribution, we proposed a recursive trilateral filter based method. The computational complexity of the trilateral filter based method is $O(Nr^2)$, where N denotes the image size and r denotes the support window radius. However, the support window size should be large enough to better handle untexture regions, e.g. 35×35 on the Middlebury stereo pairs. Therefore, the trilateral filter based method is quite time consuming. In order to improve the computational efficiency, we proposed a recursive trilateral filter method, inspired by recursive filter. In this method, the reference image is represented in a four-connected and weighted grid, whose vertices are image pixels and edges are weights, computed from the difference of intensity, between two neighboring pixels. Then, the weighted costs are aggregated on this grid graph by four passes, i.e. two horizontal passes and two vertical passes. The computational complexity of the recursive trilateral filter method is $O(N)$, which is independent of the window size r. The runtime time of recursive trilateral filter cost aggregation processing $375 * 450$ resolution image is roughly $260ms$ on a PC with a 3.4 GHz Inter Core i7 CPU, which is hundreds times faster than the original trilateral filter cost aggregation.

In our third contribution, we proposed a depth edge trilateral filter based method. The previous trilateral filter based method introduced a boundary strength term, which is the key part of this method. The boundary strength term is com-

puted from a color edge map. However, we verified that, if the boundary strength term is computed from a depth edge map, the final result is more accurate than that computed from a color edge map. Motivated by this observation, we present a depth edge detection method to obtain an accurate depth edge map, which replaces the color edge map used in the trilateral filter based method. The experimental evaluation on the Middlebury benchmark shows that the proposed depth edge trilateral filter method outperforms the original trilateral filter method and also other local stereo matching methods.

1.4 Organization of the thesis

The rest of this dissertation is organized as follows. In Chapter 2, we introduce the fundamental theory of stereo vision, including *pinhole camera model*, *epioplar geometry*, *image rectification*, *stereo matching* and *triangulation*. This chapter could be skipped for readers who are familiar with these concepts. In Chapter 3, we review various stereo matching methods, which are the key part of a stereo vision system. In Chapter 4, we propose a novel trilateral filter based method, which is our first contribution. This method is the most accurate local stereo matching method at the time of submission (April 2013). However, this method is quite time consuming, whose computational complexity is $O(Nr^2)$ where N denotes the image size and r denotes the support window size. Therefore, we propose in Chapter 5 an efficient recursive trilateral filter based method, which is our second contribution. This method is 300× faster than the previous trilateral filter based method, while still providing a high accuracy. In Chapter 6, we verified that depth edge map can improve the accuracy of the trilateral filter based method, and thus propose a depth edge based trilateral filter method, which is our third contribution. In Chapter 7, our conclusions as well as some perspective for future research directions are proposed.

Chapter 2

Fundamental of stereo vision

Contents

In this Chapter, we present the fundamental theory behind stereo vision, including *Pinhole Camera Model*, *Epipolar Geometry*, *Image Rectification*, *Stereo Matching* and *Triangulation*. The following discussion is at a general level and could be skipped for readers who are familiar with these concepts. Interested readers could refer to some books for more discussion about stereo vision [11] [12].

The framework of a typical stereo vision system is illustrated in Figure 2.1. Stereo vision system is motivated from the human vision system, which can perceive depth properties of a scene. The human vision system obtains two different images of a scene from the slightly different views of the eyes and interprets the images for depth perception. Similarly, stereo vision system consists of two cameras (two eyes), which simultaneously capture the same scene from different views, and a hardware (a brain), which compute the 3D information of the scene from these two views.

We sequentially explain each stage of the framework of stereo vision system as follows. First of all, a point on the surface of an object is projected onto image planes of two cameras, respectively. This projection from 3D point to 2D pixel can be described by *pinhole camera model*, which is explained in Section 2.1. The essence of

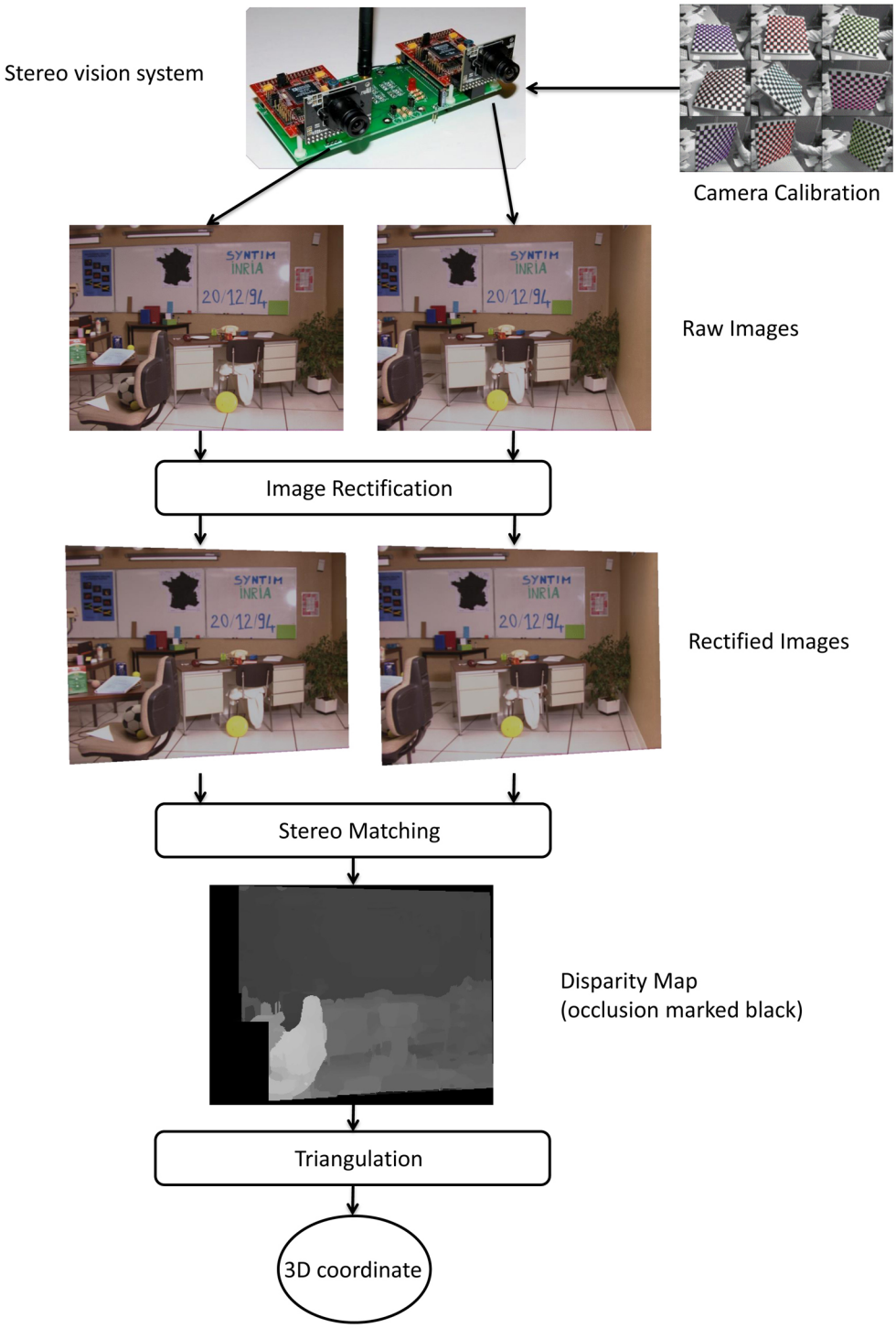


Figure 2.1: The framework of a typical stereo vision system.

pinhole camera model is a 3×4 camera projection matrix, which consists of internal camera parameters and external camera parameters. These camera parameters can be computed by camera calibration techniques. After obtaining camera parameters, the inherent problem in stereo vision comes out, *i.e.* correspondence problem, which refers to the problem of ascertaining which parts of one image correspond to which parts of the other image. It seems to require a 2D searching through the whole image. However, after understanding *Epipolar Geometry* introduced in Section 2.2, it is known that the 2D searching can be simplified to a 1D searching on a single Epipolar line due to Epipolar constraint. Moreover, this searching problem can be further simplified by *Image Rectification* technique, which is introduced in Section 2.3. After image rectification, the corresponding pairs locate on the horizontal scan-line with the same height and the Epipolar lines are rectified horizontal, because these two images are transformed onto a common image plane. Then, stereo correspondences are matched on the rectified images by stereo matching algorithms. We briefly explain *Stereo Matching* in Section 2.4 and review the state-of-the-art algorithms in Chapter 3 in detail. The output of stereo matching is a disparity map, which refers to the apparent pixel location difference between a pair of stereo image. Finally, the 3D coordinate of each pixel on the disparity map can be computed by *Triangulation* as presented in Section 2.5. In this framework, stereo matching is considered as the most challenging and unsolved problem.

2.1 Pinhole Camera Model

The pinhole camera model represents a mathematical mapping between the coordinate of a 3D point and its 2D projection on the image plane. This mathematical mapping is illustrated in Figure 2.2. The camera aperture of the pinhole camera is described as a point and no lenses are used to focus light.

The *camera center* C, where the camera aperture is located, is set to be the origin of a Euclidean coordinate system. The axis Z, which points in the viewing direction of the camera, is referred to as the *principal axis*. The *image plane* is parallel to axes X, Y and is located at distance f from the camera center C in the

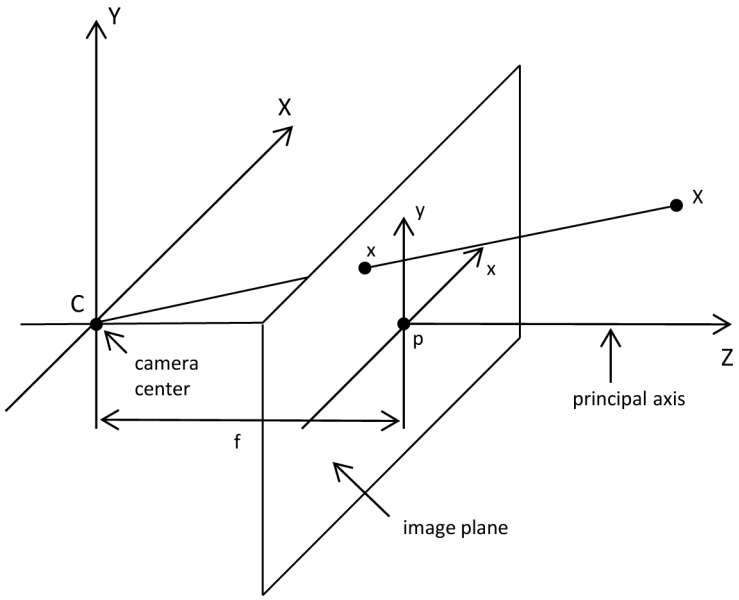


Figure 2.2: pinhole camera model.

direction of the Z axis, *i.e.* the image plane is $Z = f$, where f is referred to as the focal length. The point p, intersection of the principal axis and the image plane, is referred to as the *principal point* or *image center*. Given a point X in the 3D world, whose homogeneous coordinate is $(X, Y, Z, 1)^T$. This point is mapped to the image plane at point x, where a line joining the point X to the camera center C meets the image plane. By similar triangles, the coordinate of x is as,

$$\frac{f}{Z} = \frac{x}{X} = \frac{y}{Y} \tag{2.1}$$

which gives us

$$\begin{aligned} x &= \frac{fX}{Z} \\ y &= \frac{fY}{Z} \end{aligned} \tag{2.2}$$

Therefore, the homogeneous coordinates of the mapped point x on the image plane is $(fX/Z, fY/Z, 1)^T$. This projection can be written in terms of matrix multiplication

as,

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.3}$$

The above expression, Equation (2.3), assumed that the origin of the 2D image coordinate system coincides with where the Z axis intersects the image plane. However, it may not be. We need to translate the 2D point x to the desired origin. Let this translation be defined by $(p_x, p_y)^T$. Thus, the coordinate of x is

$$\begin{aligned} x &= \frac{fX}{Z} + p_x \\ y &= \frac{fY}{Z} + p_y \end{aligned} \tag{2.4}$$

This can be expressed in a similar form as Equation (2.3) in homogeneous coordinates as

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.5}$$

In the above expression, Equation (2.5), the coordinate is expressed in millimetre (mm). We need to know the resolution of the camera in *pixels/mm*. If the pixels are square, the resolution will be identical in both x and y directions of the camera image coordinates. However, for a more general case, the image Euclidean coordinates are possibly having unequal scales in both axial directions and we assume rectangle pixels with resolution m_x and m_y *pixels/mm* in those two direction respectively. Therefore, to measure the point x in pixels, its coordinates should be multiplied by m_x and m_y respectively as,

$$\begin{aligned} x &= m_x \frac{fX}{Z} + m_x p_x \\ y &= m_y \frac{fY}{Z} + m_y p_y \end{aligned} \tag{2.6}$$

This can be expressed in matrix form as

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} m_x f & 0 & m_x p_x & 0 \\ 0 & m_y f & m_y p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} a_x & 0 & q_x & 0 \\ 0 & a_y & q_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.7}$$

where a_x and a_y represent the focal length of the camera in terms of pixel dimensions in the x and y direction respectively. Sometimes if the image coordinate axes x and y are not orthogonal to each other, then the skew parameter s is needed. Therefore, the general matrix form of the mapping from 3D point to 2D pixel is

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} a_x & s & q_x & 0 \\ 0 & a_y & q_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.8}$$

These parameters (a_x, a_y, s, q_x, q_y) are the internal camera parameters, which encompass focal length (a_x, a_y), principle point (q_x, q_y) and skew coefficient s. Nonlinear internal parameters such as lens distortion are also important although they cannot be included in the linear camera model described by the internal parameter matrix as Equation (2.8). Many modern camera calibration algorithms estimate these nonlinear internal parameters.

The relationship between the 3D point $(X, Y, Z, 1)$ and its 2D projection (x, y, z) in Equations (2.8) is in the camera coordinate frame. In general, the camera coordinate frame is not the world coordinate frame, but they are related by a rotation and a translation, which are external camera parameters. Specifically, external camera parameters define the position of the camera center and the camera's heading in world coordinates. Let $\tilde{X}$ denotes the coordinate of a point in the world coordinate frame and X represents the same point in the camera coordinate frame as used in Equation (2.8), then their relationship is $X = R\tilde{X} + t$, where t represents a 3×1 translation matrix and R is a 3×3 rotation matrix. This relationship can be written

in terms of matrix multiplication as,

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \\ 1 \end{pmatrix} \tag{2.9}$$

Therefore, in the world coordinate frame, the mapping from 3D point to 2D pixel can be rewritten as,

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} a_x & s & p_x & 0 \\ 0 & a_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \\ 1 \end{pmatrix}. \tag{2.10}$$

This mapping has a concise form as $x = K[R|t]\tilde{X}$, where

$K = \begin{bmatrix} a_x & s & p_x \\ 0 & a_y & p_y \\ 0 & 0 & 1 \end{bmatrix}$, $R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ and $t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$.

Let $P = K[R|t]$ and the 3×4 matrix P is known as camera projection matrix, which is the essence of pinhole camera model. The parameters in matrix K are internal camera parameters and those in matrix R and t are external camera parameters, which can be found by camera calibration technique [13] [14] [3].

2.2 Epipolar Geometry

Pinhole camera model represents the mapping from 3D world to 2D points on the image plane. Stereo vision system includes two pinhole cameras, which simultaneously capture the same scene from two different views. The geometric relations between these two views can be represented by *Epipolar Geometry* as shown in Figure 2.3. Suppose a point X in the 3D world is imaged in these two cameras, at x_1 in the left one and x_2 in the right one. After camera calibration, the camera parameters of left camera and right camera can be computed respectively and two

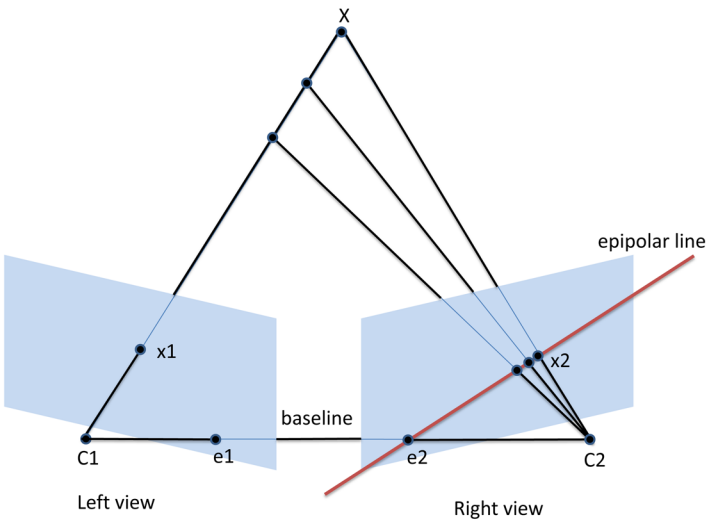


Figure 2.3: Epipolar Geometry.

camera projection matrixes P_l and P_r are obtained. Then the coordinates of point x_1 and x_2 can be calculated from the coordinate of X by Equation (2.10). Conversely, if the coordinates of the point x_1 and its correspondence x_2 are found, then the coordinate of X can be calculated by rays back-projection from the camera center C_1 and C_2 to the projection x_1 and x_2. Therefore, matching a given point x_1 in one image with its correspondence point x_2 in the other image is the key part of a stereo vision system.

This correspondence matching problem seems to require a 2D search through the whole image. However, the epipolar constraint reduces the search space to a single line. The *Epipolar Constraint* is that the corresponding pixel x_2 must lie on the epipolar line of pixel x_1. This constraint is described as follows. Suppose that only the coordinate of x_1 is known, the 3D point X must at the ray back-projected from the camera center C_1 to the point x_1. From Equation (2.10), we known that this ray back-projection can be obtained by solving $x_1 = P_l X$, where P_l is the left camera projection matrix. The one-parameter family of solutions is as,

$$X(\lambda) = P_l^{+} x_1 + \lambda C_1 \tag{2.11}$$

where P_l^+ is the pseudo-inverse of P_l, *i.e.* $P_l P_l^+ = I$ and the left camera center C_1 is the null-vector of P_l, *i.e.* $P_l C_1 = 0$. The ray is parameterized by the scalar λ and contains two points, *i.e.* $P_l^+ x_1$ (at $\lambda = 0$) and C_1 (at $\lambda = \infty$). These two points are imaged by the right camera, whose camera projection matrix is P_r, at $P_r P_l^+ x_1$ and $P_r C_1$ respectively. The ray is mapped to the right camera as a line, marked in red color in Figure 2.3. This line is the epipolar line of x_1, which joining the two projected points $P_r P_l^+ x_1$ and $P_r C_1$, *i.e.* $l = (P_r C_1) \times (P_r P_l^+ x_1)$. The corresponding point x_2 must lie on the epipolar line l. Therefore, we can conclude that, for any pixel x in one image, its corresponding pixels x' in the other image must lie on its epipolar line $l = (P_r C_1) \times (P_r P_l^+ x)$. The left camera center C_1 projects on the right image plane at point $e_2 = P_r C_1$, which is called *epipolar point* and each epipolar line in the right image must pass through the epipolar point e_2. The line joining two camera center C_1 and C_2 is named *baseline*.

2.3 Image Rectification

Although the epipolar constraint simplifies the correspondence search along a single line, the arbitrary orientation of the line makes it inconvenient for algorithms to compare pixels. The image rectification technique is used to overcome this issue. Image rectification is a transformation process used to project two images onto a common image plane and pairs of conjugate epipolar lines become collinear, as shown in Figure 2.4. The idea behind rectification is to define two new camera matrix P_l' and P_r' obtained by rotating the old ones P_l and P_r around their camera center until the image planes becomes coplanar, parallel to the baseline. Any pair of images can be transformed so that epipolar lines are parallel and horizontal in each image, the epipolar points are at infinity. The new camera matrix P_l' and P_r' will have both the same internal parameters, the same orientation but different position. As analyzed in [15], the new camera matrixes are as

$$\begin{aligned} P_l' &= A[R|-RC_1] \\ P_r' &= A[R|-RC_2] \end{aligned} \tag{2.12}$$

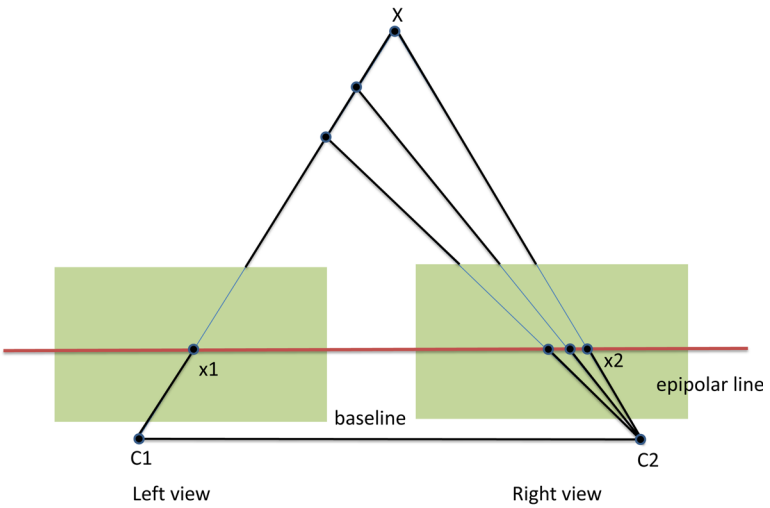


Figure 2.4: Image Rectification.

The internal parameters matrix A is the same for both camera matrixes and can be chosen arbitrarily. C_1 and C_2 denote two camera centers. The matrix R, which gives the camera's pose, is the same for both camera matrixes. It will be specified by means of its row vectors $R = [r1^T, r2^T, r3^T]^T$ that are the X,Y and Z axes, respectively. The new X axis is parallel to the baseline: $r1 = (C_1 - C_2)/||C_1 - C_2||$. The new Y axis is orthogonal to X and to an arbitrary unit vector k, that fixes the position of the new Y axis in the plane orthogonal to X: $r2 = k \times r1$. The new Z axis is orthogonal to XY: $r3 = r1 \times r2$.

In order to rectify the image, taking left image as an example, we need to compute the transformation that maps the image plane of the old camera matrix $P_l = [Q_1|q_1]$, where Q_1 is the 3×3 matrix and q_1 is the 3×1 matrix, onto the image plane of new camera matrix $P_l' = [Q_1'|q_1']$. Then, the transformation matrix for the left image from the old camera matrix to the new one is $T = Q_1'Q_1^{-1}$. The transformation T is then applied to the original left image to produce the rectified image.

2.4 Stereo Matching

The goal of stereo matching is to identify corresponding pairs, *e.g.* x_1 and x_2, in the left and right image. After image rectification, the stereo matching problem becomes easier since the corresponding pairs locate on the scan-line with the same height. The stereo matching is the most important part for a stereo vision system and state-of-the-art stereo matching methods are reviewed in Chapter 3 in detail. The output of stereo matching is a grey-scale image of the disparity values for each point, also named as *disparity map*, where disparity refers to the distance between two corresponding points of the left and right image.

2.5 Triangulation

Assume that the disparity map is obtained by stereo matching, then the 3D position of each pixel of the disparity map can be determined by triangulation, as shown in Figure 2.5. P denotes a point in 3D world, whose coordinate is (X, Y, Z) and it projects on the left camera as $p_1 = (x_1, y_1)$ and on the right camera as $p_2 = (x_2, y_2)$. Due to the image rectification described in Section 2.3, epipolar lines are horizontal and correspondence pixels lie on the scanline with the same height, *i.e.* $y_1 = y_2$. The baseline b represents the distance between two camera centers C_1 and C_2. f is the camera focal length. Z denotes the depth of P from the camera and is what we should compute. Then, an auxiliary line (the red line in Figure 2.5) passing through C_2, which is parallel to line PC_1, is made and meets the right image plane at A. Obviously, the triangle C_2Ap_2 is similar with triangle PC_1C_2, and then we have the following equation as,

$$\frac{Z}{b} = \frac{f}{x_1 - x_2}, \tag{2.13}$$

and finally the depth of point P is derived as,

$$Z = \frac{f \times b}{x_1 - x_2}, \tag{2.14}$$

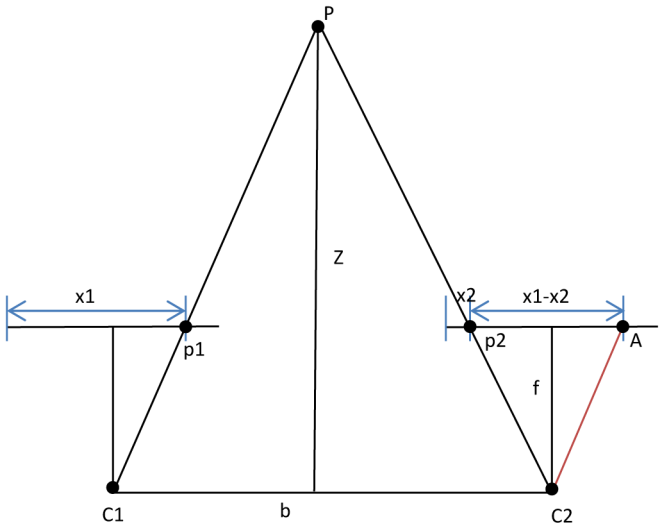


Figure 2.5: Triangulation.

where $x_1 - x_2$ is the disparity. From this equation, we know that the disparity map is inversely proportional to the depth map.

2.6 Conclusion

In this Chapter, we introduced the fundamental theory behind stereo vision. Firstly, we explain pinhole camera model, which is the mapping between a 3D point and its 2D projections on the image plane. Then, the stereo matching problem can be simplified by epipolar constraint that limits the matching problem along an epipolar line. The stereo matching is further simplified by image rectification, which transforms two views to share the common image plane. After image rectification, the corresponding pairs locate on the scan-line with the same height. Then, corresponding pairs are matched and a disparity map is calculated by stereo matching algorithm. Finally, the 3D coordinates of each pixels on the disparity map can be computed by triangulation.

Chapter 3

Literature Review

Contents

Stereo matching is the key part of stereo vision, which aims to find the corresponding pairs between two images of a 3D scene. According to the classical taxonomy proposed by Scharstein and Szeliski [16], stereo matching methods can be broadly categorized into two classes: global methods and local methods. Global methods compute all disparities simultaneously by minimizing a defined 2D energy function through a global optimization algorithm, *e.g.*, graph cut [17], belief

propagation [18], scanline optimization [19] and dynamic programming [20]. Global methods tend to produce more accurate matching results, but they are generally computationally expensive due to the iterative nature of the underlying optimization process. Local methods consider correlations between intensity patterns in support windows. The raw matching costs of all pixels within a support window are aggregated to the center pixel at each disparity; then an optimal disparity that gives a minimum aggregated cost is selected through an efficient local optimization process. Compared to global methods, local methods generally better satisfy the requirement of high-speed applications, but mostly at the expense of accuracy. In recent years, local methods have again become very popular, since some state-of-the-art local methods [9] [21] [22] [8] [10] [23] achieve high accuracy comparable to that of global methods.

The stereo matching methods, *i.e.* global methods and local methods, usually preform four steps as, (1) matching cost computation, (2) cost aggregation, (3) disparity optimization and (4) disparity refinement. The actual sequence of these four steps taken depends on the specific algorithm. Global methods typically do not perform the cost aggregation step, while some local methods do not perform the disparity refinement step. We review state-of-the-art stereo matching methods from these four aspects in the following sections.

3.1 Matching Cost Computation

Matching cost computation is the necessary step for both local methods and global methods. A matching cost, which indicates the similarity of two pixels corresponding to the same scene point, is computed at each pixel for all disparities under consideration. Local methods usually aggregate the sum of the matching cost of each pixel within a local window. This aggregation is named as cost aggregation, which will be described in Section 3.2. Global methods use the differences pixel-wise directly in disparity optimization step, which will be described in Section 3.3.

Various cost functions have been proposed to compute the matching costs such as [24], Absolute Differences (AD), Squared Differences (SD), Truncated Absolute

Difference of Color and Gradient (TADCG), Mutual Information (MI), transformed based method, *etc.* To our knowledge, truncated absolute difference of color and gradient is the most popular cost function and usually used in state-of-the-art stereo matching methods [9] [21] [25] [22].

3.1.1 Absolute Differences and Squared Differences

The absolute differences (AD) and squared differences (SD) cost functions are simple but usually used due to their low computational complexity. They are also used in other applications, such as motion estimation for video compression [26] [27], optical flow [28], object tracking [29], *etc.* They assume brightness constancy for corresponding pixels in the left and right images.

The pixel-wise absolute difference between pixel p in the left image I_l and pixel $p-d$ in the right image I_r is as

$$C_{AD}(p,d) = |I_l(p) - I_r(p-d)|, \tag{3.1}$$

and the pixel-wise squared difference is in a similar way as

$$C_{SD}(p,d) = (I_l(p) - I_r(p-d))^2. \tag{3.2}$$

3.1.2 Truncated Absolute Difference of Color and Gradient

In order to handle the brightness inconstancy of corresponding pixels in both images, gradient cue is considered, since gradient cue is more robust to additive brightness changes than the color cue. Truncated absolute difference of color and gradient (TADCG) is proposed, which consists of two parts, *i.e.* color part and gradient part. The color part is the same as the absolute difference function shown in Equation (3.1), while the gradient part is expressed as,

$$C_{gradient}(p,d) = |\nabla_x I_l(p) - \nabla_x I_r(p-d)|\,, \tag{3.3}$$

where ∇_x is the derivative in the x direction. Then, these two parts are combined with truncation to be the final truncated absolute difference of color and gradient cost function as,

$$\begin{aligned} C_{TADCG}(p,d) =&(1-\theta) \times min(|I_l(p) - I_r(p-d)|, \tau_1) \\ &+ \theta \times min(|\nabla_x I_l(p) - \nabla_x I_r(p-d)|, \tau_2), \end{aligned} \tag{3.4}$$

where θ balances the color part and gradient part; τ_1, τ_2 are truncation values in order to reduce the influence of occluded pixels.

The TADCG cost function is also used in optical flow estimation [30] [31].

3.1.3 Transformed based Function

In order to remove the noise and outliers, the input images are preprocessed by filters in literatures [24] [32] [33]; then matching costs are computed on the transformed images using absolute difference, squared difference or truncated absolute difference of color and grandient, *etc.* We present two filters, *i.e.* mean filter and Laplacian of Gaussian filter.

Mean filter is a simple, intuitive and easy to implement method of removing noises and outliers. The idea behind mean filter is to replace each pixel value in an image with the mean value within a window, including itself, as

$$I_{MF}(x,y) = \frac{1}{(2n+1)^2} \sum_{i=-n}^{n} \sum_{j=-n}^{n} I(x+i, y+j), \tag{3.5}$$

where the window size is $(2n+1) \times (2n+1)$.

Laplacian of Gaussian filter (LoG) performs smoothing, removing noise and changes in bias. This filter is often used in local real-time methods [34] [35] and is expressed as [32]

$$K_{LoG} = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right) \exp -\frac{x^2+y^2}{2\sigma^2}, \tag{3.6}$$

where σ refers to the standard deviation. Then, the input image is convoluted with

the above LoG filter as,

$$I_{LoG} = I \bigotimes K_{LoG} \tag{3.7}$$

where $\bigotimes$ denotes a convolution operator.

3.2 Cost Aggregation

Cost aggregation is the most important step for local methods, since both the accuracy and efficiency of local methods highly depend on cost aggregation. However, cost aggregation is not performed in global methods. Researches on cost aggregation date back to the 70s [36] [37] [38] [39]. In this step, a support window is selected for each pixel and all matching costs within this window are aggregated as its overall cost. That is, an entire window rather than a single pixel is considered for stereo matching in order to obtain a more reliable result. We present in this section various cost aggregation approaches and some of them have been analyzed in [40] [41], including the fixed window approach [16], the shiftable window approach [42], the variable window approach [43], the multiple window approach [34] and the adaptive support weight based approach [9] [21] [22] [8] [10] [23]. The fixed window approach, the shiftable window approach, the variable window approach and multiple window approach are out of date, while the adaptive support weight based approach is the state-of-the-art.

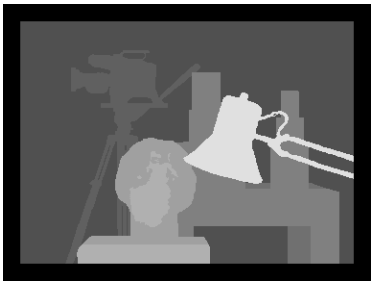
3.2.1 Out-of-date Approaches

3.2.1.1 Fixed Window Approach

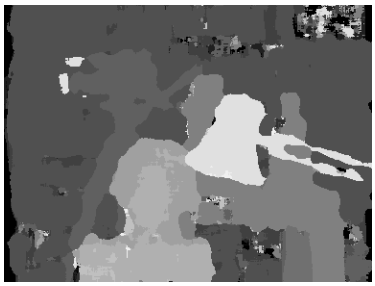
The fixed window approach [16] is a traditional and the simplest approach. A support window with fixed size is set for each pixel and all matching costs within this window are aggregated as

$$C_{aggr}(p,d) = \sum_{q \in \omega_p} C_{raw}(q,d) \tag{3.8}$$

(a) reference image



(b) ground truth disparity map



(c) Disparity map by fixed window

Figure 3.1: Disparity map computed by fixed window approach. The fixed window approach violate the basic assumption at disparity discontinuities that all support pixels in the support window have the same disparity value as the center one.

where ω_p denotes a support window centered at pixel p; q is a support pixel in ω_p. The raw cost C_{raw} can be computed by different cost functions described in Section 3.1, *e.g.* absolute difference, square difference, truncated absolute difference of color and gradient, *etc.* If the absolute difference (AD) based cost function is chosen, then the aggregated cost is known as the sum of absolute difference (SAD).

This fixed window approach implicitly assumes that all support pixels in the fixed window ω_p have the same disparity value as the center one; but this assumption is violated at disparity discontinuities. Kanade and Okutomi [44] claimed that the fixed window approach is likely to fail when the support window covers a region with non-constant disparity. The final disparity map computed by the fixed window approach is shown in Figure 3.1 and it is observed that the depth discontinuity borders are always inaccurate.

Figure 3.2: Disparity map computed by normalized cross-correlation.

3.2.1.2 Normalized Cross-Correlation

The normalized cross-correlation (NCC) is another commonly used cost aggregation strategy, which is an optimal method for dealing with Gaussian noise. The normalized cross-correlation cost is computed as follows,

$$C_{aggregated}(p,d) = \frac{\sum_{q\in\omega_p} I_l(q)I_r(q-d)}{\sqrt{\sum_{q\in\omega_p} I_l(q)^2 \sum_{q\in\omega_p} I_r(q-d)^2}} \tag{3.9}$$

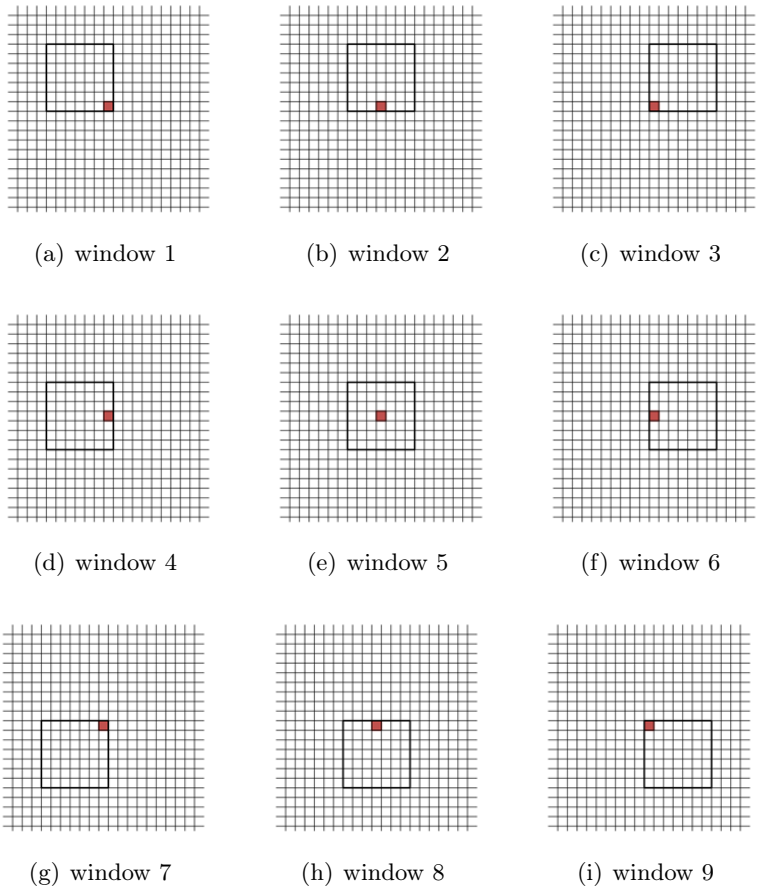
where ω_p denotes a support window centered at pixel p; q is a support pixel in ω_p.

The disparity map computed by this approach is presented in Figure 3.2. This approach also set a fixed window for each pixel and compare two fixed window at two images, which is the same as the previous fixed window approach. We can observe that this approach also leads to inaccurate estimation at depth discontinuity borders as the fixed window approach.

In order to handle the disparity discontinuity, shiftable window approach, multiple window approach and variable window approach are proposed and introduced in the following subsections.

3.2.1.3 Shiftable Window Approach

In shiftable window approach [39] [42] [45], nine windows centered at different locations, as shown in Figure 3.3, are considered for each pixel, and an optimal window that gives the smallest average cost is selected from these nine windows. The theory

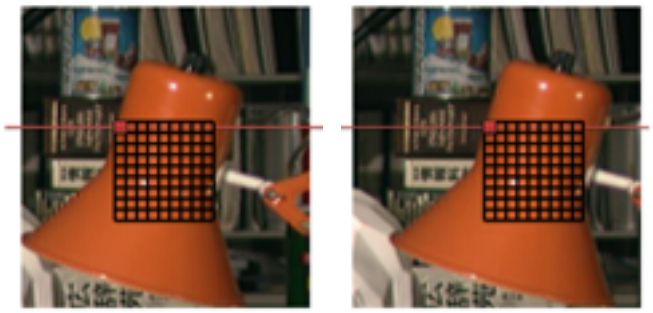


(a) window 1 (b) window 2 (c) window 3

(d) window 4 (e) window 5 (f) window 6

(g) window 7 (h) window 8 (i) window 9

Figure 3.3: The nine asymmetric support windows. The pixel for which disparity is computed is highlighted in red. An optimal support window is adaptively picked out from these nine candidates for each pixel.



(a) left image (b) right image

Figure 3.4: An optimal support window is adaptively picked out for the pixel highlighted in red.

Figure 3.5: Disparity map computed by shiftable window approach.

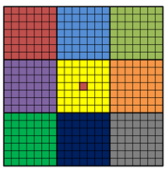
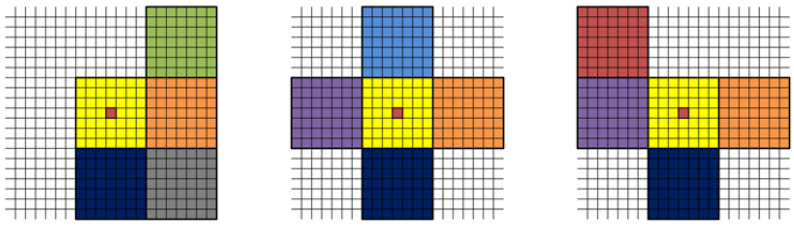


Figure 3.6: Nine window candidates marked in different color for the center pixel highlighted in red.

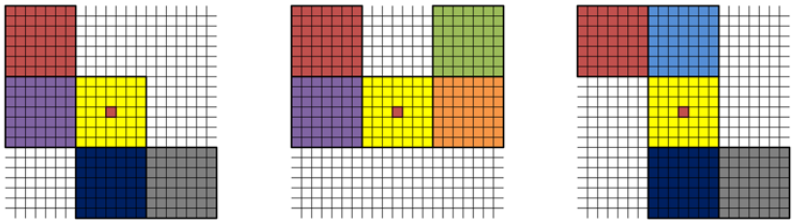
behind this approach is that a window yielding a smaller average cost is more likely to cover a region in which all pixels share a common disparity. In this way, the disparity profile itself drives the selection of an appropriate window, as illustrated in Figure 3.4. The disparity map computed by shiftable window approach is shown in Figure 3.5.

3.2.1.4 Multiple Window Approach

Multiple window approach [34] aims to handle disparity discontinuity regions by changing the window shape. Three configurations with different window amounts are proposed in [34], *i.e.* 5 windows, 9 windows and 25 windows. Take the configuration with 9 windows as an example, which is shown in Figure 3.6. Nine sub-window candidates are set for each pixel, different sub-window candidates are marked by different colors. Then, five windows are adaptively selected from these nine sub-windows to be the final support window. In this adaptive selection, the center sub-window is required; thus four sub-windows are picked out from the rest eight



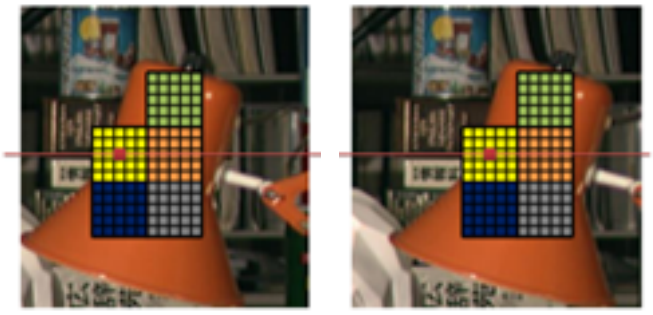
(a) shape 1 (b) shape 2 (c) shape 3



(d) shape 4 (e) shape 5 (f) shape 6

Figure 3.7: Selected windows in different shapes.

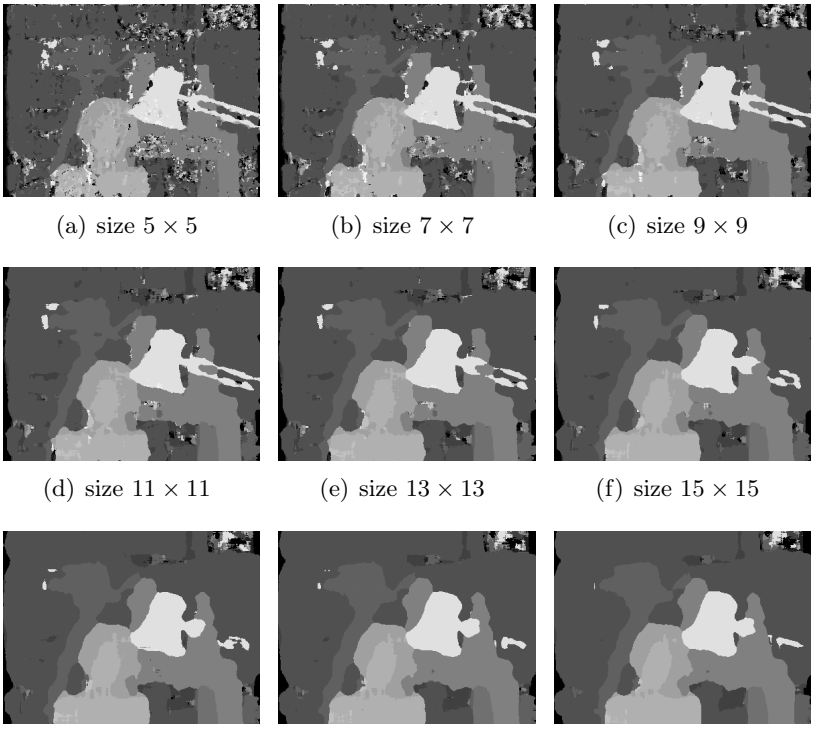
sub-windows. Therefore, windows in different shapes are selected depending on the local image content. We present six different shapes in Figure 3.7. Ideally, pixels in the selected windows should be located at the same disparity, as illustrated in Figure 3.8. The disparity map computed by multiple window approach is shown in Figure 3.9.



(a) left image (b) right image

Figure 3.8: An optimal support window is adaptively picked out for the pixel highlighted in red.

Figure 3.9: Disparity map computed by multiple window approach.

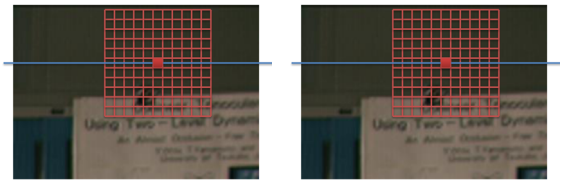


(a) size 5×5 (b) size 7×7 (c) size 9×9

(d) size 11×11 (e) size 13×13 (f) size 15×15

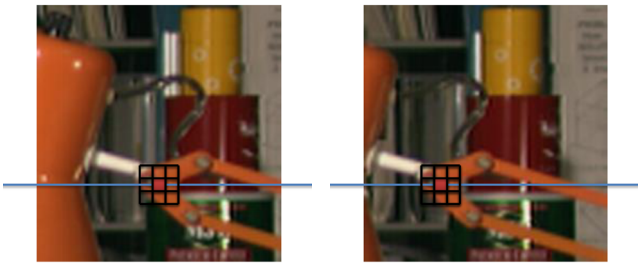
(g) size 17×17 (h) size 19×19 (i) size 21×21

Figure 3.10: Results with different window sizes.



(a) left image (b) right image

Figure 3.11: Large window is set for untextured region.



(a) left image (b) right image

Figure 3.12: Small window is set for region with complex depth variation.

3.2.1.5 Variable Window Approach

Variable window approach [43] [37] handles the disparity discontinuities by changing the window size and shape. The theory behind this approach is twofold, (1) a window must be large enough to have sufficient intensity variation, but must be small enough to contain only pixels at approximately equal disparity. We present the results computed with different window sizes in Figure 3.10. It is observed that large window performs better at untextured regions, since it contains more information, as illustrated in Figure 3.11; and small window performs better at regions with complex depth variation, since it avoids including pixels in different disparities, as illustrated in Figure 3.12. Therefore, the window size should be adaptively changed according the local image content; (2) near disparity boundaries, windows of different shapes are needed to avoid crossing the boundary, thus the window shape should also be adaptively changed. The idea of changing the window shape is the same as the shiftable window approach.

The disparity map computed by variable window approach is presented in Figure 3.13. We can observe that this approach performs better on both untextured regions

Figure 3.13: Disparity map computed by variable window approach.

and regions with complex depth variation.

3.2.2 State-of-the-art Approaches

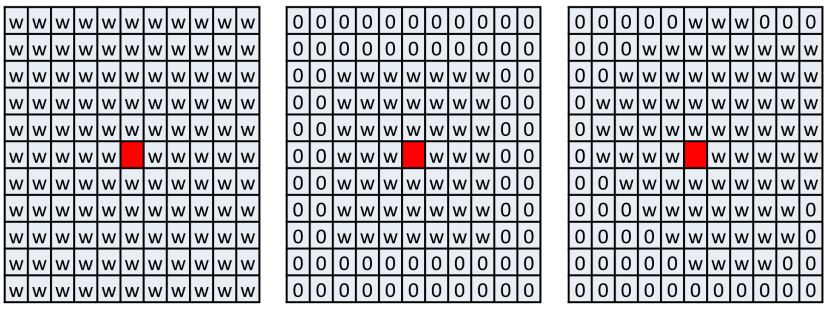
Before describing the state-of-the-art approaches, we shortly conclude previous approaches. The fixed window approach and normalized correlation cross approach violates the basic assumption [1] at disparity discontinuities. In order to handle this problem, various cost aggregation strategies have been proposed, including the shiftable window approach [39] [45] [42], the multiple window approach [34], the variable window approach [43], *etc.* The main idea of these methods is to model the optimal support window for each pixel by changing window size [43] [46] [47] [44] and shape [34] [16] [48] [42] [43] [39] [45] [42].

The adaptive support weight approach represents the state-of-the-art cost aggregation which is firstly proposed by Yoon and Kweon [8]. This approach dramatically outperforms the previous cost aggregation approaches. The key idea is to estimate an individual weight for each pixel within the support window and then aggregate the weighted costs as

$$C_{aggr}(p,d) = \sum_{q \in \omega_p} w(p,q) \cdot C_{raw}(q,d), \tag{3.10}$$

where ω_p denotes a support window centered at pixel p; q is a support pixel in ω_p.

[1] all pixels in the support window should share the same disparity value.



(a) big window (b) small window (c) different shape

Figure 3.14: Adaptive support weight approach is more flexible and powerful than simultaneously change the window size and shape. w denotes weight and 0 means the weight value is 0.

$w(p, q)$ is the weight of the support pixel q. Actually, the idea of aggregation costs using adaptive weights for each pixel had been studied in literatures [49] [50], which is a decade before the adaptive support weight approach [8].

The weight of a support pixel aims to represent the probability of this pixel locating at the same disparity as the center one. That is, the more likely it locates at the same disparity as the center pixel, the higher the weight attributed to it. Essentially, assigning a different weight for each pixel within the support window is more flexible and powerful than simultaneously changing the window size and shape, as was the case in the previous traditional methods. This argument is illustrated in Figure 3.14. In (a), a fixed window is chosen for the center pixel, where w denotes the weight of each support pixel. If the weights of outer pixels are set to 0 as shown in (b), then it is equivalent to choose a small window. If the weights of some outer pixels are set to 0 as shown in (c), then it is equivalent to choose a window with different shape. In a word, the window size and shape are changed as assigning different weight for each pixel within a window.

The weight w is extremely important for the adaptive support weight approach, since it decides the window size and shape. Various weight functions were proposed in the literatures [8] [10] [9]. In the following subsections, we describe three state-of-the-art weight functions, *i.e.* bilateral filter weight [8], geodesic weight [10], and guided filter weight [9]. The computational complexity of bilateral filter weight

and geodesic weight depend on the support window size; and thus, they are time consuming. We also introduce acceleration schemes for them [22] [21] [23]. Guided filter weight is quite efficient, whose computational complexity is independent with the window size, due to box filtering.

3.2.2.1 Bilateral Filter based Approach

The bilateral filter [51] [52] is a edge preserving operator that has found widespread use in many computer vision and graphics tasks like denoising [53] [54] [55] [56] [57], texture editing and relighting [58], tone management [59] [60], demosaicking [61], stylization [62] and optical flow estimation [63] [64].

The bilateral filter weight [8], which is based on bilateral filter, obeys two rules, *i.e.* color rules and spatial rules, and contains two corresponding terms, *i.e.* color term and spatial term.

Based on the assumption that two pixels with similar colors are more likely to locate at the same disparity, the color rule assigns a weight to a support pixel according to color similarity between this one and the center pixel. This color term can be expressed as

$$w_c(p,q) = e^{-\frac{\Delta c_{pq}}{\gamma_c}}, \tag{3.11}$$

where q is a pixel within the support window centered at the pixel p. The parameter γ_c is set by users to adjust the color similarity term. The color distance Δc_{pq} represents the Euclidean distance between the colors of p and q as

$$\Delta c_{pq} = \sqrt{\sum_{j\in(r,g,b)} (I_j(p) - I_j(q))^2}. \tag{3.12}$$

Based on the assumption that two pixels with spatial proximity are more likely to locate at the same disparity, the spatial rule assigns a weight to a support pixel according to spatial proximity between this one and the center pixel. The spatial term can be expressed as

$$w_s(p,q) = e^{-\frac{\Delta s_{pq}}{\gamma_s}}, \tag{3.13}$$

where the parameter γ_s is set to adjust the spatial distance term and the spatial

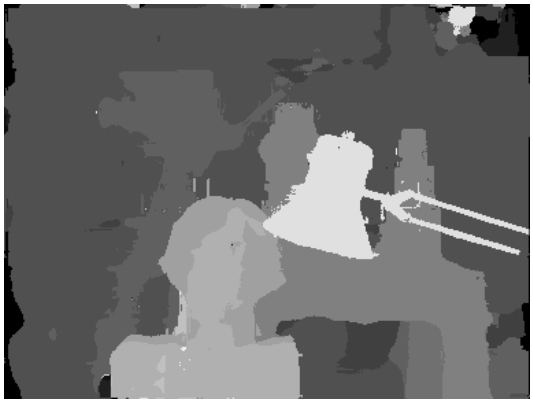


Figure 3.15: Disparity map computed by bilateral filter based ASW approach.

distance Δs_{pq} represents the Euclidean distance between the coordinates (x, y) of p and q as

$$\Delta s_{pq} = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}. \tag{3.14}$$

Bilateral filter fuse color term and spatial term by combining the respective weighting functions as [8],

$$w_{bf}(p, q) = w_c(p, q) \cdot w_s(p, q) = e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}}. \tag{3.15}$$

The disparity map computed by bilateral filter based cost aggregation is presented in Figure 3.15. It is observed that it performs well at disparity discontinuity regions and is more accurate than the previous approaches.

3.2.2.2 Acceleration Scheme for Bilateral Filter based Approach

One of the main disadvantage of the bilateral filter based approach is that it is computationally expensive, since it is required to compute a weight for all pixel in a support window. Let $(2r + 1) \times (2r + 1)$ denotes the window size and N denotes the image pixel number, then $(2r + 1) \times (2r + 1) \times N$ times weight calculation is needed for the bilateral filter based cost aggregation. The computational complexity of bilateral filter is $O(Nr^2)$, which depends on the window size. Meanwhile, the support window has to be large, *e.g.* 35×35 pixels in [8], in order to better handle untextured regions.

To improve the efficiency of bilateral filter, several acceleration methods have been proposed in literatures [65] [66] [67] [68] [69] [70] [22] [21]. S. Mattoccia *et al.* [65] proposed a Fast Bilateral Stereo method, which combines the efficiency of integral images with an adaptive support weight strategy applied on a block basis. Richardt *et al.* [66] used a bilateral grid [71] to approximate the bilateral filter. Yang *et al.* [67] implemented a piecewise-linear bilateral filtering method [72] using a recursive Gaussian filter and achieved a very low computational complexity, which is independent of the support window size. However, according to the evaluation by Hosni *et al.* [73], the above methods sacrifice quality for high computational speed and their results are not as accurate as the original bilateral filter based method [8].

Recently, two acceleration methods have been proposed which are effective but not at the expense of accuracy, *i.e.* grid graph based bilateral filter [22] and tree graph based bilateral filter [21]. The computational complexity of both these two methods are $O(N)$, which is independent of the support window size. We describe these two methods individually as follows.

Scheme I: Grid Graph based Bilateral Filer

Recursive bilateral filter based approach [22] is a kind of grid graph based bilateral filter approach. In this approach, the guidance image I is represented as a four-connected, weighted, undirected graph $G = (V, E)$, shown in Figure 3.16, in which $V = \{V_i\}$ is a set of vertices (image pixels) and $E = \{E_{ij}\}$ is a set of edges. Each edge, connecting two neighboring vertices, is mapped to a real-valued weight w_{bf} computed by bilateral filter weight function Equation (3.15), depending on the similarity of these two vertices,

$$E_{p,q} = w_{bf}(p,q) = w_c(p,q)w_s(p,q) = e^{-\frac{\Delta c_{p,q}}{\gamma_c}} e^{-\frac{1}{\gamma_s}}, \tag{3.16}$$

where p and q are two neighboring pixels, thus $w_s(p,q)$ is a constant $e^{-\frac{1}{\gamma_s}}$.

After the grid construction, cost of each pixel is aggregated by two horizontal passes and then two vertical passes. Horizontal passes are performed on each horizontal scan-line and consist of one from left to right pass and then one from right to left.

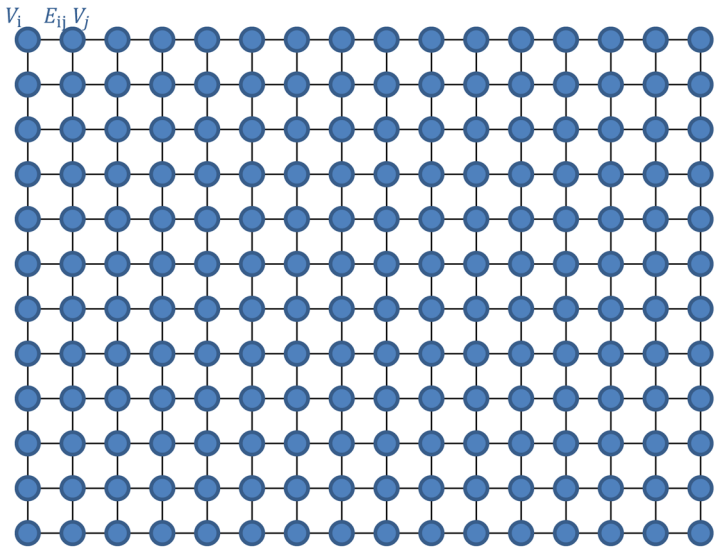


Figure 3.16: the guidance image I is represented as a four-connected, weighted, undirected graph $G = (V, E)$.

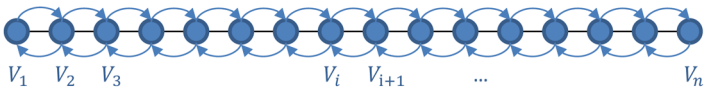


Figure 3.17: horizontal pass, the raw costs are aggregated from left to right and then from right to left.

In the first pass, from left to right pass, the aggregated cost of the first pixel is initialed to be itself,

$$C_{aggr}(V_1) = C_{raw}(V_1). \tag{3.17}$$

But, for rest pixels, the aggregated cost, *e.g.* V_i, is aggregated from its left neighbor, *e.g.* V_{i-1}, as,

$$C_{aggr}(V_i) = w_{bf}(V_{i-1}, V_i)C_{aggr}(V_{i-1}) + C_{raw}(V_i). \tag{3.18}$$

By doing so, the cost of each pixel is propagated to its right neighbor sequentially. Then in the second pass, the cost is propagated from right to left in the same way. After the above two horizontal passes, each pixel receives support from the rest pixel on the horizontal scan-line, as shown in Figure 3.17. Then, vertical passes are performed on each vertical scan-line, which is shown in Figure 3.18.

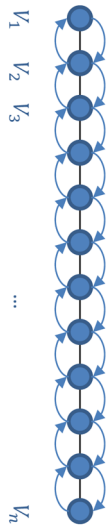


Figure 3.18: vertical pass, the raw costs are aggregated from top to bottom and then from bottom to top.

Finally, the cost of each pixel propagates to other pixels through this 4-connected grid graph and one pixel receives the support from rest pixels of the whole image. This cost aggregation strategy is a non-local cost aggregation strategy. Compared to the original bilateral filter based ASW method, this cost aggregation strategy is extremely faster, because it only needs two horizontal passes and two vertical passes. While its accuracy is even better than the original bilateral filter based ASW method, because in recursive bilateral filter method, each pixel receives support from the rest of pixels on the whole image but in original bilateral filter method, each pixel receives support from only a local window. The disparity map computed by recursive bilateral filter based ASW method is presented in Figure 3.19.

Scheme II: Tree Graph based Bilateral Filter

Extending the grid graph based bilateral filter approach, a tree graph based approach is proposed in [21], which aggregates the raw cost of each pixel on a minimum spanning tree. A spanning tree is a subgraph of the grid graph and connects all the vertices of the grid graph together. The grid graph can have many different spanning trees. In the grid graph, a weight is assigned to each edge and the sum of the weights in spanning trees can be computed. The minimum spanning tree is one of the spanning trees with weight less than that of other spanning trees.

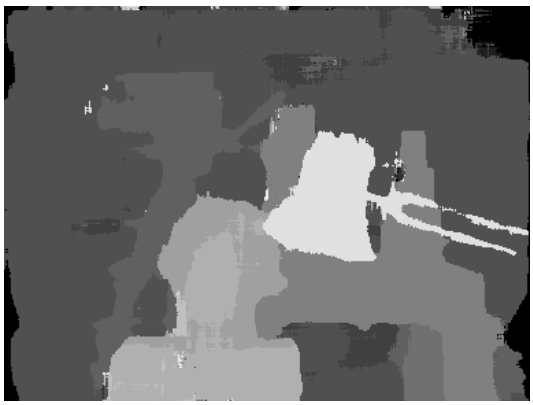
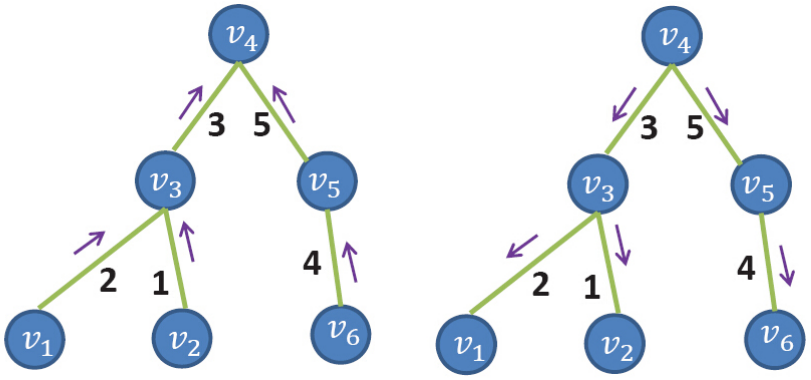


Figure 3.19: Disparity map computed by recursive bilateral filter based ASW approach.



(a) leaf to root (b) root to leaf

Figure 3.20: The cost of each pixel is aggregated from leaf to root firstly and then from root to leaf.

The first step of the tree graph based approach [21] is to construct a Minimum Spanning Tree (MST) by Kruskal's algorithm [74]. Then, the raw cost of each pixel is aggregated on the MST in two steps, *i.e.* from leaf to root and then from root to leaf, as shown in Figure 3.20 (a) and (b) respectively.

The cost aggregation from leaf to root is performed as follows. Let T_r denote a subtree of a node s and r denote the root node of T_r, then the supports node s received from this subtree is the summation of (1) the supports node s received from r and (2) $w(s,r)$ times the supports node r received from its subtrees, where $w(s,r)$ denotes the weight between nodes s and r and is calculated by bilateral filter. Let $C_d^{A\uparrow}$ denotes the aggregated cost values and $P(v_c)$ denotes parent of node v_c, then at each node $v \in V$,

$$C_d^{A\uparrow}(v) = C_d(v) + \sum_{P(v_c)=v} w(v, v_c) \cdot C_d^{A\uparrow}(v_c) \tag{3.19}$$

Note that if node v is a leaf node that has no child, then $C_d^{A\uparrow}(v) = C_d(v)$ according to Equation (3.19). After this leaf to root aggregation, the root node of MST receives supports from all nodes on MST and the rest nodes receive supports from their subtrees.

Then, cost aggregation from root to leaf is performed, in order to propagate the information of one pixel to others which are at other subtrees. The aggregated cost value $C_d^A(v)$ for any node v on a MST can be obtained from its parent $P(v)$ as follows,

$$\begin{aligned} C_d^A(v) &= C_d^{A\uparrow}(v) + w(P(v), v) \cdot [C_d^A(P(v)) - w(v, P(v)) \cdot C_d^{A\uparrow}(v)] \\ &= w(P(v), v) \cdot C_d^A(P(v)) + [1 - w^2(v, P(v))] \cdot C_d^{A\uparrow}(v)] \end{aligned} \tag{3.20}$$

The cost C_d^A obtained by Equation (3.20) is the final aggregated cost, which is tracing from the root node of MST towards its leaf nodes as shown in Figure 3.20 (b). The disparity map computed by minimum spanning tree based ASW method is presented in Figure 3.21.

In grid graph based bilateral filter approach [22] and tree graph based approach

Figure 3.21: Disparity map computed by minimum spanning tree based ASW approach.

[21], the cost of each pixel propagates to the rest through grid graph and tree graph, respectively. Therefore, these two approaches are non-local cost aggregations. Comparing to the local cost aggregation methods [8] [10] [9], the advantages of the non-local cost aggregation are, (1) the computational complexity is independent of the support window size and they are extremely fast; (2) the cost of each pixel propagating to the rest on a grid graph or tree graph is equivalent to choose a global window, the whole image, for each pixel.

3.2.2.3 Geodesic based Approach

In the geodesic based approach [10], the weight $w(p,q)$ in Equation (3.10) is inversely proportional to the geodesic distance between the center pixel p and the support pixel q as,

$$w_{geo}(p,q) = e^{-\frac{Geo(p,q)}{\gamma}} \tag{3.21}$$

where the parameter γ adjust the geodesic distance term $Geo(p,q)$, which is defined as,

$$Geo(p,q) = \min_{P \in P_{p,q}} d(P) \tag{3.22}$$

where $P_{p,q}$ denotes the set of all paths between p and q. A path P is defined as a sequence of spatially neighboring points in eight connectivity. The cost $d(P)$ of a

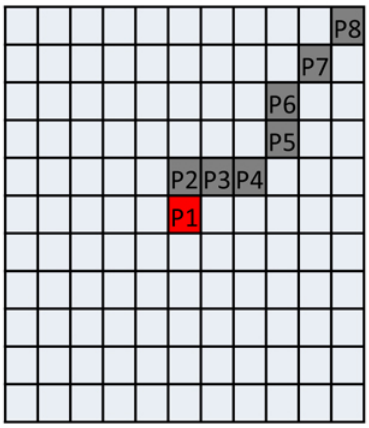


Figure 3.22: One eight-connectivity path joining p_1 and p_8. The total cost of the path is computed by summing the pixel-to-pixel Euclidean distance in the RGB color space.

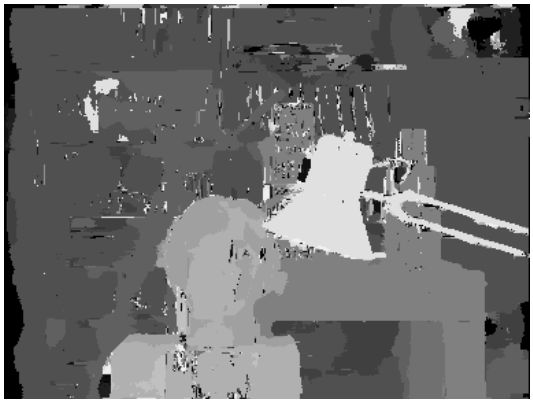


Figure 3.23: Disparity map computed by geodesic weight based ASW approach.

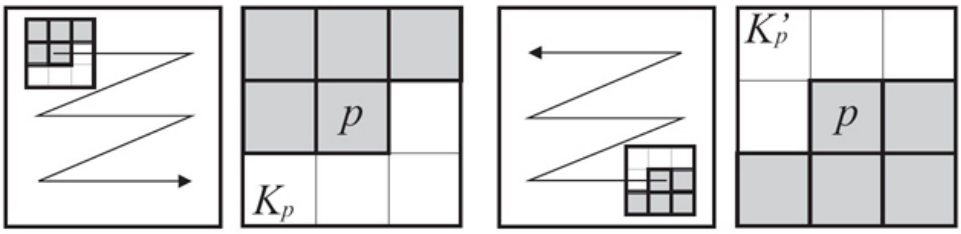
path is computed as

$$d(P) = \sum_{i=2}^{n} d_C(p_i, p_{i-1}) \tag{3.23}$$

with $d_C(p_i, p_{i-1})$ being the Euclidean distance in the RGB color space of pixel p_i and p_{i-1}. The geodesic distance is also known as shortest path, as illustrated in Figure 3.22, and can be computed by Dijkstra's algorithm [75]. The disparity map computed by geodesic weight based ASW method is presented in Figure 3.23.

Acceleration Scheme for Geodesic Weight

The geodesic based approach [10] picked out the optimal path from all candidate paths and correctly handle objects with complex outlines. The optimal path is



(a) forward pass (b) backward pass

Figure 3.24: Efficient approximation of the geodesic based method.

actually the shortest path between two pixels. Therefore, this approach needs to calculate the shortest path for each pixel, that is, $N \times (2r+1) \times (2r+1)$ times of shortest path calculation, where N denotes guidance image size and $(2r+1)$ support window size. The computational complexity of this approach is $O(Nr^2)$, which is quite time consuming. We describe two acceleration schemes for geodesic based approach.

Scheme I: distance transformation based approach

Inspired by Borgefors algorithm [76], an approximation method of geodesic based approach is proposed in [10]. The raw costs are aggregated in two row major order passes, *i.e.* forward pass and backward pass, as shown in Figure 3.24. In the forward pass, the cost of a pixel p is updated by

$$C(P) := \min_{q \in K_p} C(q) + d_C(p,q) \tag{3.24}$$

where kernel K_p is a set of pixels consisting of p itself as well as its left, left upper, upper and right upper neighbors, shown in Figure 3.24 (a). Once the forward pass is computed, the backward pass is invoked. The backward pass traverses the window in reverse direction, shown in Figure 3.24 (b). It thereby updates the costs using Equation (3.24) in conjunction with the kernel K_p', which is shown in Figure 3.24 (b). Forward and backward passes are iterated. The final costs $C(p)$ represent the estimate of the geodesic distance of p to the center pixel.

Scheme II: geodesic diffusion based approach

Anisotropic diffusion [77] is a edge-preserving technique similar to bilateral filter-

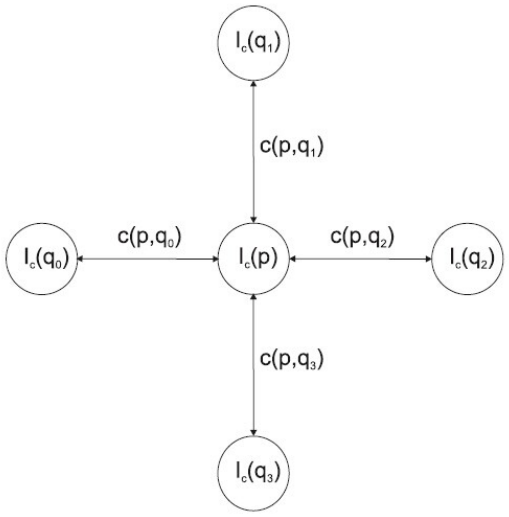


Figure 3.25: A pixel p and the four direct neighbors q used for updating the intensity value according to the diffusion coefficients $c(p, q)$.

ing [51]. The advantage of anisotropic diffusion over bilateral filtering is that, being a diffusion technique, only the comparison of each pixel with its immediate neighbors is necessary, thus saving computation time. Therefore, anisotropic diffusion is employed on stereo matching in [23] [78] [79].

The anisotropic diffusion performs as follows. Each pixel p is updated according to the intensity value of its four neighbors q_0, q_1, q_2 and q_3, as shown in Figure 3.25,

$$I_c^n(p) = I_c^{n-1}(p)(1 - \lambda \sum_{j=0}^{3} c(p, q_j)^{n-1}) + \lambda \sum_{j=0}^{3} c(p, q_j)^{n-1} I_c^{n-1}(q_j) \tag{3.25}$$

where λ controls the influence of neighboring pixels and n is the iteration number. The diffusion coefficient $c(p, q)$ can be computed from the intensity difference of p and q, which is the same as the color term in bilateral filter weight function (3.15), as

$$c(p, q) = \exp\left(-\frac{\Delta c_{pq}}{\gamma_c}\right) \tag{3.26}$$

Inspired by anisotropic diffusion approach, a geodesic diffusion based ASW approach [23] is proposed. In this approach, both costs and weights are diffused using the above anisotropic diffusion strategy in four directions. D denotes the 3D raw cost volume and D_W the 3D aggregated cost volume. D is initialized with the pixel-

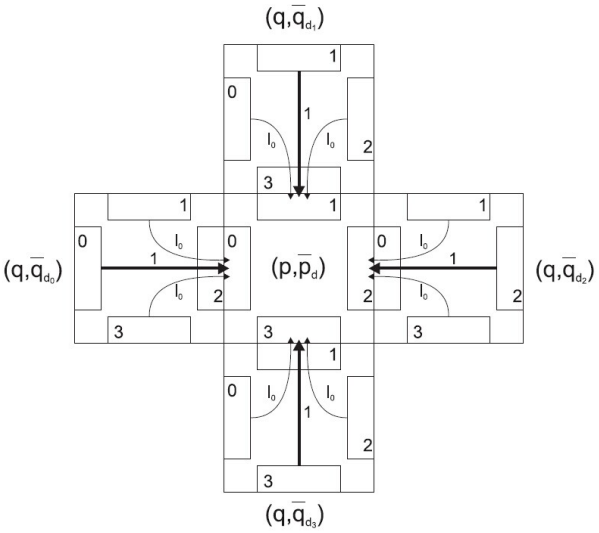


Figure 3.26: A pixel p and the four direct neighbors q used for updating the intensity value according to the diffusion coefficients $c(p, q)$.

wise raw matching cost of each pixel, while in the D_W, all positions are initialized with ones. Diffusion is performed in two other similar structures (D_d and D_{dW}), with the only difference being that they contain four data positions per pixel instead of one. The four positions refer to the four neighbors. Each of the four positions in D_d and D_{dW} are initialized with the values in the corresponding position of D and D_W, respectively. Then the cost and weight of each pixel are iteratively diffused to its four neighbors, as shown in Figure 3.26. The equations describing the diffusion process are

$$D^n_{dW}(p, \bar{p}_d, i) = w(p, q_i)w(\bar{p}_d, \bar{q}_{d_i}) \sum_{j=0}^{3} l((i-j) \mod 4) D^{n-1}_{dW}(q_i, \bar{q}_{d_i}, j) \tag{3.27}$$

$$D^n_d(p, \bar{p}_d, i) = \frac{\sum_{j=0}^{3} l((i-j) \mod 4) D^{n-1}_{dW}(q_i, \bar{q}_{d_i}, j) D^{n-1}_d(q_i, \bar{q}_{d_i}, j)}{\sum_{j=0}^{3} l((i-j) \mod 4) D^{n-1}_{dW}(q_i, \bar{q}_{d_i}, j)} \tag{3.28}$$

where i refers to the index of four neighbors, *i.e.* $i = 0$ refers to the left neighbor, $i = 1$ the upper neighbor, $i = 2$ the right neighbor and $i = 3$ the lower neighbor. The weight $w(p, q)$ are computed by Equation (3.26). Function $l(\cdot)$ are the weight

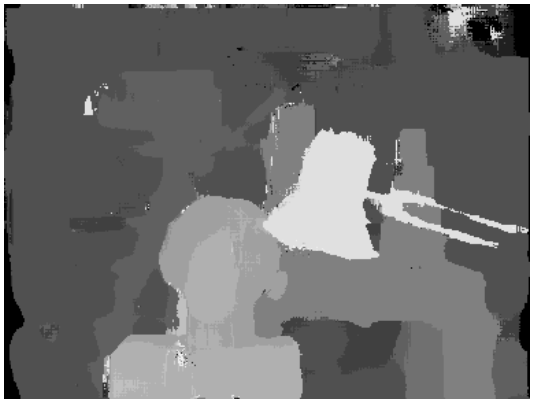


Figure 3.27: Disparity map computed by geodesic diffusion based ASW approach.

of each neighbor, defined as,

$$l(i) = \begin{cases} 1 & if \quad i = 0 \\ l_0 & if \quad i = 1 \quad or \quad 3 \\ 0 & if \quad i = 2 \end{cases} \tag{3.29}$$

After each iteration, the D_d and D_{dW} contents are accumulated in D and D_W, respectively,

$$D_W^n(p, \bar{p}_d) = D_W^{n-1}(p, \bar{p}_d) + \sum_{i=0}^{3} D_{dW}^n(p, \bar{p}_d, i), \tag{3.30}$$

$$D^n(p, \bar{p}_d) = D^{n-1}(p, \bar{p}_d) + \sum_{i=0}^{3} D_d^n(p, \bar{p}_d, i) D_{dW}^n(p, \bar{p}_d, i) \tag{3.31}$$

At the end of the diffusion process, the aggregated costs are normalized by dividing each position in D by the corresponding position in D_W. The disparity map computed by geodesic diffusion based approach is presented in Figure 3.27. This approach is implemented on GPU [23] and achieve a near real-time performance.

3.2.2.4 Guided Filter based Approach

Guided filter [80] is another edge-preserving filter. Different from the bilateral filter, the guided filter is more efficient, since it is independent of the filter size and the guided filter is proved in [80] to have better behavior near the edges. That is, guided filter outperforms bilateral filter in terms of both the accuracy and efficiency.

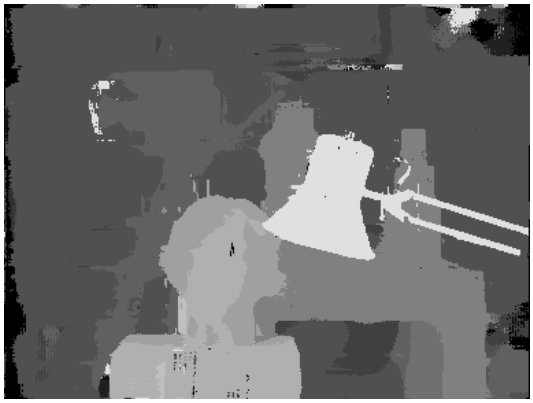


Figure 3.28: Disparity map computed by guided filter based ASW approach.

Inspired by guided filter, a guided filter based ASW approach is presented in [9].

In the case of grayscale guidance image I, the guided filter weight $w_{gf}(p,q)$ is defined as,

$$w_{gf}(p,q) = \frac{1}{|\omega|^2} \sum_{k:(p,q)\in\omega_k} \left(1 + \frac{(I_p - \mu_k)(I_q - \mu_k)}{\sigma_k^2 + \varepsilon}\right) \tag{3.32}$$

where μ_k and σ_k are the mean and the variance of guidance image I in a square window ω_k. ω denotes the number of pixels in this window and ε is a smoothness parameter.

In the case of color guidance image I, the guided filter weight is similar as,

$$w_{gf}(p,q) = \frac{1}{|\omega|^2} \sum_{k:(p,q)\in\omega_k} (1 + (I_p - \mu_k)^T(\Sigma_k + \varepsilon U)(I_q - \mu_k)) \tag{3.33}$$

here I_p, I_q and μ_k are 3×1 (color) vectors and the covariance matrix Σ_k and identity matrix U are of size 3×3.

The guided filter weight $w_{gf}(p,q)$ replaces the weight $w(p,q)$ in Equation (3.10). Note that in practice the weights $w_{gf}(p,q)$ are not computed explicitly. Instead the filtered image is obtained by running a sequence of box filters whose computational complexity is independent of the window size. Therefore, the guided filter based ASW approach is much efficient. The disparity map computed by geodesic diffusion based approach is presented in Figure 3.28. De-Maeztu *et al.* [81] also proposed a very similar guided filter-based approach for symmetrically aggregating costs.

3.3 Disparity Optimization

3.3.1 Local Optimization

The disparity optimization method used in local methods is winner-taken-all strategy [16] to select the optimal disparity $D(p)$ out of a set of candidates, which gives the minimum aggregated cost as,

$$D(p) = \arg\min_{d}(C_{aggr}(p, d)). \tag{3.34}$$

This strategy is quite simple, thus the emphasis of the local methods is on matching cost computation and cost aggregation steps.

3.3.2 Global Optimization

Global methods perform almost all of their work on the disparity optimization step and often skip the cost aggregation step. The global methods are usually formulated in an energy-minimization framework [16]. The objective is to find a disparity function d that minimizes a global energy,

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d). \tag{3.35}$$

where λ is a constant. The data term E_{data} measures how well the disparity d agrees with the input image pair, which results from the differences in intensity between corresponding pixels,

$$E_{data}(d) = \sum_{p \in I_l} C_{raw}(p, d). \tag{3.36}$$

where I_l denotes the set of pixels in the left image and the raw cost C_{raw} can be computed by the cost function described in Section 3.1. The smoothness term $E_{smooth}(d)$ makes neighboring pixels in the same image tend to have similar disparities and often restricted to only measuring the differences between neighboring

pixels's disparities. The smoothness term based on Potts model [82] is

$$E_{smooth}(d) = \sum_{p,q \in N} K \cdot T(d(p) \neq d(q)), \tag{3.37}$$

where K is the constant, N denotes a neighboring system for the pixels in the left image and function $T(\cdot)$ is 1 if its argument is true, and otherwise 0.

The optimal solution of the defined energy function Equation (3.35) can be find out through global optimization algorithms, such as iterated conditional modes [83], simulated annealing [84], graph cut [17] [85], belief propagation [18], tree-reweighted message passing [86] [87], dual decomposition Markov random field [88]. Various global optimization methods are compared and analyzed in [89].

3.4 Disparity Refinement

The disparity refinement is the last step of a stereo matching method, which is the post-processing step in order to remove mismatches and handle occlusions. This post-processing step usually performs three stages as follows. Firstly, both the left and the right disparity maps are obtained by the above three steps, *i.e.* matching cost computation, cost aggregation and disparity optimization. Secondly, invalid pixels are picked out by left-right consistency check [90], since valid pixels have the same disparity values in the left and the right disparity maps. Finally, the invalid pixels are estimated by the neighboring valid pixels. In this section, we introduce three disparity refinement methods, *i.e.* the unweighted median filtering refinement method [16], the weighted median filtering refinement method [9], and the reaggregation-based refinement method [21]. These three refinement methods differ in the final stage, invalid pixels estimation.

3.4.1 Unweighted median filtering method

In the third stage, *i.e.* invalid pixels estimation, these invalid pixels are assigned to the lowest disparity value of the spatially closest valid pixels, while lie on the same scanline. This simple invalid pixels filling strategy can generate streak-like artifacts

in the disparity map. The unweighted median filter is used in the literatures [16] to filled these invalid pixels. It replaces the value of a pixel with the median of its neighbors. For discrete signals, this median can be computed from a histogram $h(x, \cdot)$ that calculates the population around the position $x = (x, y)$,

$$h(x, i) = \sum_{x' \in \omega_x} \delta(I(x') - i) \tag{3.38}$$

where ω_x is a local window of pixel x and I is the pixel value, i is the discrete bin index, and $\delta(\cdot)$ is the Kronecker delta function, *i.e.* $\delta(\cdot)$ is 1 when the argument is 0, and is 0 otherwise. It is straightforward to pick the median value through accumulating this histogram.

The high complexity of this method becomes the timing bottleneck and sacrifices the speed of fast local cost aggregation. In order to improve its efficiency, constant time unweighted median filtering methods are proposed in [91] [92].

(x, i) denotes 3D coordinates where x represents the 2D spatial coordinates and i represents a 1D range coordinate. Define a signal $f(x, i)$ in this 3D space,

$$f(x, i) = \delta(I(x') - i) \tag{3.39}$$

Then the computation of the unweighted histogram is essentially a 2D box filtering of f in the spatial domain,

$$h(x, i) = \sum_{x' \in \omega_x} b(x, x') f(x', i) \tag{3.40}$$

where b is a box kernel. The Equation (3.40) can be simply computed by performing a 2D box filter on $f(x, i)$ for each fixed i. Since box filtering can be efficiently performed using integral images [93] [94] or moving sums in $O(1)$ time, the unweighted median filter is $O(1)$ time.

3.4.2 Weighted median filtering method

The unweighted median filtering treats each neighbor equally and blurs the disparity discontinuity boundaries. In order to preserve the disparity discontinuity regions, a edge-preserving filter, bilateral filtering, is mixed with the median filter to be the weighted median filter. The pixels are weighted in the local histograms,

$$h(x,i) = \sum_{x' \in \omega_x} w(x,x')\delta(I(x') - i) \tag{3.41}$$

where the weight is calculated by the bilateral filtering as

$$w_{x,x'} = \frac{1}{K_x} exp(-\frac{|x-x'|^2}{\sigma_s^2}) exp(-\frac{|I(x)-I(x')|^2}{\sigma_c^2}) \tag{3.42}$$

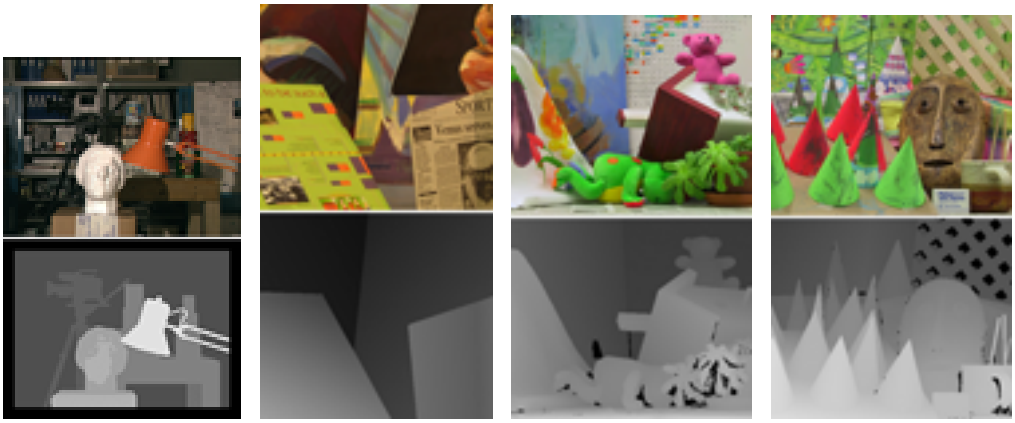
where σ_s and σ_c adjust spatial and color similarity, K is a normalization factor, and the filter dimensions is $r \times r$.

The high complexity of weighted median filtering method becomes the timing bottleneck. In order to improve its efficiency, a constant time weighted median filtering method is proposed in [95].

The constant time weighted median filtering method is similar to the constant time unweighted median filtering method described in Section 3.4.2. The box filter $b(x,x')$ in Equation (3.40) is simply replaced with any other edge-aware weight $w(x,x')$, *e.g.* bilateral filter [51], the guided filter [80], or the domain transform filter [96]. To compute the weighted histogram Equation (3.41), we only need to perform the specified edge-aware filter on $f(x,i)$ for each fixed i. If the edge aware filter is $O(1)$ time, the resulting weighted median filter is $O(1)$ time as well. Fortunately, both the guided filter [80] and the domain transform filter [96] are $O(1)$. The bilateral filter [51] is implemented in $O(1)$ in [67] [69].

3.4.3 Reaggregation-based method

The reaggregation-based method [21] is different from the weighted and unweighted median filtering methods. In the invalid pixel estimation stage, a new matching cost



(a) tsukuba (b) venus (c) teddy (d) cones

Figure 3.29: Standard four pairs provided by Middlebury Benchmark.

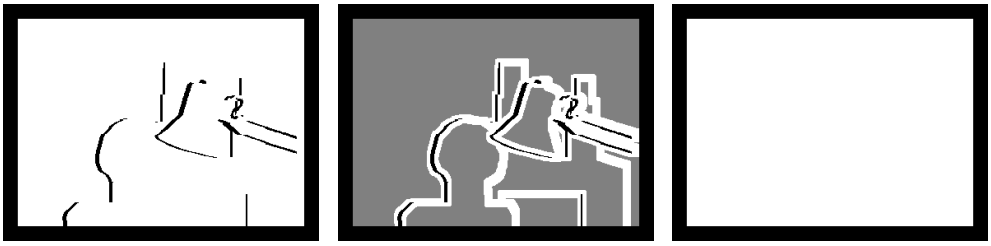
for each pixel p at each disparity level d is recomputed as,

$$C_d^{new}(p) = \begin{cases} d - D(p) & , p \; is \; stable \; and \; D(p) > 0 \\ 0 & , else \end{cases} \tag{3.43}$$

Then, the cost aggregation step is performed on this new matching cost to obtain a new aggregated cost, which is followed by the winner-take-all optimization to generate the final disparity map. This method is only used in the local stereo matching method, because cost aggregation only exist in local stereo matching methods but not in global methods.

3.5 Evaluation

The standard for stereo algorithm evaluation, widely accepted within the vision community, is the Middlebury benchmark [97]. The benchmark for stereo algorithms is done on the base of the taxonomy and quantitative evaluation of dense, two-frame stereo algorithms introduced in [16]. Four standard stereo image pairs with ground truth disparity maps are provided by the benchmark, which are presented in Figure 3.29. These stereo image pairs are preprocessed by image rectification as introduced in Section 2.3. The ground truth disparity maps are obtained by structured light technique.



(a) Non-occluded ("nocc."). (b) Discontinuity ("disc."). (c) All ("all").

Figure 3.30: Three regions evaluated on Middlebury Beachmark.

Tested stereo matching algorithms have to provide four disparity maps computed from these stereo image pairs. The computed disparity map $d_C(x, y)$ is compared with the ground truth disparity map $d_T(x, y)$ and the percentage of bad pixels with an error above a threshold δ is computed as,

$$B = \frac{1}{N} \sum_{(x,y)} (|d_C(x, y) - d_T(x, y)| > \delta) \tag{3.44}$$

where N is the total number of pixels. The evaluation of the stereo matching algorithm is done by examining the percentage of bad pixels within three regions in estimated disparity maps for all four stereo pairs, *i.e.* non-occluded regions, discontinuity regions and occluded regions. These three regions are denoted as "*nocc.*", "*disc.*", "*all*" in Table 3.1. We take the image "tsukuba" for example, which is shown in Figure 3.29 (a). Its non-occluded region is shown in Figure 3.30 (a), in which the non-occluded regions are marked in white and occluded and border regions are marked in black. The discontinuity region is shown in Figure 3.30 (b), in which the depth discontinuity regions are marked in white, occluded and border regions are marked in black and other regions are marked in gray. The all region is shown in Figure 3.30 (c), in which the all regions are marked in white and border regions are marked in black.

The overall scores of each stereo matching method are ranked within the online evaluation list [97]. Until March 2015, there are more than 150 methods are evaluated on this benchmark. This thesis focus on local stereo matching methods and we list partial related methods on Table 3.1. From this comparison, we can find

Table 3.1: The quantitative comparison on local stereo matching methods.

Algorithm	tuskuba			venus			teddy			cones			Avg.	Avg.
	nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc	Error	Runtime(ms)
MST ASW [21]	1.47	1.85	7.88	0.25	0.42	2.60	6.01	11.6	14.3	2.87	8.45	8.10	5.48	170
GeoDif ASW [23]	1.88	2.35	7.64	0.38	0.82	3.02	5.99	11.3	13.3	2.84	8.33	8.09	5.49	36646
GF ASW [9]	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	735
RBF ASW [22]	1.85	2.51	7.45	0.25	0.88	3.01	6.28	12.1	14.3	2.80	8.91	7.79	5.68	80
GEO ASW [10]	1.45	1.83	7.71	0.14	0.26	1.90	6.88	13.2	16.1	2.94	8.89	8.32	5.80	103244
BF ASW[8]	1.38	1.85	6.90	0.71	1.19	6.13	7.88	13.3	18.6	3.97	9.79	8.26	6.67	56357
variable win[43]	4.01	5.90	13.0	8.80	10.1	12.4	15.4	23.3	24.9	9.07	18.0	15.3	13.3	-
shift win[16]	5.23	7.07	24.1	3.74	5.16	11.9	16.5	24.8	32.9	10.6	19.8	26.3	15.7	-
fixed win[40]	5.13	7.11	23.2	9.18	10.3	35.4	16.9	24.5	34.0	9.94	18.9	20.8	17.9	-

that the Adaptive Support Weight (ASW) methods dramatically outperform the previous methods, *i.e.* fixed window approach, variable window approach, shiftable window approach, *etc.* The bilateral filter based method [8] is the first ASW method and is quite time consuming, since its computational complexity is depended on the window size. The minimum spanning tree (MST) [21] based method and recursive bilateral filter (RBF) [22] based method are two different acceleration schemes of the bilateral filter based method. These two methods are quite efficient, whose average runtime on the standard four stereo pairs are about 100 ms, and they are also quite accurate. Another two ASW methods, *i.e.* guided filter (GF) based method [9] and geodesic (Geo) based method [10] are proposed and display high accuracy. The guided filter based method is quite efficient (about 700ms), since it can be implemented using box filtering, whose computational complexity is independent of the window size; while the geodesic based method is as slow as the bilateral filter, thus a geodesic diffusion (GeoDif) based method [23] is proposed to speed up the geodesic based method by employing anisotropic diffusion, whose average runtime is about 36 seconds.

Besides the above four standard stereo image pairs, another 33 stereo image pairs with ground truth disparity map [98] [99] [100] are also provided, which is shown in Figure 3.31, Figure 3.32, and Figure 3.33.

(a) Map (b) Sawtooth (c) Bull (d) Poster (e) Barn1 (f) Barn2

Figure 3.31: stereo pairs provided by Middlebury Benchmark in 2001.

(a) Art (b) Books (c) Dolls (d) Laundry (e) Moebius (f) Reindeer

Figure 3.32: stereo pairs provided by Middlebury Benchmark in 2005.

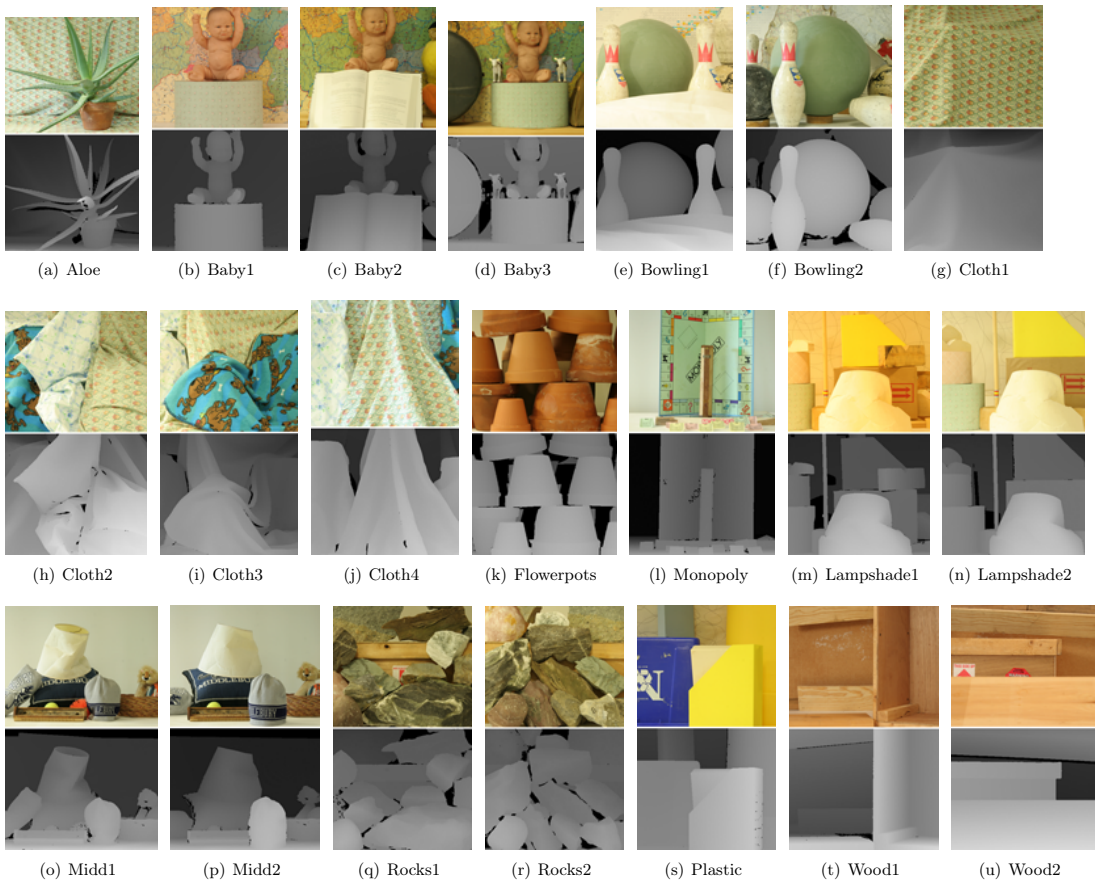


Figure 3.33: stereo pairs provided by Middlebury Benchmark in 2006.

Chapter 4

Trilateral Filter based Method

Contents

4.1 Motivation

Cost aggregation is the most important step for local stereo matching methods, since both the accuracy and efficiency of local methods largely depend on cost aggregation. As analyzed in Chapter 3, shiftable window approach [45] [42], multiple window approach [34] and variable window approach [43] adaptively change the window size and shape according the local image content. The adaptive support weight methods [8] [9] assign a weight for each pixel in a support window and the weight of a pixel represents the probability of it locating at the same disparity of the center pixel. The adaptive support weight methods are more flexible and powerful than the previous approaches, because they simultaneously change the window size and shape. Bilateral filter based method [8] and guided filter based method [9] are two outstanding adaptive support weight methods and also are state-of-the-art local stereo matching methods. However, we find that these two methods hardly sort the

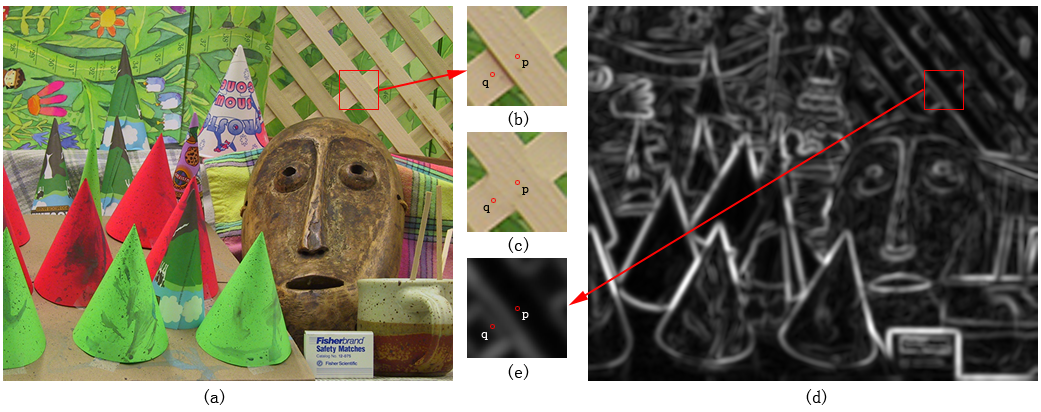
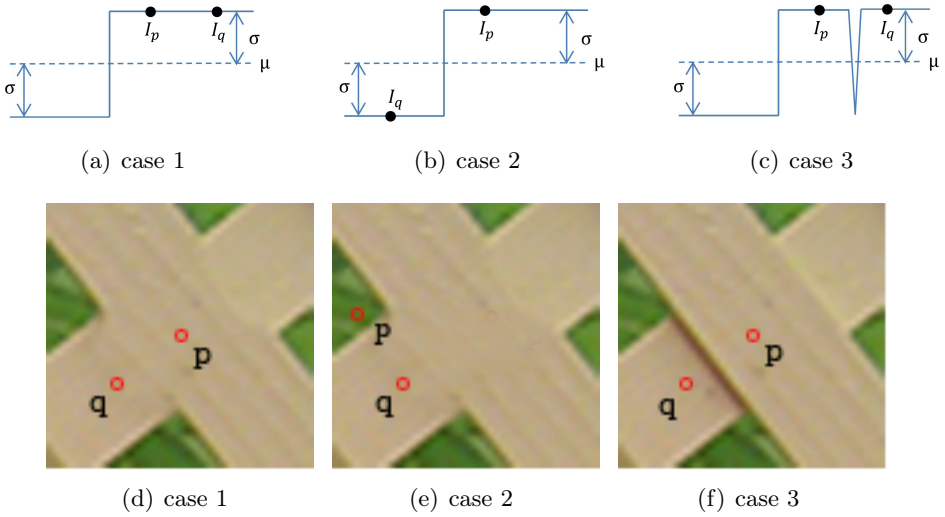


Figure 4.1: (a) Reference image. Two nearby planks with similar colors but at different disparities are shown in (b). An imaginary situation is presented in (c), these two planks are substituted by one cross-shaped plank in the same disparity. The bilateral filter weight of the pixel q in (b) is equal to that in (c), because the color similarity and spatial distance are the same as compared to the center pixel p. But these two weights should not be equal because the two pixels are at the same disparity in (c) but not in (b). The boundary cue is helpful to remedy this flaw. As shown in (d), zoomed in (e), the disparity discontinuity induces a color boundary between the pixels p and q.

ambiguity induced by nearby pixels at different disparities but with similar colors, which will be explained respectively.

Firstly, we explain the case of bilateral filter based method [8]. The assignment of an adaptive weight obeys two basic rules that the support pixel, (1) whose color is similar to the center pixel's and (2) spatially close to the center pixel, is more likely to locate at the same disparity as the center one, thus the weight should be high. Although these two simple rules can handle most depth ambiguities within a support window, they unfortunately fail to sort the disparities in the following situation as illustrated in Figure 4.1. Consider two nearby objects at different disparities but with similar colors, *e.g.*, two planks in Figure 4.1 (b). Now given two pixels at these two objects, the center p and a support q, the weight of pixel q computed by the bilateral filter weight function will be high, because of their similar colors and close positions. Whereas, this weight should be low since two pixels locate at different disparities. To further highlight this erroneous weight attribution by the bilateral filter weight function, we present an imaginary situation in (c) where those two



(a) case 1 (b) case 2 (c) case 3

(d) case 1 (e) case 2 (f) case 3

Figure 4.2: The guided filter weight function can distinguish case 1 with case 2, but fails to distinguish case 2 with case 3.

planks in (b) are substituted by a cross-shaped plank in the same disparity plane. The bilateral filter weight of the pixel q in (b) is equal to that in (c), because their color similarity and spatial distance are the same. Obviously, these two weights should not be equal, because the pixel q in (c) is at the same disparity as the center pixel p but not in (b). Thus, the bilateral filter weight fails to represent the probability of the pixel q locating at the same disparity as the center pixel p.

Guided filter based method [9] also fails to handle such ambiguity. For simplicity, we use the gray-scale guidance image I for explanation. The guided filter weight is as [9],

$$w_{gf}(p,q) = \frac{1}{|\omega|^2} \sum_{k:(p,q)\in w_k} (1 + \frac{(I_p - \mu_k)(I_q - \mu_k)}{\sigma_k^2 + \varepsilon}) \tag{4.1}$$

where μ_k and σ_k are the mean and the variance of I in a squared window ω_k with dimensions $r \times r$, centered at pixel k. $|\omega|$ denotes the number of pixels in this window and ε denotes the smoothness parameter.

Guided filter weight has edge-preserving property and is able to distinguish two pixels with different colors and at different disparities. Let us consider two cases, *i.e.* case 1 and case 2, as shown in Figure 4.2. In case 1, pixels p and q are with the same color and at the same disparity; however, in case 2, they are with different

colors and at different disparities. Theoretically, the weight of p and q should be high in case 1 and low in case 2, because they are at the same disparity in case 1 but not in case 2. Actually, according the guided filter weight in Equation (4.1), the numerator $(I_p - \mu_k)(I_q - \mu_k)$ has a positive sign in case 1, because I_j is located on the same side of the edge with I_i, as shown in Figure 4.2 (a), but a negative sign in case 2. That is, the computed weight by guided filter weight function is high in case 1 and low in case 2. Therefore, guided filter weight function can distinguish two pixels with different colors and at different disparities.

However, guided filter weight function fails to distinguish two pixels with the same color but at different disparities. Let us also consider two cases, *i.e.* case 1 and case 3. In case 1, two pixels with the same color and at the same disparity; however, in case 3, they are with the same color but at different disparities. The disparity discontinuity boundary leads to a sharp roof edge as shown in Figure 4.2 (c). The computed weight in case 1 is equal to that in case 3, because all parameters in Equation (4.1) are equivalent [1]. Therefore, guided filter weight function can not distinguish two pixels with the same color but at different disparities.

In sum, both the bilateral filter based method and guided filter based method fail to distinguish the depth ambiguity induced by nearby pixels with similar colors but at different disparities. Nevertheless, we can observe that the disparity discontinuity of the pixels p and q in Figure 4.1 (b) induces a color discontinuity between these two planks and results in color boundaries nearby, as shown in Figure 4.1 (e). Conversely, a color boundary, which can be detected by edge detection algorithm, indicates a possible disparity discontinuity. Generally speaking, color boundaries can be divided into two types, one is disparity discontinuity boundary, defined as depth boundary; the other is disparity continuity, defined as texture boundaries. In the first case: depth boundary, the support pixel is not at the same disparity with the center, defined as negative support pixel, and the support weight should be decreased; in the second case: texture boundary, the support pixel is at the same disparity with the center, defined as positive support pixel, and the support weight should not

[1] Suppose that the mean μ_k and variance σ_k are the same in these two cases. Strictly speaking, they are slightly different due to the color boundary, but the sign of the numerator $(I_p - \mu_k)(I_q - \mu_k)$ is not influenced.

(a) original image (b) four objects (c) boundary cue (d) ground truth disparity map

Figure 4.3: (a) image namely "Venus". In the right part of this image, there are four objects as indicated in (b). From this image, we have an intuition that these four objects are in four different depth, thus their contours, shown in (c), should be the depth discontinuity boundaries. But, the ground truth disparity map (d) tells us that only Obj.2 is in another depth and the other three objects are in the same depth. That is, before we know the disparity map, it is hard to distinguish depth boundaries out of texture boundaries.

be decreased. However, it is hard to distinguish a depth boundary from a texture boundary through a 2D image, unless we know its disparity map or the real scene. Take Figure 4.3 for example, in (a), we show a reference image namely "Venus", provided by Middlebury benchmark. In the right part of this image, there are four objects presented in (b). Observing this image, we have an intuition that, these four objects (from obj.1 to obj.4) are placed from near to far, in four different disparities. If these four objects are placed in different depths, their boundaries, shown in (c), should be depth boundaries. But from the ground truth disparity map (d), we can find that only Obj.1 locates in different disparity and the other three objects are in the same disparity. That means, in effect, the boundaries among the latter three objects are texture boundaries. Therefore, before we know the disparity map, it is hard for us to distinguish depth boundaries from texture boundaries.

Therefore, there are two solutions: either all color boundaries are considered as depth boundaries or as texture boundaries. We choose the first solution, no matter whether there is a texture boundary or a depth boundary between two pixels, we assign a small boundary strength weight for them. The reasons are as follows. Take Figure 4.4 as an example, if boundary l is a depth discontinuity, those pixels in the green region should be assigned small weights and we actually do that; if boundary l is not a depth discontinuity, they should be assigned high weights, but we incor-

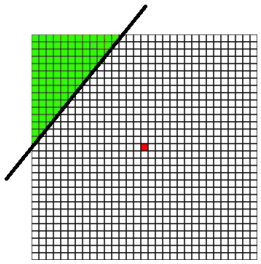


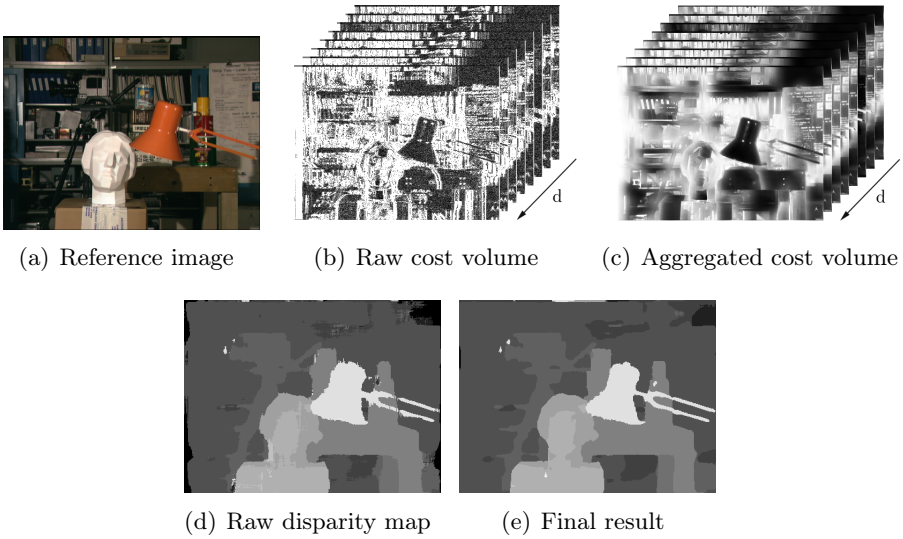
Figure 4.4: If boundary l is a depth discontinuity boundary, we correctly assign small weights for the pixels in the green region; if not, we incorrectly assign small weights for those pixels. However, aggregating a incorrect pixel is more harmful than missing a correct pixel.

rectly assign small weights to them. However, aggregating the cost of a incorrect support pixel in the second solution brings mistake information and is more harmful than missing the cost of a correct support pixel in the first solution. Therefore, if there exists a color boundary between the support pixel and the center one, they are less likely to locate at the same disparity and the support weight should be decreased. Based on it, we firstly propose a trilateral filter based weight function that extends the bilateral filter weight function by a third boundary strength term, which measures the strength of possible disparity discontinuities between two pixels. The experimental evaluation on the Middlebury benchmark demonstrates the effectiveness of the introduced boundary strength term and shows that the proposed method outperforms other local methods in terms of accuracy.

This chapter is organized as follows, in Section 4.2, we presents the framework of the proposed trilateral filter based adaptive support weight method. The trilateral filter weight function is the key part of the proposed method and this weight function is described in Section 4.3. The experimental results and analyses are given in Section 4.4, and Section 4.5 concludes this chapter.

4.2 Trilateral Filter based Method

The proposed trilateral filter based adaptive support weight method comprises the following five steps: 1) preprocessing; 2) matching cost computation; 3) cost ag-



(a) Reference image (b) Raw cost volume (c) Aggregated cost volume

(d) Raw disparity map (e) Final result

Figure 4.5: The five steps in the pipeline are described as follows. Firstly, the reference image (a) is pre-processed. Then, the raw cost volume (b) is computed from the left and right images. Thirdly, the raw cost volume is aggregated at each slice and the aggregated cost volume is shown in (c). Fourthly, the aggregated cost volume is optimized by WTA optimization to generate the raw disparity map (d). Finally, the raw disparity map is post-processed to obtain the final result (e).

gregation; 4) disparity optimization; and 5) disparity refinement. In this pipeline, we propose a trilateral filter weight function in the cost aggregation step, while we employ state-of-the-art techniques for the other four steps. These five steps are presented in Figure 4.5.

(1) Preprocessing. In order to remove isolated pixels, we apply a standard median filtering (3×3 window) to preprocess the input raw images.

(2) Matching Cost Computation. We calculate the matching cost between two pixels, using a popular matching cost function, the Truncated Absolute Difference of Color and Gradient. The raw matching cost $C_d(p)$ between a pair of candidate points, pixel p in the left image I_l and pixel $p-d$ in the right image I_r, is as in [9],

$$\begin{aligned} C_d(p) =& (1-\theta) \times min(|I_l(p) - I_r(p-d)|, \tau_1) \\ &+ \theta \times min(|\nabla_x I_l(p) - \nabla_x I_r(p-d)|, \tau_2), \end{aligned} \tag{4.2}$$

where d is the disparity and ∇_x is the derivative in the x direction; θ balances the color and derivative terms; τ_1, τ_2 are truncation values in order to reduce the influence of occluded pixels.

(3)Cost Aggregation. The ASW method is employed to aggregate the raw matching costs within a support window. Generally, two different aggregation strategies are used to compute the weight mask, the symmetric strategy, proposed in [8], and the asymmetric strategy as in (2). We chose the asymmetric strategy as suggested in [73]. The aggregated cost of a pixel p at a disparity d, $C_d^A(p)$, is as in [9, 73],

$$C_d^A(p) = \sum_{q \in \omega_p} w(p,q) \cdot C_d(q), \tag{4.3}$$

where ω_p denotes a support window centered at pixel p; q is a support pixel in ω_p. Popular ASW methods, *i.e.*, [8, 9, 10, 101], mainly differ in the form of weight function $w(p,q)$, which is the key part of this type of methods. In this paper, we introduce a novel trilateral filter weight function developed in Section 4.3.

(4)Disparity Optimization. We adopt a commonly used winner-take-all optimization strategy to select the optimal disparity $D(p)$ out of a set of candidates, which gives the minimum aggregated cost as in [16],

$$D(p) = \arg\min_d (C_d^A(p)). \tag{4.4}$$

(5)Disparity Refinement. In the final step, the generated disparity map $D(\cdot)$ is post-processed to remove mismatches and handle occlusions as in [21]. Firstly, the left and right images are used as reference images to obtain the left and right disparity maps, respectively, by performing the four steps above. Then, these two disparity maps are checked for consistency to select the stable and valid pixels. A stable pixel should pass this consistency check and a valid pixel should satisfy $D(\cdot) > 0$. A new matching cost for each pixel p at each disparity level d is recomputed as [21],

$$C_d^{new}(p) = \begin{cases} d - D(p) & , p \; is \; stable \; and \; D(p) > 0 \\ 0 & , else \end{cases} \tag{4.5}$$

Then, the cost aggregation step is performed on this new matching cost to obtain a new aggregated cost, which is followed by the disparity optimization step to generate the final disparity map.

4.3 Trilateral Filter Weight Function

We briefly describe the previous bilateral filter weight function in Section 4.3.1 and then present the proposed trilateral filter weight function in Section 4.3.2.

4.3.1 Bilateral Filter Weight Function

The bilateral filter weight function [8] obeys two important rules to measure the probability of two pixels locating at the same disparity, namely color rule and spatial rule.

Color rule. If two pixels have similar colors, they are more likely to locate at the same disparity, thus their weight should be high.

Accordingly, the color similarity term is defined as

$$w_c(p,q) = e^{-\frac{\Delta c_{pq}}{\gamma_c}}, \tag{4.6}$$

where q is a pixel within the support window centered at the pixel p. The parameter γ_c is set by users to adjust the color similarity term. The color distance Δc_{pq} represents the Euclidean distance between the colors of p and q as

$$\Delta c_{pq} = \sqrt{\sum_{j\in(r,g,b)} (I_j(p) - I_j(q))^2}. \tag{4.7}$$

Spatial rule. If two pixels are spatially close, they are more likely to locate at the same disparity, thus their weight should be high.

Accordingly, the spatial distance term is defined as

$$w_s(p,q) = e^{-\frac{\Delta s_{pq}}{\gamma_s}}, \tag{4.8}$$

where the parameter γ_s is set to adjust the spatial distance term and the spatial

distance Δs_{pq} represents the Euclidean distance between the coordinates (x, y) of p and q as

$$\Delta s_{pq} = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}. \tag{4.9}$$

Then, these two terms are combined as the final bilateral filter weight function [8],

$$w_{bf}(p, q) = w_c(p, q) \cdot w_s(p, q) = e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}}. \tag{4.10}$$

4.3.2 Trilateral Filter Weight Function

The color rule and spatial rule can solve most depth ambiguities, but unfortunately fail to assign an accurate weight to two nearby pixels at different disparities but with similar colors as analyzed in Section 4.1. As a result, we propose the following boundary rule to sort such ambiguities.

Boundary rule. If there is a boundary between two nearby pixels, they are less likely to locate at the same disparity, thus their weight should be decreased.

This rule is based on the observation that a disparity discontinuity mostly leads to a color boundary, and conversely, a color boundary suggests a potential disparity discontinuity.

According to the boundary rule, we introduce a new boundary strength term as,

$$w_e(p, q) = e^{-\frac{\Delta E_{pq}}{\gamma_e}}, \tag{4.11}$$

where the parameter γ_e is set to adjust the boundary strength term and the boundary strength distance ΔE_{pq} is calculated using a local energy model [102].

Specifically, the local energy of an image is formed as a combination of the oriented energies [103] over different orientations and the oriented energy represents the energy at a given orientation calculated via orientation selective filters. Given an image I, the local energy at a pixel p is as [102, 104],

$$E(p) = \sum_{\theta} \sqrt{(I(p) * F_{\theta,odd})^2 + (I(p) * F_{\theta,even})^2}, \tag{4.12}$$

where $*$ denotes the convolution operator; The odd-phase filter $F_{\theta,odd}$ and the even-

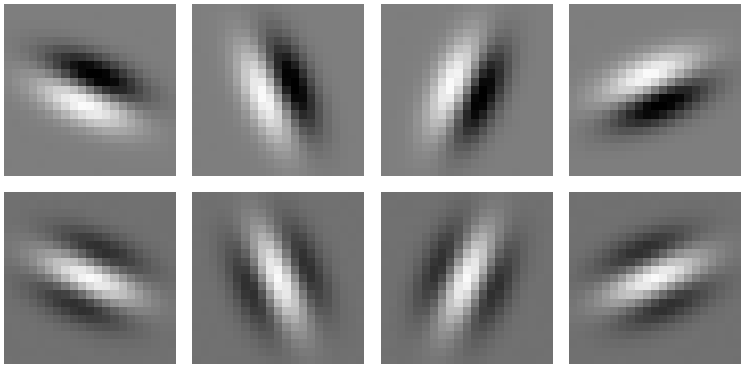


Figure 4.6: Four oriented Odd-phase filters ($\theta = 22.5°, 67.5°, 112.5°, 157.5°$) are shown in the top row and four corresponding oriented even-phase filters are shown in the bottom row.

phase filter $F_{\theta,even}$, shown in Figure 4.2, are a pair of quadrature filters at orientation θ, which is calculated by the difference of offset Gaussian functions as used in [105]. The quadrature filters can be replaced by Gabor filter [106] [107] [108], log-Gabor filters [109].

The local energy has a maximum response for the boundaries, whereas the zero-crossings of the even-phase filter $F_{\theta,even}$ locate the positions of the boundaries. The phase of a pixel p is defined as

$$\phi(p) = \begin{cases} 1 & , I(p) * F_{\theta=\theta_{max},even} > 0 \\ -1 & , I(p) * F_{\theta=\theta_{max},even} < 0 \end{cases} \tag{4.13}$$

where θ_{max} is the orientation that gives the maximum response as

$$\theta_{max} = \max_{\theta} \sqrt{(I(p) * F_{\theta,odd})^2 + (I(p) * F_{\theta,even})^2}. \tag{4.14}$$

For two neighboring pixels i and j, the distance of their boundary strength $\Delta E(i,j)$ is expressed as

$$\Delta E(i,j) = \begin{cases} E(i) + E(j) & , \phi(i) \neq \phi(j) \\ 0 & , \phi(i) = \phi(j) \end{cases} \tag{4.15}$$

For arbitrary two pixels p and q, a line linking them can be defined. Suppose

that there exist N pixels on this linking line, then the distance of their boundary strength is

$$\Delta E(p,q) = \max_{m\in[1,\cdots,N-1]} \Delta E(m, m+1). \tag{4.16}$$

where m and $m+1$ represent two neighboring pixels.

If there is no boundary between two pixels, then the boundary strength term $exp(-\Delta E_{pq}/\gamma_e)$ equals 1, because $\Delta E(p,q) = 0$; The stronger the boundary is, the smaller the boundary strength term is.

Then, we combine the proposed boundary strength term with the previous color similarity term and spatial distance term as the trilateral filter weight function $w_{tf}(p,q)$, using an empirical formula inspired by the cue combination strategy [2] proposed in [110] as,

$$w_{tf}(p,q) = e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}} + \sqrt{e^{-\frac{\Delta E_{pq}}{\gamma_e}} e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}}}. \tag{4.17}$$

The proposed trilateral filter weight function can be divided into two parts, the first part is the previous bilateral filter weight function

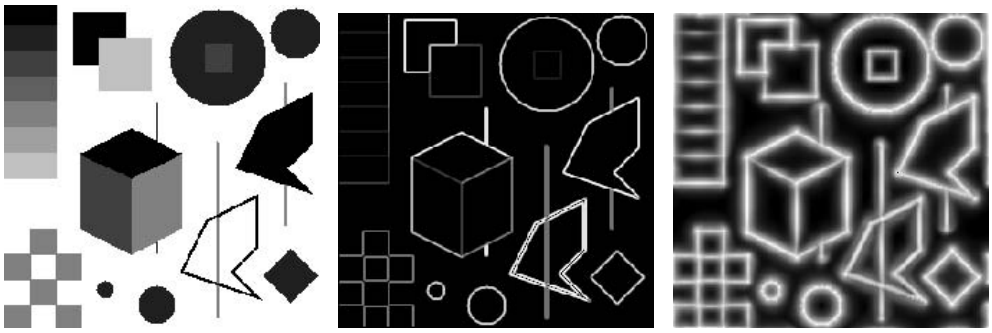
$$e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}}, \tag{4.18}$$

and the second part is the modification part, including the proposed boundary strength term

$$\sqrt{e^{-\frac{\Delta E_{pq}}{\gamma_e}} e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}}}. \tag{4.19}$$

The reason why we choose local energy model to compute the boundary strength in Equation (4.12) is as follows. Oppenheim and Lim [111], suggest that phase information is crucial to the perception of visual features, *e.g.* edges. However, the phase congruency as described in [112] is awkward to implement. Fortunately, the phase congruency is proved [113] to be proportional to local energy and the local maxima of phase congruency correspond to local maxima in local energy. The local energy is easier to obtain. Therefore, we used the local energy (phase congruency) to

[2] this cue combination strategy is compared with another combination strategy in Appendix A.



(a) Test Image (b) Output of Sobel (c) Output of local energy

Figure 4.7: Visual comparison of the Sobel operator and local energy.

compute the boundary strength, which proves to be more robust than the Sobel operator against changes in illumination and contrast. We present a visual comparison of the Sobel operator and local energy made in [112] in Figure 4.7. Figure (a) shows the test image that contains a variety of features at different contrasts, including in particular contours resulted from graded gray level changes. Figure (b) shows the output of a simple gradient-based edge detector (here, the Sobel operator). Figure (c) shows the output of the local energy (or phase congruency) detector. As it can be seen, while Sobel has missed the contours from graded gray level changes, the local energy detector shows its effectiveness on all the cases in that figure. Moreover, the authors in [114] compared the local energy (phase congruency) with Canny and Sobel using real-world driving scenes. They also concluded that local energy (phase congruency) is more robust than Canny and Sobel. It works even under different illuminations and weak image gradients.

4.4 Experimental Results

The proposed Trilateral Filter based method (TF) was implemented in C++ and evaluated on Middlebury benchmark [97], using four standard pairs of stereo images, namely "Tsukuba", "Venus", "Teddy", and "Cones". All the following experiments were conducted on a PC with a 3.4 GHz Inter Core $i7$ CPU and 8 GB memory. The experimental parameter setup is defined as follows: the parameters for matching cost computation are $\{\theta, \tau_1, \tau_2\} = \{0.11, 7, 2\}$ in Equation (4.2) and the parameters

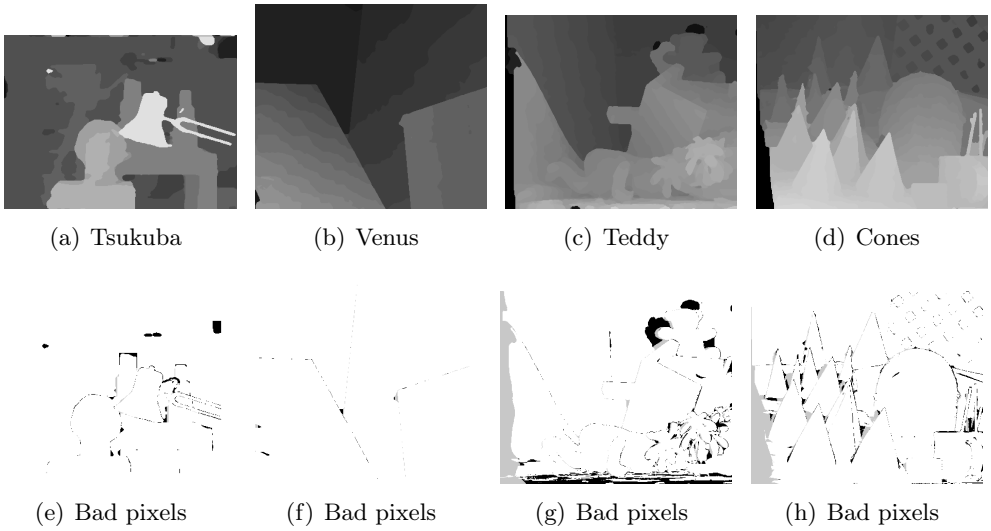
Algorithm	tuskuba			venus			teddy			cones			Avg.
	nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc	Error
Our TF	1.75	2.08	**6.51**	0.16	0.34	**1.76**	**5.99**	**11.5**	14.8	**2.46**	8.28	**6.87**	5.21
MST [21]	1.47	1.85	7.88	0.25	0.42	2.60	6.01	11.6	14.3	2.87	8.45	8.10	5.48
GeoDif [23]	1.88	2.35	7.64	0.38	0.82	3.02	5.99	11.3	**13.3**	2.84	8.33	8.09	5.49
GF [9]	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
RBF [22]	1.85	2.51	7.45	0.25	0.88	3.01	6.28	12.1	14.3	2.80	8.91	7.79	5.68
GEO [10]	1.45	1.83	7.71	**0.14**	**0.26**	1.90	6.88	13.2	16.1	2.94	8.89	8.32	5.80
BFSeg [101]	**1.25**	**1.62**	6.68	0.25	0.64	2.59	8.43	14.2	18.2	3.77	9.87	9.77	6.44
BF [8]	1.38	1.85	6.90	0.71	1.19	6.13	7.88	13.3	18.6	3.97	9.79	8.26	6.67

Table 4.1: The quantitative comparison of our trilateral filter based ASW algorithm with state-of-the-art methods on the Middlebury benchmark with error threshold 1. Our algorithm outperforms others in most columns, especially in "disc." column.

for trilateral filter weight function are $\{\gamma_c, \gamma_s, \gamma_e\} = \{0.03, 0.02, 0.02\}$. The above parameters were kept constant for all data sets.

In the first experiment, we compare the proposed trilateral filter based method, as described in Section 4.2, with state-of-the-art local stereo matching methods on Middlebury benchmark. The quantitative comparison is shown in Table 4.1. From this comparison, we find that the proposed trilateral filter based method outperforms other methods and was the most accurate local method at the time of submission (April 2013). The disparity maps of the standard four stereo pairs computed by trilateral filter based method are presented in Figure 4.8. On the Middlebury benchmark, the errors are evaluated over three different areas in the reference image, classified as non-occlusion (nocc.), discontinuous (disc.) and the entire image (all). From Table 4.1, we can observe that our proposed method outperforms these four methods, especially in the "disc." column. The term "disc." represents the regions near depth discontinuities (white areas in Figure 4.9), which mainly contain boundaries. The proposed algorithm obtains a better performance in "disc." column due to the effective boundary strength term in our weight function, which is further proved by the following comparison.

In order to demonstrate the effectiveness of our boundary strength term, we compare our trilateral filter weight function with the bilateral filter weight function particulary, since the only difference of these two functions is the boundary strength



(a) Tsukuba (b) Venus (c) Teddy (d) Cones

(e) Bad pixels (f) Bad pixels (g) Bad pixels (h) Bad pixels

Figure 4.8: Disparity maps computed by trilateral filter based method.

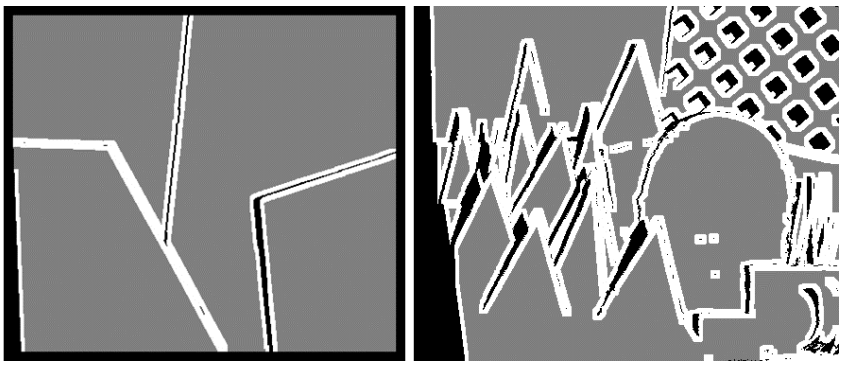


Figure 4.9: The term 'disc.' in Table 4.1 means the regions near depth discontinuities (white areas), occluded and border regions (black), and other regions (gray). In 'disc.' column, errors are only evaluated in the white areas, which mainly contain boundaries.

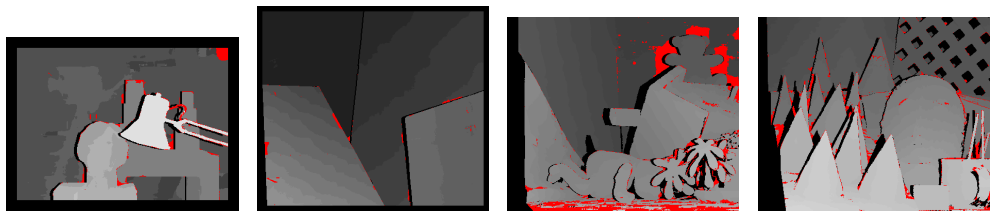
Algorithm	tuskuba			venus			teddy			cones			Avg.
	nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc	Error
Our TF	1.93	2.62	7.86	0.46	1.09	5.15	7.94	14.6	18.3	3.36	10.5	9.21	6.94
BF [8]	2.58	3.14	8.87	0.95	1.72	6.77	8.92	15.9	19.3	4.88	12.4	11.7	8.10

Table 4.2: The quantitative comparison of the trilateral filter weight function with the bilateral filter weight function on the Middlebury benchmark with error threshold 1. In order to better observe the performance of our trilateral filter cost aggregation, the pipeline of these methods are set the same, without post-processing step.

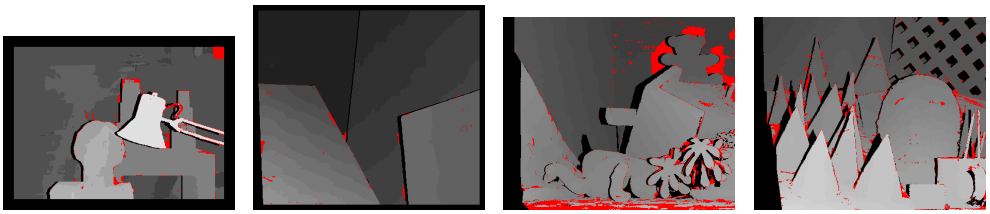
term. In this comparison, we set a fixed pipeline for these two weight functions, *i.e.* preprocessing, matching cost computation, cost aggregation and WTA optimization as described in Section 4.2, and the parameters in the matching cost computation step are set the same. We do not perform the post-processing step in order to better compare the effectiveness of two weight functions, since the post-processing step covers the performance of the cost aggregation step. The quantitative comparison is presented in Table 4.2 and the visual comparison is presented in Figure 4.10, the error pixels are marked in red. From this comparison, we can observe that the proposed trilateral filter weight function outperforms the bilateral filter weight function, which proves the effectiveness of the boundary strength term once more. In Chapter 5, the effectiveness of the trilateral filter based method will be further evaluated on a large dataset.

4.5 Conclusion

We proposed a novel trilateral filter based method by introducing a new boundary strength term. Comparing to bilateral filter based method, the trilateral filter based well handle the ambiguity caused by nearby pixels with similar color but at different disparities, which due to the boundary strength term. In the calculation of the boundary strength term, we employ the local energy model, which is proved to be more robust than the gradient-based edge detector, *e.g.* Sobel detector, canny detector, against changes in illumination and contrast. As evaluated on the Middlebury benchmark, the proposed trilateral filler based method is the most accurate



(a) Tsukuba by TF (b) Venus by TF (c) Teddy by TF (d) Cones by TF



(e) Tsukuba by BF (f) Venus by BF (g) Teddy by BF (h) Cones by BF

Figure 4.10: Disparity maps computed by trilateral filter based method.

local stereo matching method at the time of submission (April 2013).

Chapter 5

Recursive Trilateral Filter based Method

Contents

5.1 Motivation

We proposed a trilateral filter based adaptive support weight method in Chapter 4, which is more accurate than other state-of-the-art local methods at the time of submission (April 2013). However, the computational complexity of the trilateral filter based method is $O(Nr^2)$, where N denotes the image size and r denotes the support window radius. The support window has to be large, *e.g.* 35×35 in [8],

in order to better handle untextured regions. Therefore, this method is quite time consuming.

In the literatures, several methods have been proposed to speed up the bilateral filter weight function, which can be employed to accelerate the proposed trilateral filter based method. Richardt *et al.* [66] used a bilateral grid [71] to approximate the bilateral filter. Yang *et al.* [67] implemented a piecewise-linear bilateral filtering method [72] using a recursive Gaussian filter and achieved a very low computational complexity, which is independent of the support window size. However, according to the evaluation by Hosni *et al.* [73], the above methods sacrifice quality for high computational speed and their results are not as accurate as the guided filter based method [9] and bilateral filter based method [8].

Recently, two kinds of non-local cost aggregation methods have been proposed, *i.e.* grid graph based cost aggregation [22] [25] [20] and tree graph based cost aggregation [21]. In the grid graph based cost aggregation [25] [22], the cost of each pixel is aggregated horizontally (from left to right and then from right to left) and then vertically (from top to bottom and then from bottom to top). Therefore, the complicated cost aggregation is changed to 1D aggregation in four passes, two horizontal and two vertical. By doing so, the cost of each pixel propagates to other pixels through a 4-connected grid graph. In the tree graph based cost aggregation [21], a minimum spanning tree (MST) is computed from the 4-connected grid. The MST represents a global optimal path for the whole image. Then the weighted cost of each pixel is aggregated on the MST in two pass (from leaf to root and then from root to leaf). The cost of each pixel propagates to the other pixels through this tree path. Compared to the traditional local cost aggregation methods [8] [10] [9], the advantages of the non-local cost aggregation are, (1) the computational complexity is independent of the support window size and they are extremely fast; (2) the cost of each pixel propagating to the others on a grid graph or tree graph is equivalent to choose a global window (the whole image) for each pixel. Inspired by the non-local cost aggregation methods, we present in this chapter a recursive trilateral filter weight function, which dramatically improves the computational efficiency of the trilateral filter weight function (300× faster), while still providing the best accuracy

among current ASW methods, whose average error rate is 4.95% on the Middlebury benchmark.

This chapter is organized as follows. We present the proposed recursive trilateral filter based method in Section 5.2. We briefly describe the recursive filter in Section 5.2.1 and presented the recursive bilateral filter in Section 5.2.2, which are the fundamental of the proposed recursive trilateral filter. Then, the recursive trilateral filter is presented in Section 5.2.3, while its computational complexity is analyzed in Section 5.2.4. The experimental results and analyses are given in Section 5.3, and Section 5.4 concludes this Chapter.

5.2 Recursive Trilateral Filter

The pipeline of the recursive trilateral filter based method is the same as that of the trilateral filter based method, which is described in Section 4.2 in Chapter 4. The recursive trilateral filter based method also consist of five steps, 1) preprocessing; 2) matching cost computation; 3) cost aggregation; 4) disparity optimization; and 5) disparity refinement. The difference between these two methods is the cost aggregation step.

5.2.1 Recursive Filter

Recursive filter is an efficient filtering scheme for one dimensional kernels. Let x denote the one-dimensional input of a causal recursive system of order n, and y denote the output, then the general recursive system is as [115],

$$y_i = \sum_{l=0}^{n-1} a_l x_{i-l} - \sum_{k=1}^{n} b_k y_{i-k}. \tag{5.1}$$

According to the above general formula, the output of the 1^{st}-order recursive filter can be simplified as,

$$y_i = a_0 x_i + c_1 y_{i-1}. \tag{5.2}$$

where $c_1 = -b_1$.

5.2.2 Recursive Bilateral Filter

As presented in Chapter 4, the bilateral filter based cost aggregation is expressed as,

$$C_{aggr}(p,d) = \sum_{q \in \omega_p} w_{bf}(p,q) \cdot C_{raw}(q,d), \tag{5.3}$$

where the bilateral filter weight is,

$$w_{bf}(p,q) = w_c(p,q) \cdot w_s(p,q) = e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}}. \tag{5.4}$$

The recursive implementation of the bilateral filter based cost aggregation is as follows.

Let x denote a scanline of a 2D image, according to Equation (5.3) and (5.4), the 1D bilateral filtered value of x at pixel q is

$$y_q = \sum_{p=0}^{q} w_c(p,q) w_s(p,q) x_p. \tag{5.5}$$

Then, the color similarity term measuring the color distance between two pixels is redefined as [22],

$$\begin{aligned} w_c(p,q) &= w_c(p,p+1) w_c(p+1,p+2) ... w_c(q-1,q) \\ &= \prod_{i=p}^{q-1} w_c(i,i+1), \end{aligned} \tag{5.6}$$

where i and $i+1$ represent two neighboring pixels and the color distance between them as

$$w_c(i,i+1) = w_c(i+1,i) = e^{-\frac{\Delta c_{i,i+1}}{\gamma_c}}. \tag{5.7}$$

The spatial distance term is redefined in the same way as

$$w_s(p,q) = \prod_{i=p}^{q-1} w_s(i,i+1), \tag{5.8}$$

where the spatial distance between two neighboring pixels is a constant,

$$w_s(i, i+1) = w_s(i+1, i) = e^{-\frac{\Delta s_{i,i+1}}{\gamma_s}} = e^{-\frac{1}{\gamma_s}}. \tag{5.9}$$

Using the new color similarity term and spatial distance term, the 1^{st}-order recursive bilateral filter weight function is obtained with a small modification of the coefficients (a_0 and c_1) of the 1^{st}-order recursive system as [22],

$$y_i = a_0 x_i + w_c(i, i-1) c_1 y_{i-1}. \tag{5.10}$$

The output of this modified recursive system can then be proved to be [22],

$$\begin{aligned} y_i &= a_0 \sum_{k=0}^{i} w_c(i,k) c_1^{i-k} x_k \\ &= a_0 \sum_{k=0}^{i} w_c(i,k) w_s(i,k) x_k \end{aligned} \tag{5.11}$$

where the constant c_1 is actually the spatial distance between two neighboring pixels. Therefore, the recursive implementation as Equation (5.10) yields an exact bilateral filter weight function with the redefined color similarity term and the constant spatial distance term.

The final 2D bilateral filter result can then be computed by performing the above 1D recursive implementation horizontally and vertically.

5.2.3 Recursive Trilateral Filter

The proposed trilateral filter weight function

$$w_{tf}(p,q) = e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}} + \sqrt{e^{-\frac{\Delta E_{pq}}{\gamma_e}} e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}}}. \tag{5.12}$$

consists of two parts, the bilateral filter part

$$e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}}, \tag{5.13}$$

and modification part

$$\sqrt{e^{-\frac{\Delta E_{pq}}{\gamma_e}} e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}}}. \tag{5.14}$$

These two parts can be calculated separately and then added.

Let x denote a scanline, then the 1D trilateral filter value of x at pixel q is,

$$\begin{aligned} y_q = &\sum_{p=0}^{q} w_c(p,q) w_s(p,q) x_p \\ &+ \sum_{p=0}^{q} w'_c(p,q) w'_s(p,q) w'_e(p,q) x_p, \end{aligned} \tag{5.15}$$

where the parameter γ_c in $w'_c(p,q) = e^{-\frac{\Delta c_{pq}}{2\gamma_c}}$ is multiplied by 2 to take into account the square root operation in the modification part, and the parameters in $w'_s(p,q)$ and $w'_e(p,q)$ also do the same.

Inspired by the recursive implementation of the bilateral filter based cost aggregation in Section 5.3, the recursive implementation of the modification part as Equation (5.14) is achieved as follows.

We redefine the boundary strength term as,

$$\begin{aligned} w'_e(p,q) &= w'_e(p,p+1) w'_e(p+1,p+2) ... w'_e(q-1,q) \\ &= \prod_{i=p}^{q-1} w'_e(i,i+1), \end{aligned} \tag{5.16}$$

where the boundary strength distance between two neighboring pixels was defined as,

$$w'_e(i,i+1) = w'_e(i+1,i) = e^{-\frac{\Delta E_{i,i+1}}{2\gamma_e}}. \tag{5.17}$$

The color similarity term and spatial distance term are the same as defined in the recursive bilateral filter weight function; but the parameter γ_c and γ_s are now multiplied by 2 due to the square root operation in the modification part as Equation (5.14).

Using the new boundary strength term, the recursive implementation of the

modification part is defined as,

$$y_i = a_0 x_i + w'_c(i, i-1) w'_e(i, i-1) c_2 y_{i-1}. \tag{5.18}$$

This formula is proved using mathematical induction to be (the detailed proof is provided in the Appendix B),

$$y_i = a_0 \sum_{k=0}^{i} w'_c(i,k) w'_e(i,k) w'_s(i,k) x_k \tag{5.19}$$

Therefore, the recursive implementation as Equation (5.18) yields an exact modification part with the new boundary strength term.

The 2D recursive results of the bilateral filter part and modification part can then be computed by performing the above 1D recursive implementation horizontally and vertically, respectively. Then, the final trilateral filter result is achieved in adding the results of these two parts.

5.2.4 Complexity Analysis

In the proposed trilateral filter weight function, we detect the boundaries using four pairs of quadrature filters at first to calculate the boundary strength term as Equation (4.12). These filters are pre-computed and constants for all test images. The convolution operation can be divided into three steps, discrete Fourier transform (DFT), multiplication and inverse discrete Fourier transform (iDFT). The computational complexity of both DFT and iDFT is $O(Nlog_2(N))$, where N denotes the image size. Thus the computational complexity of boundary detection is $O(Nlog_2(N))$.

Then, the recursive implementation of the bilateral filter part and modification part are performed individually. The computational complexity of the bilateral filter part is $O(N)$ as analyzed in [22], and the computational complexity of the modification part is $O(N)$ as well, because the computational complexity of the extra boundary strength term calculation as Equation (4.11) and Equation (4.15) is linear to the image size N. Thus, except the boundary detection, the computational

complexity of the recursive implementation of the trilateral filter is $O(N)$, which is independent of the support window size.

5.3 Experimental Results

The proposed Recursive Trilateral Filter based method (RTF) was implemented in C++ and evaluated on both the Middlebury benchmark [97] and the real-world stereo sequence. All the following experiments were conducted on a PC with a 3.4 GHz Inter Core $i7$ CPU and 8 GB memory.

5.3.1 Experimental Settings

As described in Section 3.5, Middlebury benchmark contains not only four standard pairs of stereo images, namely "Tsukuba", "Venus", "Teddy", and "Cones", but also another 33 pairs of stereo images with ground truth disparity maps [16] [98] [99] [100]. They are listed in Figure 3.31, Figure 3.32 and Figure 3.33. Therefore, we evaluated the proposed recursive trilateral filter based method not only on the four standard pairs for comparison with state-of-the-art methods, but also on the whole dataset of 37 pairs for a more comprehensive evaluation.

The experimental parameter setup is defined as follows: the parameters for matching cost computation are $\{\theta, \tau_1, \tau_2\} = \{0.11, 7, 2\}$; parameters for recursive trilateral filter cost aggregation are $\{\gamma_c, \gamma_s, \gamma_e\} = \{0.13, 0.03, 0.05\}$. The above parameters were kept constant for all data sets.

5.3.2 Evaluation on Accuracy

According to Hosni *et al.* [73] [116] comparison, the Guided Filter based method (GF) [9] and the Bilateral Filter based method (BF)[8] outperform the Geodesic based method (Geo) [10], the Dual-Cross-Bilateral Grid based method (DCBGrid) [66], *etc.* They concluded that GF and BF are more accurate than the others. In this paper, we not only compare the proposed methods with GF [9] and BF [8], but also with four other recently proposed cost aggregation methods, the Geodesic Diffusion based method (GeoDif) [23], the Recursive Bilateral Filter based method

(RBF) [22], the Minimum Spanning Tree based method (MST) [21] and the Domain Transformation based method (DTAggr) [25], which employs a domain transform filter [96].

We carried out three experiments to compare the proposed RTF method with the state-of-the-art methods. In the first experiment, we evaluate it on the Middlebury benchmark using 37 pairs of stereo images. To better compare the cost aggregation step of each comparative method, we chose a three-step pipeline for them, *i.e.*, matching cost computation, cost aggregation and WTA optimization, without the influence of pre-processing and post-processing. Note that the raw disparity maps are directly delivered by the WTA optimization strategy, thus without any post-processing, *e.g.*, median filtering. The parameters for the matching cost computation step are set to the same values. For the cost aggregation parameters, we tested more than twenty parameter settings for each method and chose the optimal one, which leads to the lowest average error on the 37 pairs. The error metric is the same as in previous works [16] [73], *i.e.*, measuring the percentages of the erroneous pixels in the non-occlusion regions with error threshold 1[1]. Their quantitative comparison is shown in Table 5.1, while their visual comparison is presented in Figure 5.1. To better observe the raw disparity maps computed by each method, the error pixels are marked in red in Figure 5.1. This comparison is analyzed from three aspects as follows.

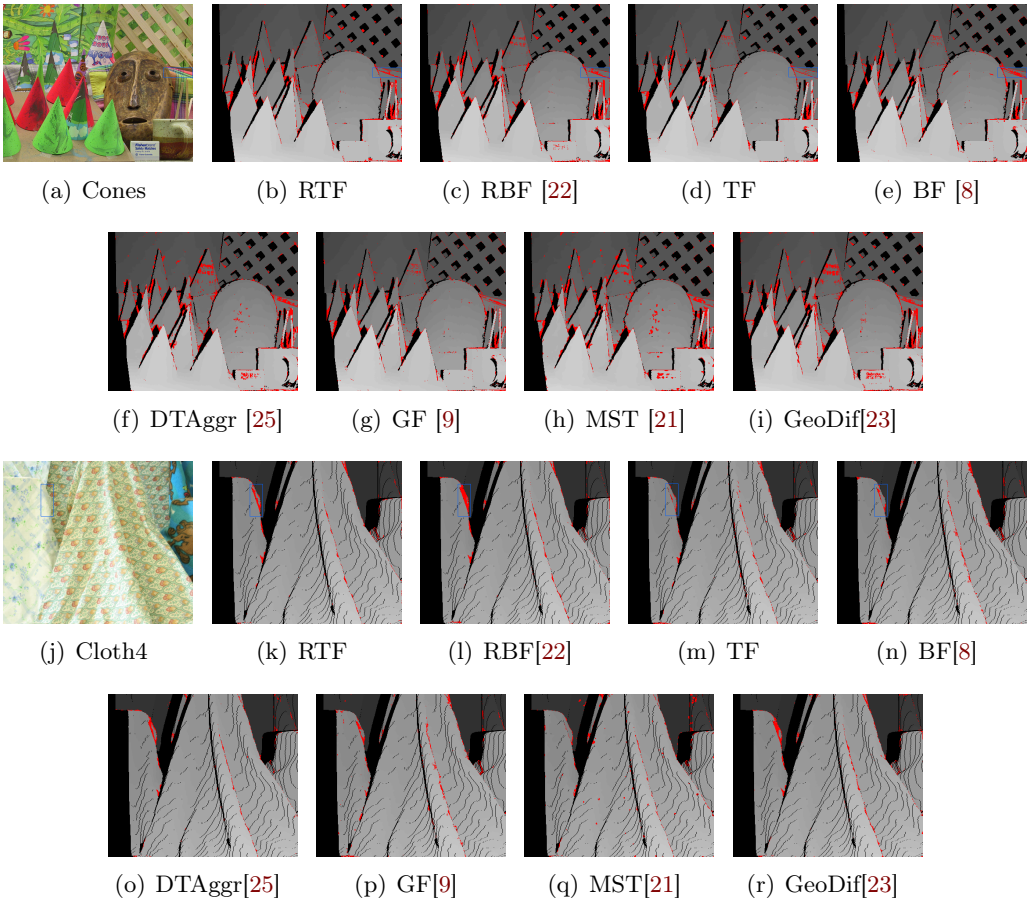
Firstly, the proposed boundary strength term is effective. From Table 5.1, we observe that RTF (TF, *resp.*) is more accurate than RBF (BF, *resp.*) in terms of average errors and also in most stereo pairs. These facts thus highlight the interest of the boundary strength term, since RTF (TF, *resp.*) differs from RBF (BF, *resp.*) only by the boundary strength term. In the visual comparison in Figure 5.1, we marked the regions containing nearby pixels with similar colors but at different disparities by a blue box. In comparing RTF (TF, *resp.*) with RBF (BF, *resp.*) in these regions in a blue box, we find that RTF (TF, *resp.*) produced less error pixels.

Secondly, we explain the difference between TF and RTF. TF is based on a

[1]The benchmark only provides the non-occlusion ground truth maps of the four standard pairs. Thus, we generated the non-occlusion ground truth maps of other pairs by left-right consistency checking on the left and right ground truth disparity maps.

Table 5.1: The quantitative comparison on the errors of raw disparity maps of 37 stereo pairs.

	Non-Local Cost Aggregation				Local Cost Aggregation			
	Our RTF	DTAggr [25]	MST[21]	RBF [22]	Our TF	BF[8]	GF[9]	GeoDif[23]
Tsukuba	2.77	3.29	2.89	3.21	2.41	2.65	2.68	3.59
Map	3.04	3.09	4.27	5.69	1.30	1.56	1.25	9.23
Sawtooth	0.79	1.08	1.56	1.35	0.61	0.80	0.91	1.82
Venus	0.56	1.13	1.39	1.21	0.74	0.71	1.25	2.11
Bull	0.24	0.36	0.33	0.33	0.21	0.22	0.25	0.44
Poster	1.22	2.11	2.31	2.61	1.07	1.11	1.75	3.50
Barn1	0.67	1.04	1.41	1.53	0.45	0.54	0.74	1.70
Barn2	0.98	1.29	1.34	1.35	1.06	1.09	1.14	1.47
Cones	3.60	3.99	4.73	4.54	3.07	3.42	3.43	4.83
Teddy	7.21	8.49	8.70	8.83	8.18	8.43	8.68	9.40
Art	8.71	9.77	8.28	10.40	8.80	8.52	8.83	9.25
Books	6.77	7.39	8.56	7.27	8.09	8.55	8.92	8.41
Dolls	4.46	4.91	3.70	5.17	4.00	4.50	4.54	5.17
Laundry	13.39	12.66	14.28	13.73	14.31	12.68	14.69	15.00
Moebius	8.81	8.61	7.53	8.77	8.71	9.10	9.24	9.21
Reindeer	5.48	5.31	7.69	5.92	5.90	5.62	6.31	5.05
Aloe	3.74	4.45	3.82	4.36	4.24	4.79	5.17	4.07
Baby1	3.71	4.33	5.62	3.99	3.87	3.66	4.25	3.87
Baby2	3.56	4.08	9.09	3.87	3.04	3.20	3.78	4.02
Baby3	4.87	5.83	5.23	5.25	5.01	5.14	5.35	5.52
Bowling1	13.65	14.42	15.40	14.49	14.67	14.54	14.98	15.64
Bowling2	6.29	7.54	6.66	7.38	6.45	6.21	6.54	6.83
Cloth1	0.23	0.19	0.43	0.20	0.53	0.51	0.98	0.20
Cloth2	2.07	2.27	1.93	2.21	2.05	2.26	2.73	2.62
Cloth3	1.60	1.43	1.50	1.48	1.51	1.63	1.85	1.70
Cloth4	1.18	1.15	1.13	1.47	1.04	1.19	1.40	1.38
Flowerpots	13.13	13.52	12.96	13.73	12.69	12.37	12.29	12.23
Lampshade1	10.09	11.40	10.00	11.00	14.08	14.41	14.21	15.95
Lampshade2	18.83	20.87	22.43	19.65	22.81	22.59	22.71	24.12
Midd1	37.79	39.77	32.08	39.50	41.46	42.01	41.93	43.26
Midd2	35.10	37.29	34.10	36.57	40.50	41.12	41.09	42.15
Monopoly	27.25	28.02	22.16	27.58	27.48	27.87	28.41	29.31
Plastic	32.11	29.86	41.48	32.07	39.07	39.49	39.34	40.78
Rocks1	1.66	1.74	1.50	1.72	1.70	2.15	2.22	1.75
Rocks2	1.47	1.40	1.08	1.54	1.22	1.40	1.33	1.46
Wood1	2.70	3.36	5.87	3.27	4.00	4.28	4.19	3.12
Wood2	1.22	1.38	0.70	1.72	1.31	1.37	1.31	1.50
Avg. Error	7.86	8.35	8.49	8.51	8.59	8.70	8.94	9.50



(a) Cones (b) RTF (c) RBF [22] (d) TF (e) BF [8]

(f) DTAggr [25] (g) GF [9] (h) MST [21] (i) GeoDif[23]

(j) Cloth4 (k) RTF (l) RBF[22] (m) TF (n) BF[8]

(o) DTAggr[25] (p) GF[9] (q) MST[21] (r) GeoDif[23]

Figure 5.1: Visual comparison with state-of-the-art methods on 37 stereo pairs. The error pixel is marked in red. The regions marked by a blue box contain nearby pixels with similar colors but at different disparities. TF and RTF produce fewer error pixels in these regions than BF and RBF, respectively. The parameter setting in each method is as follows. RTF: $\{\gamma_c, \gamma_s, \gamma_e\} = \{0.8, 0.03, 0.07\}$, RBF: $\{\gamma_c, \gamma_s\} = \{0.3, 0.05\}$, TF: $\{\gamma_c, \gamma_s, \gamma_e, r\} = \{0.08, 0.02, 0.06, 17\}$, BF: $\{\gamma_c, \gamma_s, r\} = \{30, 7, 17\}$, DTAggr: $\{\sigma_r, \sigma_s\} = \{0.12, 13\}$, GF: $\{r, \varepsilon\} = \{9, 0.0005\}$, MST: $\sigma = 0.06$, GeoDif: $\{\sigma, l, iteration\} = \{130, 0.1, 15\}$.

traditional local cost aggregation technique. The weighted costs of the support pixels within a local window are aggregated to the center. Therefore, for each center pixel, its aggregated cost is only determined by neighboring pixels within a fixed window. In the other three local cost aggregation methods, BF, GF and GeoDif, the aggregated cost of each pixel is also only influenced by neighboring pixels. However, RTF is based on the non-local cost aggregation technique, which is different from the previous local cost aggregation technique. In RTF, the guidance image is represented as a undirected graph $G = (V, E)$. The vertices V are all the image pixels, while the edges E are all the edges between the neighboring pixels. All the pixels are connected by a 4-connected grid and the weighted cost of each pixel is aggregated first horizontally (from left to right and then from right to left) and then vertically (from top to bottom and then from bottom to top). This causes the cost of each pixel to be propagated to the other pixels through a 4-connected grid path. Therefore, for each pixel, its aggregated cost is influenced by all the other pixels of the image. We conduct an experiment to change the support window sizes of TF method and the results are presented in Table 5.2. From this evaluation, we conclude that when the window size is increasing, it takes more time for cost aggregation and also we can obtain more accurate results. Compared to TF, RTF set a global window window for all images, therefore, this is the reason why RTF is more accurate than TF. The other two methods, RBF and DTAggr, are based on the same aggregation strategy, while MST is based on a similar one. In MST, a minimum spanning tree is computed from the 4-connected grid, which is a subgraph of the grid graph. The minimum spanning tree determines a global optimal path and the weighted cost of each pixel is propagated to the other pixels through this optimal tree path. Compared to the local cost aggregation technique, the advantages of the non-local cost aggregation technique are twofold. Firstly, the cost of each pixel is propagated to the whole image and eliminates the window size limitation of traditional local methods. Therefore, non-local cost aggregation methods consider more information for each pixel than local methods. From Table 5.1, we observe that non-local cost aggregation methods are usually more accurate than local cost aggregation methods. Secondly, the non-local cost aggregation methods, RTF, RBF, MST, DTAggr are

Table 5.2: Evaluation on TF method with different window size.

Window	tuskuba			venus			teddy			cones			Avg.	Avg.
Radius	nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc	Error	Time(s)
13	2.00	2.33	7.01	0.15	0.33	1.62	6.30	12.16	15.0	2.39	8.77	6.80	5.41	45.8
15	1.87	2.20	6.54	0.15	0.33	1.69	6.18	11.8	15.0	2.43	8.51	6.83	5.30	61.6
17	1.75	2.08	6.51	0.16	0.34	1.76	5.99	11.5	14.8	2.46	8.28	6.87	5.21	79.8
19	1.54	1.87	6.45	0.16	0.35	1.78	5.92	11.4	14.7	2.50	8.18	6.92	5.15	102.3
21	1.51	1.84	6.56	0.17	0.35	1.88	5.76	11.2	14.5	2.54	8.13	6.98	5.11	126.0
23	1.50	1.82	6.65	0.17	0.37	1.83	5.70	11.0	14.4	2.57	8.12	7.05	5.10	155.5
25	1.48	1.81	6.59	0.17	0.36	1.85	5.66	11.0	14.4	2.57	8.09	7.07	5.08	187.2
27	1.49	1.81	6.66	0.18	0.36	1.98	5.58	10.9	14.2	2.58	8.08	7.08	5.07	222.8

much faster than the local cost aggregation methods, TF, BF, GF, GeoDif. This will be studied in greater depth in Section 5.3.3.

Thirdly, we compare the proposed RTF method with the most related method, GeoDif [23]. In GeoDif, the weighted cost of each pixel is iteratively diffused to neighboring pixels and an optimal path (similar to geodesic distance) between two neighboring pixels is dynamically decided in each iteration. The weight in this method is calculated using Euclidean distance in the RGB color space of two neighboring pixels, which is similar to the gradient difference. However, in the proposed method, the local energy (phase congruency) detector is employed to compute the boundary strength, while the trilateral filter weight of two pixels is calculated from the boundary strength along the line connecting them. The local energy (phase congruency) detector has proved to be more robust than the gradient-based edge detector against changes in illumination and contrast. Therefore, the local energy based method is more robust than the gradient information based GeoDif method. Although GeoDif does not use gradient based edge detectors, its weight calculation only includes the gradient information. From Table 5.1, we observe that the proposed RTF method is more accurate than GeoDif. C. Pham *et al.* [25] also concluded that GeoDif is even worse than GF and BF in terms of accuracy, which is consistent with our comparison presented in Table 5.1. On the other hand, the proposed RTF is almost 10 times faster than GeoDif, their efficiencies are compared in Section 5.3.3.

[1] 8.43% refers to the error percentage on that single frame.

In the second experiment, we evaluate the proposed RTF method using the real-world stereo sequence. We chose the commonly used sequence, namely, *Book Arrival*[2], which contains 100 frames, because the ground truth disparity maps of this sequence are provided and we can quantitatively compare these methods. In this experiment, we only compare the cost aggregation step of each method without influence due to pre-processing and post-processing as in the first experiment. The parameters of the matching cost computation step are set to the same values. The optimal parameters of each cost aggregation step are selected after a large number of tests. The best result on all the 100 frames computed by each method is reported in Table 5.3 and the visual comparison is shown in Figure 5.2. The error pixels are marked in red. The quantitative comparison shows that the proposed methods perform well and that the rank of each method is almost consistent with the evaluation on 37 stereo pairs, as listed in Table 5.1. To evaluate each method on handling noise, we add the pepper noise[3] on the stereo sequence, as shown in Figure 5.2 (k). The quantitative comparison and the visual comparison are presented in Table 5.3 and Figure 5.2, respectively. As can be seen, MST seems to be the best method for dealing with pepper noise, since its average error percentage has slightly increased from 11.47% (original sequences) to 11.63% (noisy sequences). The reason is that the pepper noises, *i.e.* isolated black pixels, are virtually the leaves of the minimum spanning tree and thus influence other pixels slightly. With an average error rate increased from 11.27% to 11.70%, RTF has a similar ability to handle noise as RBF, whose average error rate has increased from 11.75% to 12.38%. The reason is that isolated noisy pepper pixels have no effect on the boundary term, which is based on local energy edge detection. Nevertheless, the overall behavior of RTF in terms of handling noise can be judged as satisfactory, because its average error rate (11.70%) is just slightly worse than that of MST (11.63%).

In the third experiment, we evaluate the proposed method with pre-processing and post-processing, and compare it with the state-of-the-art methods. The quantitative comparison is shown in Table 5.4. As can be seen from this table, RTF

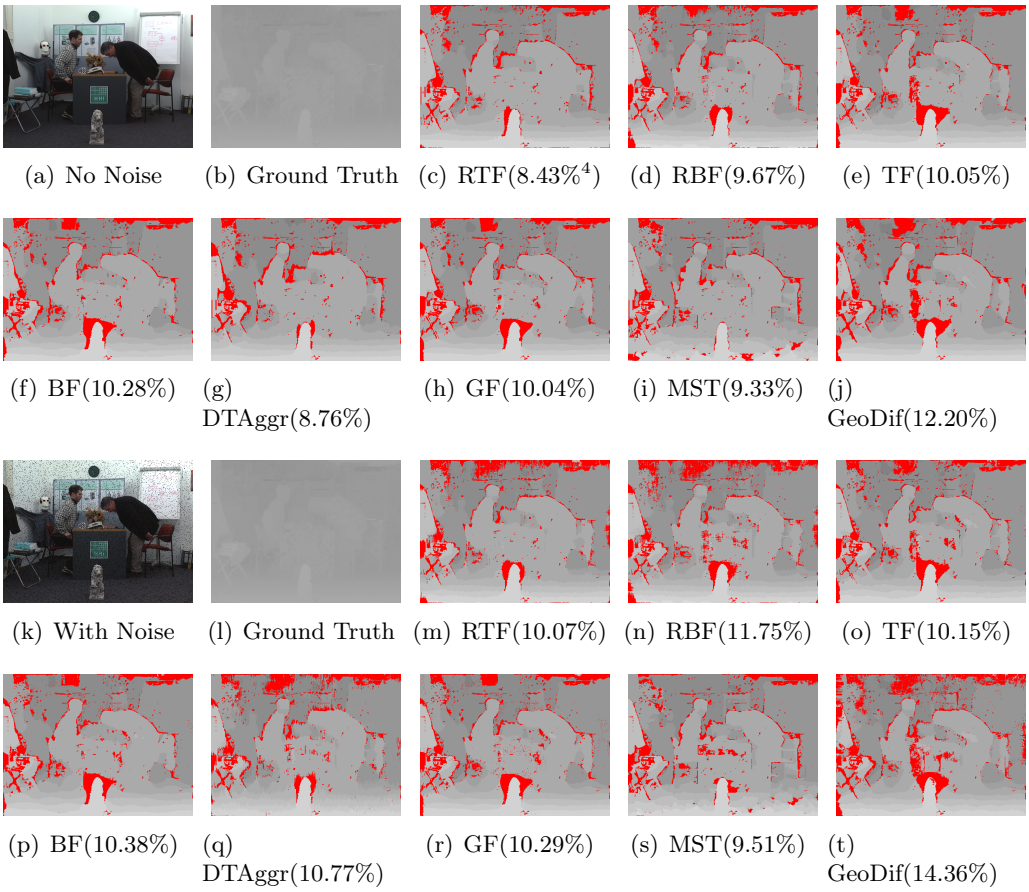
[2]Image resolution is 512×384 pixels and disparity range is 18.
[3]Coordinates of pepper noises are random and noise density is 5%.

Table 5.3: The quantitative comparison on stereo video (100 frames).

Algorithm	Average Error	
	No Noise	with Noise
Our RTF	11.27	11.70
MST[21]	11.46	11.63
DTAggr[25]	11.58	12.40
RBF[22]	11.75	12.38
Our TF	12.24	12.55
GF[116]	12.40	12.81
BF[8]	12.90	13.11
GeoDif[23]	14.03	15.60

Table 5.4: The quantitative comparison on the Middlebury benchmark with error threshold set to 1.

Algorithm	tuskuba			venus			teddy			cones			Avg.
	nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc	Error
Our RTF	1.62	2.14	**5.78**	0.21	0.50	1.86	**5.44**	**11.2**	**13.1**	**2.44**	8.18	**6.84**	4.95
Our TF	1.75	2.08	6.51	0.16	0.34	**1.76**	5.99	11.5	14.8	2.46	8.28	6.87	5.21
DTAggr [25]	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
MST [21]	1.47	1.85	7.88	0.25	0.42	2.60	6.01	11.6	14.3	2.87	8.45	8.10	5.48
GeoDif [23]	1.88	2.35	7.64	0.38	0.82	3.02	5.99	11.3	13.3	2.84	8.33	8.09	5.49
GF [9]	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
RBF [22]	1.85	2.51	7.45	0.35	0.88	3.01	6.28	12.1	14.3	2.80	8.91	7.79	5.68
GEO [10]	1.45	1.83	7.71	**0.14**	**0.26**	1.90	6.88	13.2	16.1	2.94	8.89	8.32	5.80
BFSeg [101]	**1.25**	**1.62**	6.68	0.25	0.64	2.59	8.43	14.2	18.2	3.77	9.87	9.77	6.44
BF [8]	1.38	1.85	6.90	0.71	1.19	6.13	7.88	13.3	18.6	3.97	9.79	8.26	6.67



(a) No Noise (b) Ground Truth (c) RTF(8.43%[4]) (d) RBF(9.67%) (e) TF(10.05%)

(f) BF(10.28%) (g) DTAggr(8.76%) (h) GF(10.04%) (i) MST(9.33%) (j) GeoDif(12.20%)

(k) With Noise (l) Ground Truth (m) RTF(10.07%) (n) RBF(11.75%) (o) TF(10.15%)

(p) BF(10.38%) (q) DTAggr(10.77%) (r) GF(10.29%) (s) MST(9.51%) (t) GeoDif(14.36%)

Figure 5.2: Visual comparison with state-of-the-art methods on stereo sequence. In the top two rows, the raw disparity maps are generated from the original stereo sequence, whereas in the bottom two rows, the raw disparity maps are computed from the noise stereo sequence. The parameter setting of each method is as follows. RTF: $\{\gamma_c, \gamma_s, \gamma_e\} = \{0.8, 0.1, 0.01\}$, RBF: $\{\gamma_c, \gamma_s\} = \{0.1, 0.3\}$, TF: $\{\gamma_c, \gamma_s, \gamma_e, r\} = \{0.3, 0.03, 0.03, 25\}$, BF: $\{\gamma_c, \gamma_s, r\} = \{100, 35, 25\}$, DTAggr: $\{\sigma_r, \sigma_s\} = \{0.5, 85\}$, GF: $\{r, \varepsilon\} = \{15, 0.01\}$, MST: $\sigma = 0.15$, GeoDif: $\{\sigma, l, iteration\} = \{200, 0.1, 35\}$.

Table 5.5: The evaluation on the robustness of parameter γ_c.

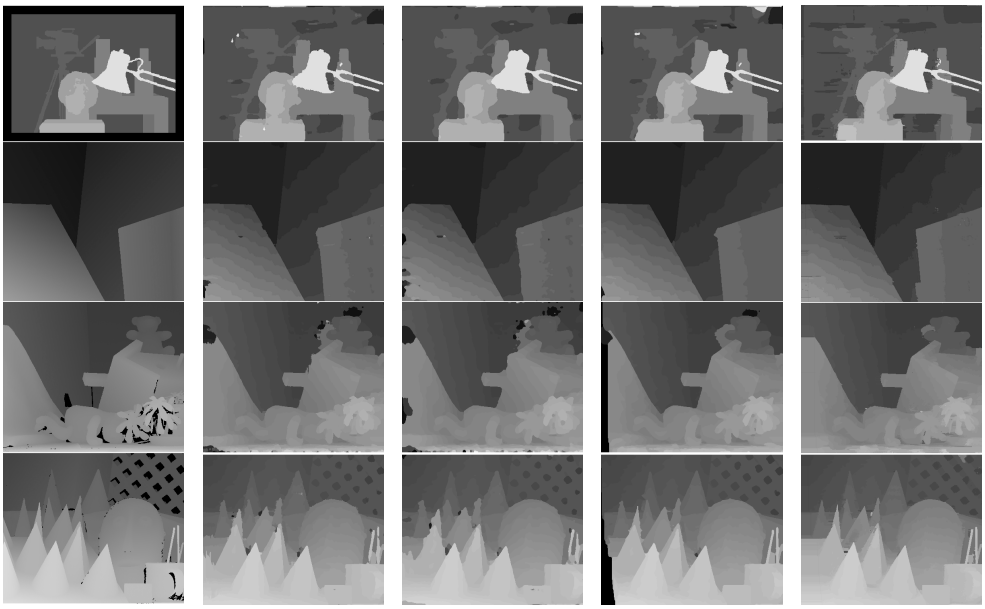
γ_c	tuskuba			venus			teddy			cones			Avg.
	nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc	Error
0.07	1.86	2.39	5.71	0.33	0.71	2.02	5.89	11.7	13.6	2.67	8.53	7.37	5.23
0.09	1.78	2.29	5.69	0.28	0.63	2.05	5.62	11.5	13.3	2.60	8.43	7.25	5.11
0.11	1.67	2.18	5.67	0.24	0.55	2.04	5.47	11.3	13.1	2.48	8.33	6.93	5.00
0.13	1.62	2.14	5.78	0.21	0.50	1.86	5.44	11.2	13.2	2.44	8.18	6.85	4.95
0.15	1.61	2.13	6.00	0.20	0.46	1.78	5.45	11.2	13.2	2.44	8.16	6.88	4.96
0.17	1.61	2.15	6.05	0.19	0.44	1.80	5.50	11.2	13.4	2.47	8.18	7.01	4.99
0.19	1.62	2.18	6.26	0.19	0.41	1.85	5.55	11.2	13.5	2.48	8.17	7.01	5.03
0.21	1.66	2.24	6.54	0.18	0.40	1.92	5.58	11.2	13.7	2.51	8.15	7.08	5.09
0.23	1.67	2.26	6.64	0.17	0.39	1.92	5.60	11.2	13.7	2.59	8.20	7.32	5.14
RBF [22]	1.85	2.51	7.45	0.35	0.88	3.01	6.28	12.1	14.3	2.80	8.91	7.79	5.68

outperforms other methods in terms of average error rate, and shows better accuracies in most columns, especially in the column named "disc.", which represents the regions near depth discontinuities mainly containing boundaries. This accuracy gain in the "disc." column gives hints as to the effectiveness of the boundary strength term. Figure 5.3 gives a visual comparison with the state-of-the-art methods along with the ground truth.

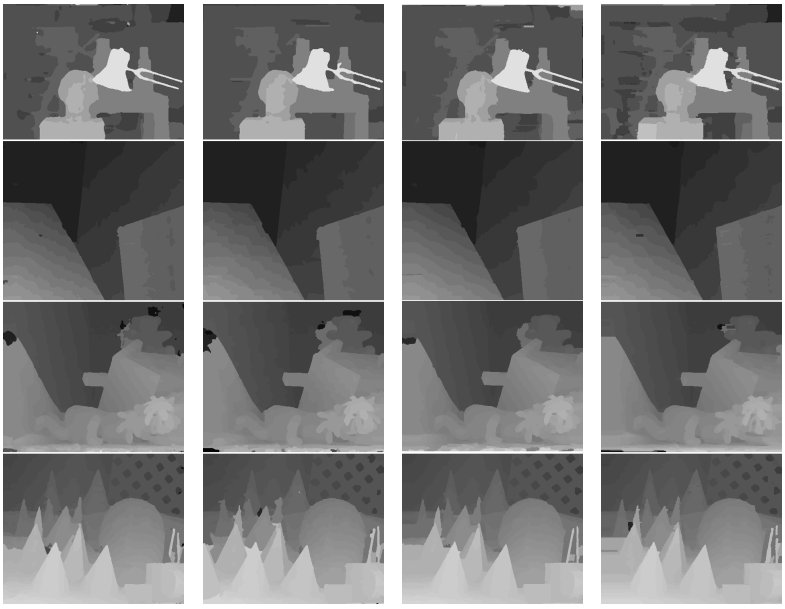
5.3.3 Discussion on Parameters for Cost Aggregation

In this section, we evaluate the robustness of the parameters for recursive trilateral filter cost aggregations on the four standard stereo pairs. The three parameters are $\{\gamma_c, \gamma_s, \gamma_e\} = \{0.13, 0.03, 0.05\}$. We conduct three experiments to individually evaluate these three parameters.

In the first experiment, we fixed two parameters $\gamma_s = 0.03$ and $\gamma_e = 0.05$ and change the value of γ_c from 0.07 to 0.23. The results are presented in Table 5.5. In the second experiment, we fixed $\gamma_c = 0.13$ and $\gamma_e = 0.05$ and change the value of γ_s from 0.02 to 0.06. The results are presented in Table 5.6. In the third experiment, we fixed $\gamma_c = 0.13$ and $\gamma_s = 0.03$ and change the value of γ_s from 0.01 to 0.09. The results are presented in Table 5.7. Observing these three experiments, we conclude that these three parameters are robust, and all these results are more accurate then RBF method.



(a) Truth (b) RTF (c) RBF (d) TF (e) BF



(f) DTAggr (g) MST (h) GF (i) GeoDif

Figure 5.3: Visual comparison with state-of-the-art methods.

Table 5.6: The evaluation on the robustness of parameter γ_s.

γ_s	tuskuba			venus			teddy			cones			Avg.
	nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc	Error
0.02	1.74	2.31	6.02	0.25	0.61	2.10	5.55	11.3	13.3	2.50	8.29	7.01	5.09
0.03	1.62	2.14	5.78	0.21	0.50	1.86	5.44	11.2	13.2	2.44	8.18	6.85	4.95
0.04	1.57	2.09	6.08	0.20	0.45	1.89	5.46	11.1	13.2	2.47	8.25	6.93	4.97
0.05	1.55	2.06	6.27	0.20	0.43	2.01	5.51	11.2	13.4	2.49	8.30	7.01	5.03
0.06	1.48	1.99	6.37	0.21	0.43	2.18	5.58	11.2	13.4	2.50	8.25	7.01	5.06
RBF [22]	1.85	2.51	7.45	0.35	0.88	3.01	6.28	12.1	14.3	2.80	8.91	7.79	5.68

Table 5.7: The evaluation on the robustness of parameter γ_e.

γ_s	tuskuba			venus			teddy			cones			Avg.
	nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc	Error
0.01	1.71	2.26	6.05	0.22	0.59	1.72	5.53	11.4	13.2	2.45	8.19	6.85	5.01
0.03	1.69	2.21	5.87	0.21	0.51	1.73	5.49	11.3	13.2	2.47	8.25	6.92	4.99
0.05	1.62	2.14	5.78	0.21	0.50	1.86	5.44	11.2	13.2	2.44	8.18	6.85	4.95
0.07	1.64	2.15	6.02	0.21	0.49	1.96	5.42	11.2	13.1	2.43	8.17	6.84	4.96
0.09	1.63	2.16	6.07	0.20	0.48	1.97	5.43	11.1	13.2	2.45	8.19	6.91	4.98
RBF [22]	1.85	2.51	7.45	0.35	0.88	3.01	6.28	12.1	14.3	2.80	8.91	7.79	5.68

5.3.4 Evaluation on Efficiency

In this section, we compare the computational efficiency of the comparative methods, *i.e.* RTF, RBF [22], MST [21], DTAggr [25],TF, GF [9], BF [8] and GeoDif [23]. These methods were all implemented in C++ and tested on the same PC using single core. We compare the runtime of the cost aggregation step of each method on the four standard pairs provided by the Middlebury benchmark, which is presented in Table 5.8. This comparison is analyzed in the following five aspects.

Firstly, the non-local cost aggregation methods RTF, RBF [22], MST [21], DTAggr [25] are extremely fast as shown in Table 5.8. The reason is as follows. The computational complexity of RTF is analyzed in Section 5.2.4, which proves to be on $O(N)$ after boundary detection operation. The computational complexity of the other three non-local methods [22] [21] [25] is also on $O(N)$, which is also independent of the support window size. Specifically speaking, cost aggregation in RTF, RBF and DTAggr needs four passes, *i.e.*, two horizontal passes and two vertical passes, while cost aggregation of MST needs two passes, *i.e.*, from leaf to root and

Table 5.8: The runtime (milliseconds) comparison on cost aggregation.

Algorithm	Tuskuba	Venus	Teddy	Cones	Avg. Time
RBF[22]	24	63	116	116	80
DTAggr[25]	43	133	244	241	165
MST[21]	83(51)	160(85)	216(92)	220(95)	170(81)
Our RTF	142(80)	241(93)	337(92)	333(92)	264(90)
GF-Gray[116]	189	577	1090	1084	735
GF-Color[116]	3087	9916	18722	18717	12610
GeoDif[23]	9602	28905	54033	54046	36646
BF[8]	22000	48358	77557	77513	56357
Our TF	45202(80)	78589(90)	97607(91)	97620(92)	79754(88)

then from root to leaf. Note that the minimum spanning tree should be constructed in MST [21] and that the average execution time of MST construction is $81ms$ as shown in Table 5.8. Therefore, these non-local cost aggregation methods are quite efficient.

Secondly, RTF ($264ms$) is almost 3 times slower than RBF ($80ms$). This can be explained by two reasons. Firstly, in contrast to RBF, the boundary detection operation is needed in RTF. The average execution time on boundary detection is about $90ms$; secondly, RTF includes one more part, *i.e.* the modification part described in Equation (5.18); thus the RTF based cost aggregation ($174ms$) is about 2 times slower than the RBF based cost aggregation ($80ms$).

Thirdly, recursive implementation dramatically improves computational efficiency. Computational complexity of TF, BF depends on support window size and these two methods are very slow. From Table 5.8, we observe that the RTF (RBF, *resp.*) is hundreds times faster than TF (BF, *resp.*), which demonstrates the effectiveness of recursive implementation.

Fourthly, RTF is more efficient than GeoDif [23], which is the method most related to our work. GeoDif needs more than twenty iterations to obtain a satisfactory result as reported in [23]. Thus, this method is not as efficient as RTF, which only needs four 1D aggregation passes.

Finally, GF is slower than the non-local cost aggregation methods RTF, RBF [22], MST [21], DTAggr [25]. Although the computational complexity of GF is

proved in [116] to be independent of window size, this cost aggregation needs 6 box filtering on each slice if guided by the gray scale image and 18 box filtering on each slice if guided by color image, according to its matlab code provided by the authors [9]. If we take "cones" as an example, GF needs $6 * 60$ box filtering [5] if guided by gray scale image and $18 * 60$ box filtering if guided by color image. This explains why GF is slower than the non-local cost aggregation methods.

5.4 Conclusion

The trilateral filter based method as presented in Chapter 4, whose computational complexity is $O(Nr^2)$, is quite time consuming. In order to improve its computational efficiency, we proposed a recursive trilateral filter based method, whose computational complexity is independent of the support window size r. Evaluated on the stereo pairs provided by middlebury benchmark, whose resolution is about $450 * 350$, the runtime of the proposed trilateral filter cost aggregation is roughly 260 milliseconds, which is 300 times faster than the brute force implementation of the trilateral filter based method. Moreover, the accuracy of the recursive trilateral filter based method still outperforms other local stereo matching methods.

[5] The maximum disparity of "cones" is 60.

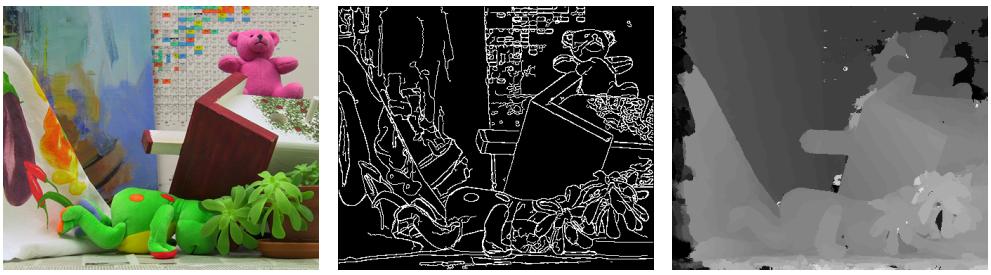
Chapter 6

Depth Edge Trilateral Filter based Method

Contents

6.1 Motivation

As presented in Chapter 4, the trilateral filter based method extends the bilateral filter based method by introducing a new boundary strength term, in order to handle the ambiguity induced by nearby pixels at different disparities but with similar colors. The boundary strength term is computed from the *color edge map*, as shown in Figure 6.1 (b), which is detected from the intensity image (Figure 6.1 (a)) by local energy edge detector. The edges on the color edge map can be divided into two types, one is disparity discontinuities, defined as depth edges; the other is disparity continuities, defined as texture edges. Actually, only depth edges are needed in the trilateral filter based method, but it is hard to distinguish depth edges from texture edges, as analyzed in Section 4.1. There are two solutions, either all color edges are



(a) intensity image (b) Color edge map (c) Disparity map by TF

Figure 6.1: (a) intensity image. (b) color edge map detected from the intensity image and used to compute the boundary strength term in the original trilateral filter based method. (c) The computed disparity map from the color edge map based trilateral filter method without disparity refinement step (post-processing step).

considered as depth edges or as texture edges. We choose the first solution, because aggregating incorrect pixels in the second solution brings mistake information and is more harmful than missing correct pixels in the first solution. The disparity map computed from the original trilateral filter based method without disparity refinement step is presented in Figure 6.1 (c).

We have an intuition that if depth edges can be found and used to calculate boundary strength term, then the accuracy of the trilateral filter based method is perhaps improved. In order to verify this intuition, we conduct an experiment using the ground truth disparity map provided by Middlebury benchmark, as shown in Figure 6.2 (a). The *ground truth depth edge map* is detected from this ground truth disparity map using edge detector, as shown in Figure 6.2 (b). This ground truth depth edge map is used to compute the boundary strength term of the trilateral filter based method and the final disparity map is present in (c).

Comparing to the disparity map computed by original TF method in Figure 6.1 (c), the disparity map computed from the ground truth depth edge map based TF method, as shown in Figure 6.2 (c), is visually more accurate. We also quantitatively compared these two methods using the standard four pairs on the Middlebury benchmark and conducted two experiments, *i.e.* one is with disparity refinement step and one is without disparity refinement step. The quantitative comparison is presented in Table 6.1, where "GT" represents the ground truth depth edge map

(a) ground truth disparity map (b) ground truth depth edge map (c) Disparity map

Figure 6.2: (a) ground truth disparity map. (b) Ground truth depth edge map detected from the ground truth disparity map and used to calculated the boundary strength term of the trilateral filter based method. (c) The computed disparity map from the ground truth depth edge map based trilateral filter method without disparity refinement step.

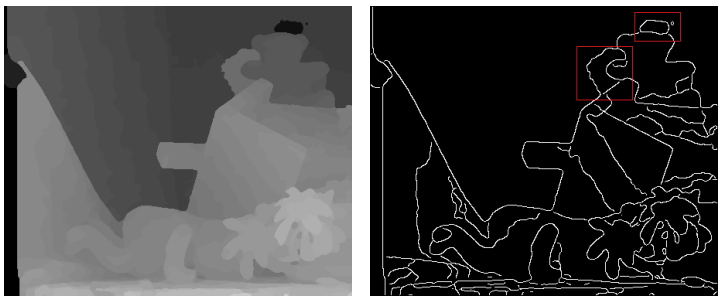
Table 6.1: The quantitative comparison on the Middlebury benchmark with error threshold set to 1.

Algorithm	tuskuba			venus			teddy			cones			Avg.
	nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc	Error
GT with refine	0.77	1.09	3.80	0.05	0.19	0.58	5.45	11.0	13.4	2.09	7.50	5.95	4.32
TF with refine	1.75	2.08	6.51	0.16	0.34	1.76	5.99	11.5	14.8	2.46	8.28	6.87	5.21
GT without refine	1.19	1.66	4.57	0.18	0.65	2.29	6.78	13.3	15.5	2.49	9.20	7.02	5.40
TF without refine	1.96	2.58	7.92	0.48	1.08	5.11	7.95	14.6	18.3	3.54	10.7	9.46	6.98

based TF method and "TF" represents the original TF method using color edge map.

From this comparison, we conclude that, when the ground truth depth edge map is used to calculate the boundary strength term, the accuracy will be highly enhanced. Motivated by this observation, we present a depth edge detection method to calculate the *depth edge map*. The experimental evaluation on the Middlebury benchmark shows that using the calculated depth edge map, the trilateral filter based method leads to a more accurate result than the original one and thereby demonstrates the effectiveness of the proposed depth edge detection method.

The remainder of this chapter is organized as follows, Section 6.2 presents the proposed depth edge detection method. The experimental results and analysis are given in Section 6.3 and Section 6.4 concludes the paper.



(a) Disparity map by TF (b) Initial depth edge map

Figure 6.3: (a) is the disparity map computed from the trilateral filter based method with disparity refinement step and (b) is the initial depth edge map computed directly from (a) the disparity map by edge detector.

6.2 Depth edge detection method

In this section, we present the proposed depth edge detection method, which makes use of two cues, the depth edge cue and the color edge cue, which are described as follows.

Depth edge cue. A disparity map, shown in Figure 6.3 (a), can be computed by the trilateral filter based method described in Section 4.2. An *initial depth edge map* can be detected from this disparity map by edge detector, which is shown in Figure 6.3 (b). This initial depth edge map contains depth edges, but the position of some depth edges are not accurate or even incorrect, marked in the red box in Figure 6.3 (b). These incorrect edges should be detected and removed.

Color edge cue. A color edge map, which is shown in Figure 6.1 (b), can be detected from the intensity image, shown in Figure 6.1 (a), by edge detector. This color edge map contains depth edges and texture edges. The position of all edges are accurate, but depth edges are mixed with texture edges and it is hard to distinguish them, which is analyzed in Section 4.1.

Our solution is to combine the color edge cue (color edge map) with the depth edge cue (initial depth edge map) in order to obtain an accurate depth edge map, which is named as *final depth edge map*. The proposed depth edge detection method consists of three parts, *i.e.* edge segmentation, edge intersection and edge linking, which are presented in the following subsections, respectively.

(a) junction pixel (b) ending pixel

Figure 6.4: The junction pixel and ending pixel are marked in red.

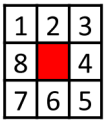


Figure 6.5: A 3×3 window is set for the center pixel. The value of pixels in the window are binary.

6.2.1 Edge segmentation

Edge segmentation [117] aims to set different edge segments with different labels. We segment both the initial depth edge map and color edge map using the following edge segmentation method.

Firstly, find junction pixels (Figure 6.4 (a)) and ending pixels (Figure 6.4 (b)) on both edge maps. To identify whether a pixel is a junction or ending, a 3×3 window is set for it as shown in Figure 6.5. Two arrays are constructed from this window as,

$$\begin{aligned} a &= [I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8] \\ b &= [I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_1] \end{aligned} \tag{6.1}$$

where I_i denotes the value of i^{th} pixel in the window. Note that both the initial depth edge map and color edge map are binary maps; thus, the value of i^{th} pixel, I_i, equals 1 if it is a edge pixel or 0 if not.

The sum of the crossing of these two arrays are computed as

$$y = \sum_{i=1}^{8} |a_i - b_i|, \tag{6.2}$$

If the result y equals 6 or 8, then the center pixel is a junction; while if it equals 2, then the center pixel is a ending.

Figure 6.6: Intersected depth edge map.

Then, the edge segmentation is developed in three steps, (1) from every ending pixel track until encounter an ending or junction and set a label for them; (2) from every junction track out on any edges that have not been labeled yet until encounter another junction, and set a label for them; (3) scan through the image looking for any unlabeled pixels, which correspond to isolated loops that have no junctions, and set a label for them.

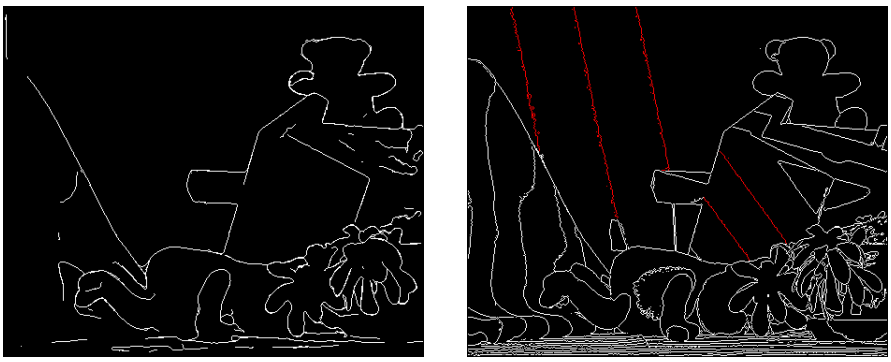
6.2.2 Edge intersection

In the edge intersection step, the initial depth edge map D_{mn}, which is shown in Figure 6.3 (b), and the color edge map C_{mn}, which is shown in Figure 6.1 (b), are logically combined to find their intersections Y_{mn}, named as *intersected edge map*, using dot production as

$$Y_{mn} = D_{mn} \cdot C_{mn}. \tag{6.3}$$

where mn denotes the image dimension.

Observing the intersected edge map Y_{mn}, which is shown in Figure 6.6, we can find that most depth edges are picked out. However, this intersection operation only compares the positions of pixels and ignore the orientation information, thus incurs some incorrect pixels. For example, a vertical edge pixel on the initial depth edge map intersects with a horizontal edge pixel at the same position on the color edge map, producing an incorrect pixel. This kind of incorrect pixels can be removed



(a) final depth edge map (b) ground truth depth edge map

Figure 6.7: (a) the detected depth edge map, which is a binary edge map; (b) the ground truth depth edge map, detected from the ground truth disparity map.

by checking their orientations as follows. For a pixel p on the intersected edge map Y_{mn}, its segment on the initial depth edge map can be found by edge segmentation technique as described in Section 6.2.1. Then, a curve is fitted to a subsegment of this segment within a local region, using third degree polynomial. The slopes of the fitted curve at pixel p can be computed by derivative and its orientation, $\theta_{depth,p}$, can be found by arctangent operation. Similarly, the orientation of p on the color edge map, $\theta_{color,p}$, can also be computed. If the difference of these two orientations beyond a threshold δ,

$$|\theta_{color,p} - \theta_{depth,p}| > \delta \tag{6.4}$$

then the pixel p is consider as a mismatch and should be excluded.

6.2.3 Edge linking

The intersected edge map is discontinuous and should be linked. In the edge linking step, we link the discontinuous edges to obtain the final depth edge map.

The ending pixels on the intersected edge map are detected by the ending pixel detection method presented in Section 6.2.1. These ending pixels are expected to be linked. For a ending pixel p, we search its nearest ending pixel q, within a local region. Then, we check p and q on the color edge map whether they are at the same segment or not. If they are, then the pixels between these two ending pixels on the color edge map are picked out to fill the holes on the intersected edge map; If not,

we keep searching another ending pixel, until we linked most edges. The final depth edge map is presented in Figure 6.7 (a).

Comparing to the ground truth depth edge map shown in Figure 6.7 (b), the final depth edge map contains most depth edges. In the ground truth depth edge map, some depth edges, *e.g.* edges marked in red color, can not be detected, because these depth edges actually caused by a continuous slanted plane in different disparities and there are no color edges corresponding to this kind of depth edges. Therefore, except this kind of depth edges, other depth edges can be detected by our depth edge detection method.

6.3 Experimental results

We implemented the proposed depth edge detection method using C++ and evaluated the depth edge based trilateral filter method on the common accepted Middlebury benchmark [97].

As described in Chapter 4, the trilateral filter based method consist of five steps as (1) preprocessing, (2) matching cost computation, (3) cost aggregation, (4) disparity optimization and (5) disparity refinement. All the experiments follow this fixed pipeline with five steps.

Ma *et. al.* [95] claimed that guided filter based cost aggregation [16] is more accurate than the simplest box-filter based cost aggregation [16], but after the disparity refinement step, the results of these two cost aggregations are almost the same. Thus, they concluded that the disparity refinement step covers the performance of the cost aggregation step. This paper focuses on depth edge detection method, which is used on the cost aggregation step. Therefore, in order to fairly evaluate the performance of the proposed method, we do not perform the disparity refinement step or any other post-processing step, *e.g.*, median filtering. In Table 6.2, we present the quantitative comparison of the proposed depth edge based trilateral filter method with two state-of-the-art cost aggregation methods, Guided Filter (GF) based method [9] and Trilateral Filter (TF) based method [118]. In this comparison, we conduct considerable tests to tune parameters for each cost aggregation

Table 6.2: The quantitative comparison of our depth edge based trilateral filter cost aggregation and two state-of-the-art cost aggregation methods on the Middlebury benchmark with error threshold set to 1. These results are computed directly from cost aggregation without disparity refinement and any other post-processing.

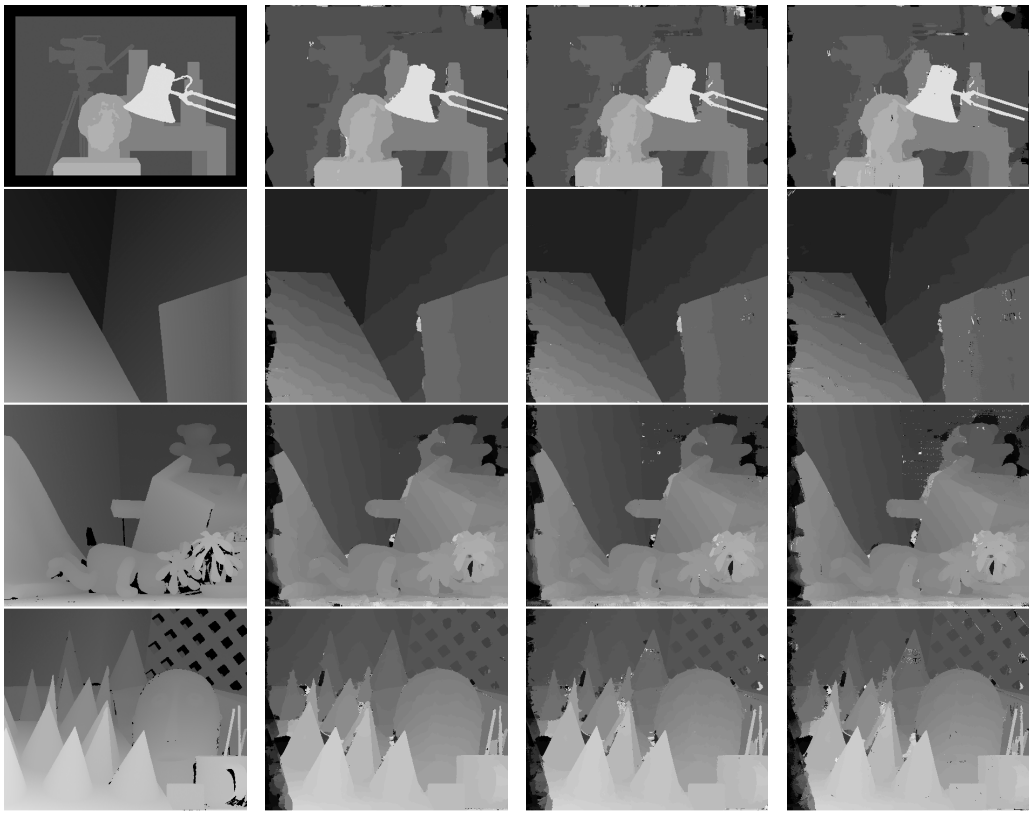
Algorithm	tuskuba			venus			teddy			cones			Avg.
	nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc	Error
Ours	1.92	2.65	7.35	0.39	1.09	4.38	7.63	14.3	17.7	3.01	10.0	8.37	6.56
TF[118]	1.96	2.58	7.92	0.48	1.08	5.11	7.95	14.6	18.3	3.54	10.7	9.46	6.98
GF[9]	2.67	3.50	9.58	1.18	2.17	11.8	8.64	15.8	18.9	3.39	11.3	9.13	8.17

Table 6.3: The quantitative comparison of the proposed depth edge based trilateral filter ASW method with other state-of-the-art methods on the Middleburry benchmark with error threshold set to 1, with refinement step.

Algorithm	tuskuba			venus			teddy			cones			Avg.
	nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc	Error
GT Depth Edge	0.77	1.09	3.80	0.05	0.19	0.58	5.45	11.0	13.4	2.09	7.50	5.95	4.32
Ours	1.57	1.99	5.88	0.13	0.38	1.49	6.04	11.7	15.0	2.39	8.27	5.70	5.13
TF[118]	1.75	2.08	6.51	0.16	0.34	1.76	5.99	11.5	14.8	2.46	8.28	6.87	5.21
GF[9]	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

and the best results are presented. The visual comparison can be found in Figure 6.8. From this comparison, it is observed that our depth edge based trilateral filter method is more accurate than the original trilateral filter method, which proves the effectiveness of the depth edge detection method; moreover, it also outperforms another state-of-the-art method, guided filter based method.

Also, we evaluate the depth edge based trilateral filter method with disparity refinement step, in order to evaluate its overall performance. The quantitative results are listed in Table 6.3, the proposed method outperforms other state-of-the-art methods. The disparity maps computed by the proposed method are presented in Figure 6.9. Importantly, the ground truth depth edge map based trilateral filter method ("GT" in Table 6.3) ranks 2^{nd} on the Middlebury benchmark and is even more accurate (error percentage is 4.32%) than most global methods. This demonstrates the potential of the proposed method and its performance can be improved by a better depth edge detection method.



(a) Ground Truth (b) Ours (c) TF (d) GF

Figure 6.8: Visual comparison the raw disparity maps generated by different cost aggregation methods. These raw disparity maps are computed from the cost aggregation step without refinement and any other post-processing.

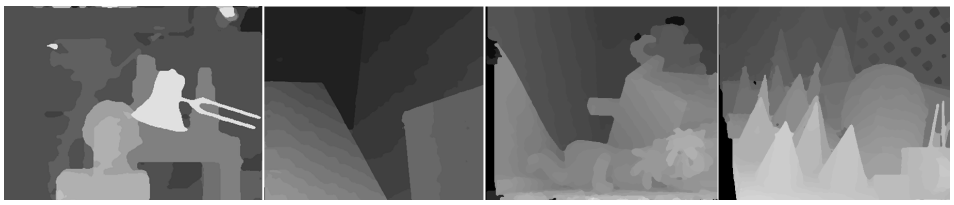


Figure 6.9: Final disparity maps computed by the proposed method.

6.4 Conclusions

The trilateral filter based method [118] is an outstanding ASW method. In this dissertation, we presented an effective depth edge detection method to obtain a depth edge map, which replaces the color edge map that used in the trilateral filter based method [118]. Evaluated on the Middlebury benchmark, we can find that the results from the proposed depth edge based trilateral filter method is more accurate than those from the original trilateral filter method, which proves the effectiveness of the proposed method. Importantly, we replaced the color edge map by the ground truth depth edge map and the average error of the final disparity map is 4.32%, which ranks 2^{nd} on the Middlebury benchmark. That means, if we can calculate a more accurate depth edge map, we can obtain more accurate results. It proves that our method has potential to outperform other stereo matching methods. In the future, the improvement of the depth edge detection method will be further studied.

Chapter 7

Conclusions and Future Work

Contents

7.1 Conclusions

This thesis mainly concentrates on improving the accuracy and efficiency of the stereo matching method. We investigated various stereo matching methods and found that the adaptive support weight method is a milestone in stereo vision field. The bilateral filter based adaptive support weight method represents the state-of-the-art local stereo matching method. However, this method fails to handle the ambiguity induced by nearby pixels at different disparities but with similar colors. In order to solve this case, we proposed a novel trilateral filer based method which extends the bilateral filter based method by introducing a boundary strength term. This is our first contribution. We evaluated the proposed trilateral filter based method on the commonly accepted Middlebury benchmark and compared it with state-of-the-art methods. The proposed method proves to be the most accurate local stereo matching method at the time of submission (April 2013).

The trilateral filter based method is quite time consuming, since its computational complexity depends on the support window size r, which is $O(Nr^2)$. In order to improve its efficiency, we proposed a recursive trilateral filter based method, inspired by the recursive filter. This is our second contribution. The proposed recursive trilateral filter based method aggregates the raw cost on a four connect-

ed grid graph by four one-dimensional cost aggregation passes. Its computational complexity proves to be $O(N)$, which is independent of support window size. The practical runtime on processing $375*450$ resolution image is roughly $260ms$ on a PC with a 3.4 GHz Inter Core i7 CPU, which is $300\times$ faster than the previous trilateral filter based method. Moreover, the proposed recursive trilateral filter based method still outperforms other local stereo matching methods in terms of accuracy.

Finally, we further studied the trilateral filter based method. The boundary strength term is the key part of this method, which is computed from a color edge map. However, we verified that, if the boundary strength term is computed from a depth edge map, the final result will be improved. Motivated by this observation, we presented a depth edge detection method to obtain an accurate depth edge map and proposed a depth edge based trilateral filter method. Evaluated on the Middlebury benchmark, the proposed depth edge based trilateral filter method outperforms the original trilateral filter method and also other local stereo matching methods.

In sum, we proposed accurate and efficient stereo matching methods, which can be used to obtain an accurate depth map in a fast time. The stereo matching problem is a basic problem in 3D reconstruction. For example, as presented in Figure 1.4 in Chapter 1, we can obtain a depth map from two related images, then we can calculate five depth maps from seven input images. Then this five depth maps are fused to obtain a high quality 3D model. In this process, the accuracy and efficiency of calculating the depth map decides the effectiveness of the final 3D model. Therefore, the proposed methods contribute to the 3D reconstruction field.

7.2 Perspectives for Future Work

We present in this section some perspectives for future research directions.

Firstly, the runtime of the recursive trilateral filter based method can be further improved by implemented on GPU. Some GPU implementations of recursive filtering have been proposed in literatures [119] [120], these works can be employed on GPU implementation of the proposed recursive trilateral filter based stereo matching method.

Secondly, the stereo pairs tested in this thesis are low resolution, about $375 * 450$ pixels. However, the image resolution captured by CCD cameras is increasing day by day. Therefore, stereo matching on high resolution stereo pairs is needed. Normally, hierarchical pyramid acceleration scheme can be employed to handle this problem. In future, we will focus on stereo matching on high resolution image pairs.

Finally, the dense stereo matching is the basic of 3D reconstruction. In future, we will pay more attention on multi-view stereo vision, structure from motion, simultaneous localization and mapping, stereo videos *etc.*

Appendix A

Weight Combination Strategy Comparison

In the proposed trilateral filter based method and recursive trilateral filter based method, we employ a weight combination strategy proposed in literatures [110] [121] [122] as,

$$w_{tf}(p,q) = e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}} + \sqrt{e^{-\frac{\Delta E_{pq}}{\gamma_e}} e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}}}. \tag{A.1}$$

This formula could be replaced by other weight combination strategies, for example

$$w_{tf}(p,q) = e^{-\frac{\Delta E_{pq}}{\gamma_e}} e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}}. \tag{A.2}$$

In fact, the Equation (A.1) is an empirical formula. We compare these two equations on Middlebury benchmark and the quantitative comparison is presented in Table A.1. From this comparison, we conclude that both Equation (A.1) and (A.2) can produce accurate results. We choose Equation (A.1) because it produces even more accurate result.

Table A.1: Weigh combination strategies comparison.

Equations	tuskuba			venus			teddy			cones			Avg.
	nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc	Error
Eq. A.1	1.62	2.14	5.78	0.21	0.50	1.86	5.44	11.2	13.1	2.44	8.18	6.84	4.95
Eq. A.2	1.66	2.19	6.06	0.28	0.57	1.96	5.50	11.2	13.1	2.55	8.36	7.14	5.05

Appendix B

Proof for Recursive Trilateral Filter

In Chapter 4, we proposed a trilateral filter weight function as,

$$w_{tf}(p,q) = e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}} + \sqrt{e^{-\frac{\Delta E_{pq}}{\gamma_e}} e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}}}. \tag{B.1}$$

which can be divided into two parts, the first part is the previous bilateral filter weight function

$$e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}}, \tag{B.2}$$

and the second part is the modification part, including the proposed boundary strength term

$$\sqrt{e^{-\frac{\Delta E_{pq}}{\gamma_e}} e^{-\frac{\Delta c_{pq}}{\gamma_c}} e^{-\frac{\Delta s_{pq}}{\gamma_s}}}. \tag{B.3}$$

The output of the recursive bilateral filter part is proved to be [22]

$$\begin{aligned} y_i &= a_0 \sum_{k=0}^{i} w_c(i,k) c_1^{i-k} x_k \\ &= a_0 \sum_{k=0}^{i} w_c(i,k) w_s(i,k) x_k \end{aligned} \tag{B.4}$$

In this appendix, we aims to prove the defined modification part,

$$y_i = a_0 x_i + w_c(i, i-1) w_e(i, i-1) c_1 y_{i-1}, \tag{B.5}$$

to be

$$y_i = a_0 \sum_{k=0}^{i} w_c(i,k) w_e(i,k) c_1^{i-k} x_k \tag{B.6}$$

using mathematical induction as in [22].

The initial condition in Equation (B.5) is,

$$y_0 = a_0 x_0, \tag{B.7}$$

when $i = 1$, from Equation (B.5), we can obtain

$$\begin{aligned} y_1 &= a_0 x_1 + w_c(1,0) w_e(1,0) c_1 y_0, \\ &= a_0 (x_1 + w_c(1,0) w_e(1,0) c_1 x_0), \\ &= a_0 \sum_{k=0}^{1} w_c(1,k) w_e(1,k) c_1^{1-k} x_k. \end{aligned} \tag{B.8}$$

Thus, Equation (B.6) is true when $i = 1$.

Induction: assume that Equation (B.6) for i from 1 up through j are all true. We need to show that Equation (B.6) is true for $i = j + 1$.

According to Equation (B.5),

$$y_{j+1} = a_0 x_{j+1} + w_c(j+1,j) w_e(j+1,j) c_1 y_j. \tag{B.9}$$

Substitute Equation (B.6) into Equation (B.9), we obtain

$$\begin{aligned} y_{j+1} =& a_0 x_{j+1} + w_c(j+1,j) w_e(j+1,j) c_1 \\ & \cdot (a_0 \sum_{k=0}^{j} w_c(j,k) w_e(j,k) c_1^{j-k} x_k) \\ =& a_0 (x_{j+1} + \sum_{k=0}^{j} w_c(j+1,k) w_e(j+1,k) c_1^{j-k+1} x_k) \\ =& a_0 \sum_{k=0}^{j+1} w_c(j+1,k) w_e(j+1,k) c_1^{j-k+1} x_k \end{aligned} \tag{B.10}$$

Because $w_c(j+1,j) w_c(j,k) = w_c(j+1,k)$, $w_c(j+1,j+1) = 1$, $w_e(j+1,j) w_e(j,k) =$

$w_e(j+1,k)$ and $w_e(j+1,j+1) = 1$.

Thus, Equation (B.6) is true for $i = j+1$.

Appendix C

Publications

During this thesis, 4 papers have been published, including 1 paper in an international journal and 3 papers in international conferences. In addition, 1 conference paper has been submitted for review.

C.1 Accepted Paper in International Journal

1. **Dongming Chen**, Mohsen Ardabilian, Liming Chen. "A Fast Trilateral Filter based Adaptive Support Weight Method for Stereo Matching". *IEEE Transactions on Circuits and Systems for Video Technology (T-CSVT)*. 2015.

C.2 Accepted Papers in International Conferences

1. **Dongming Chen**, Mohsen Ardabilian, Liming Chen. "A Novel Trilateral Filter based Adaptive Support Weight Method for Stereo Matching". *British Machine Vision Conference (BMVC)*, Bristol, UK. 2013. (Oral)

2. **Dongming Chen**, Mohsen Ardabilian, Xiaofang Wang, Liming Chen. "An Improved Non-Local Cost Aggregation Method for Stereo Matching based on Color and Boundary Cue". *IEEE International Conference on Multimedia and Expo (ICME)*, San Jose, California, USA. 2013.

3. Wei Wang, Lili Chen, **Dongming Chen**, Shile Li, Kolja Kuhnlenz. "Fast Object Recognition and 6D Pose Estimation using Viewpoint Oriented Color-Shape Histogram" . *IEEE International Conference on Multimedia and Expo (ICME)*, San Jose, California, USA. 2013.

C.3 Submitted Papers in International Conference

1. **Dongming Chen**, Mohsen Ardabilian, Liming Chen. "Depth Edge based Trilateral Filter Method for Stereo Matching". *IEEE International Conference on Image Processing (ICIP)* 2015.

Bibliography

[1] M. Pollefeys, "Visual 3d modeling from images," http://www.cs.unc.edu/~marc/tutorial/index.html, tutorial Notes, Accessed Novermber 4, 2014. 3, 4, 5

[2] T. Beeler, B. Bickel, P. Beardsley, B. Sumner, and M. Gross, "High-quality single-shot capture of facial geometry," *ACM Transactions on Graphics*, vol. 29, no. 3, pp. 40:1–40:9, 2010. 4, 5

[3] P. Sturm and S. Ramalingam, "A generic concept for camera calibration." in *Proceedings of European Conference on Computer Vision*, vol. 3022, 2004, pp. 1–13. 5, 15

[4] P. Sturm, "Critical motion sequences for monocular self-calibration and uncalibrated euclidean reconstruction." in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1997, pp. 1100–1105. 5

[5] P. Sturm and S. J. Maybank, "On plane-based camera calibration: A general algorithm, singularities, applications," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1999, pp. 432–437. 5

[6] P. Sturm, "A case against kruppa's equations for camera self-calibration." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1199–1204, 2000. 5

[7] B. K. P. Horn and M. Brooks, *Shape from Shading.* MIT Press, 1989. 5

[8] K. Yoon and I. Kweon, "Adaptive support-weight approach for correspondence search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 650–656, Apr. 2006. 6, 22, 25, 33, 34, 35, 36, 37, 42, 55, 59, 60, 66, 67, 68, 72, 74, 77, 78, 84, 86, 87, 91, 95, 96

[9] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," in *Proceedings of IEEE*

Conference on Computer Vision and Pattern Recognition, 2011, pp. 3017–3024. 6, 22, 23, 25, 34, 42, 48, 50, 55, 59, 61, 65, 66, 72, 78, 84, 86, 87, 91, 95, 97, 106, 107

[10] A. Hosni, M. Bleyer, M. Gelautz, and C. Rhemann, "Local stereo matching using geodesic support weights," in *Proceedings of IEEE International Conference on Image Processing*, 2009, pp. 2069–2072. 6, 22, 25, 34, 42, 43, 44, 55, 66, 72, 78, 84, 91

[11] D. A. Forsyth and J. Ponce, *Computer Vision A Modern Approach.* Prentice Hall, Upper Saddle River, 2003. 9

[12] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000. 9

[13] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations." in *Proceedings of IEEE International Conference on Computer Vison*, 1999, pp. 666–673. 15

[14] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1997. 15

[15] A. Fusiello, E. Trucco, and A. Verri, "A compact algorithm for rectification of stereo pairs," *Machine Vision and Applications*, vol. 12, no. 1, pp. 16–22, Jul. 2000. 17

[16] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 7–42, Apr. 2002. 21, 25, 33, 49, 50, 51, 53, 55, 66, 84, 85, 106

[17] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, Nov. 2001. 21, 50

[18] T. Meltzer, C. Yanover, and Y. Weiss, "Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation," in *Proceedings of IEEE International Conference on Computer Vison*, 2005, pp. 428–435. 22, 50

[19] S. Mattoccia and F. Tombari, "Stereo vision enabling precise border localization within a scanline optimization framework," in *Proceedings of Asian Conference on Computer Vision*, 2007. 22

[20] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister, "High-quality real-time stereo using adaptive cost aggregation and dynamic programming," in *Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, 2006, pp. 798–805. 22, 78

[21] Q. Yang, "A non-local cost aggregation method for stereo matching," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1402–1409. 22, 23, 25, 35, 37, 39, 41, 42, 50, 52, 55, 66, 72, 78, 85, 86, 87, 91, 95, 96

[22] ——, "Recursive bilateral filtering," in *Proceedings of European Conference on Computer Vision*, 2012. 22, 23, 25, 35, 37, 41, 55, 72, 78, 80, 81, 83, 85, 86, 87, 91, 93, 95, 96, 117, 118

[23] L. De-Maeztu, A. Villanueva, and R. Cabeza, "Near real-time stereo matching using geodesic diffusion." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 2, pp. 410–416, 2012. 22, 25, 35, 45, 47, 55, 72, 84, 86, 87, 89, 91, 95, 96

[24] H. Hirschmuller, "Evaluation of cost functions for stereo matching," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 22, 24

[25] C. Pham and J. Jeon, "Domain transformation-based efficient cost aggregation for local stereo matching." *IEEE Transactions on Circuits and Systems for*

Video Technology, vol. 23, no. 7, pp. 1119–1130, 2013. 23, 78, 85, 86, 87, 89, 91, 95, 96

[26] S. Vassiliadis, E. Hakkennes, J. Wong, and G. Pechanek, "The sum-absolute-difference motion estimation accelerator," in *Proceedings of the 24th Conference on EUROMICRO*, 1998. 23

[27] K. Sun, "Adaptive motion estimation based on statistical sum of absolute difference," in *Proceedings of IEEE International Conference on Image Processing*, 1998. 23

[28] R. Cutler and M. Turk, "View-based interpretation of real-time optical flow for gesture recognition," in *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 416–421. 23

[29] I. Haritaoglu, D. Harwood, and L. S. Davis, "W 4 s: A real-time system for detecting and tracking people in 2 1/2d," in *Proceedings of Proceedings of European Conference on Computer Vision*, 1998. 23

[30] A. Bruhn and J.Weickert, "Optical flow estimation on coarse-to-fine region-trees using discrete optimization," in *Proceedings of IEEE International Conference on Computer Vision*, 2005. 24

[31] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, 2010. 24

[32] H. Hirschmuller and D. Scharstein, "Evaluation of stereo matching costs on images with radiometric differences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 9, pp. 1582–1599, 2009. 24

[33] H. Hirschmuller, P. Innocent, and J. Garibaldi, "Real-time correlation-based stereo vision with reduced border errors," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 229–246, Apr. 2002. 24

[34] H. Hirschmuller, P. R. Innocent, and J. M. Garibaldi, "Real-time correlation-based stereo vision with reduced border errors," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 229–246, 2002. 24, 25, 29, 33, 59

[35] K. Konolige, "Small vision system: Hardware and implementation," in *Proceedings of the International Sympposium of Robotics Research, year = 1997, pages = 111-116.* 24

[36] R. D. Arnold, "Automated stereo perception," Ph.D. dissertation, Stanford University, 1983. 25

[37] M. D. Levine, D. A. O'Handley, and G. M. Yagi, "Computer determination of depth maps," *Computer Graphics and Image Processing*, vol. 2, no. 2, 1973. 25, 32

[38] M. Okutomi and T. Kanade, "A locally adaptive window for signal matching," *International Journal of Computer Vision*, vol. 7, no. 2, pp. 143–162, 1992. 25

[39] D. Geiger, B. Ladendorf, and A. Yuille, "Occlusions and binocular stereo," *International Journal of Computer Vision*, vol. 14, no. 3, pp. 211–226, 1995. 25, 27, 33

[40] 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, no. 2, pp. 283–296, 2007. 25, 55

[41] F. Tombari, S. Mattoccia, L. D. Stefano, and E. Addimanda, "Classification and evaluation of cost aggregation methods for stereo correspondence." in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 25

[42] A. Fusiello and V. Roberto, "Efficient stereo with multiple windows," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1997, pp. 858–863. 25, 27, 33, 59

[43] O. Veksler, "Fast variable window for stereo correspondence using integral images," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2003, pp. 556–561. 25, 32, 33, 55, 59

[44] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: Theory and experiment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 9, 1994. 26, 33

[45] S. S. Intille and A. F. Bobick, "Disparity-space images and large occlusion stereo," in *Proceedings of European Conference on Computer Vision*, 1994. 27, 33, 59

[46] S. Adhyapak, N. Kehtarnavaz, and M. Nadin, "Stereo matching via selective multiple windows," *Journal of Electronic Imaging*, vol. 16, no. 1, p. 013012, 2007. 33

[47] M. Okutomi and T. Kanade, "A locally adaptive window for signal matching," *International Journal of Computer Vision*, vol. 7, no. 2, pp. 143–162, 1992. 33

[48] A. F. Bobick and S. S. Intille, "Large occlusion stereo," *International Journal of Computer Vision*, vol. 33, no. 3, 1999. 33

[49] R. M. Zhong Dan Lan, "Robust matching by partial correlation," in *Proceedings of British Machine Vision Conference*, 1995. 34

[50] K. Prazdny, "Detection of binocular disparities," *Biological Cybernetics*, vol. 52, no. 2, 1985. 34

[51] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color image." in *Proceedings of IEEE International Conference on Computer Vison*, 1998. 35, 45, 52

[52] F. Crow, "Joint bilateral upsampling," in *Proceedings of ACM SIGGRAPH*, 2007. 35

[53] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Modeling and Simulation*, vol. 4, pp. 490–530, 2005. 35

[54] Q. Yang, R. Yang, J. Davis, and D. Nister, "Spatial-depth super resolution for range images," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 35

[55] W. Wong, A. Chung, and S. Yu, "Trilateral filtering for biomedical images," in *Proceedings of IEEE International Symposium on Biomedical Imaging*, 2004. 35

[56] E. P. Bennett and L. McMillan, "Video enhancement using per-pixel virtual exposures," in *Proceedings of ACM SIGGRAPH*, 2005. 35

[57] E. P. Bennett, J. L. Mason, and L. McMillan, "Multispectral bilateral video fusion," *IEEE Transactions on Image Processing*, vol. 16, no. 5, 2007. 35

[58] B. Oh, M. Chen, J. Dorsey, and F. Durand, "Imagebased modeling and photo editing," in *Proceedings of ACM SIGGRAPH*, 2001. 35

[59] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," in *Proceedings of ACM SIGGRAPH*, 2002. 35

[60] G. Petschnigg, M. Agrawala, H. Hoppe, R. Szeliski, M. Cohen, and K. Toyama, "Digital photography with flash and no-flash image pairs," in *Proceedings of ACM SIGGRAPH*, 2004. 35

[61] R. Ramanath and W. Snyder, "Adaptive demosaicking," *Journal of Electronic Imaging*, vol. 12, 2003. 35

[62] H. Winnemoller, S. Olsen, and B. Gooch, "Real-time video abstraction," in *Proceedings of ACM SIGGRAPH*, 2006. 35

[63] J. Xiao, H. Cheng, H. Sawhney, C., and M. Isnardi, "Bilateral filtering-based optical flow estimation with occlusion detection," in *Proceedings of European Conference on Computer Vision*, 2006. 35

[64] P. Sand and S. Teller, "Particle video: Long-range motion estimation using point trajectories," in *Proceedings of European Conference on Computer Vision*, 2006. 35

[65] S. Mattoccia, S. Giardino, and A. Gambini, "Accurate and efficient cost aggregation strategy for stereo correspondence based on approximated joint bilateral filtering," in *Proceedings of Asian Conference on Computer Vision*, 2009, pp. 371–380. 37

[66] C. Richardt, D. Orr, I. P. Davies, A. Criminisi, and N. A. Dodgson, "Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid," in *Proceedings of European Conference on Computer Vision*, 2010, pp. 510–523. 37, 78, 84

[67] Q. Yang, K.-H. Tan, and N. Ahuja, "Real-time o(1) bilateral filtering." in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 557–564. 37, 52, 78

[68] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," *International Journal of Computer Vision*, vol. 81, pp. 24–52, Apr. 2009. 37

[69] F. Porikli, "Constant time o(1) bilateral filtering." in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 37, 52

[70] Q. Yang, S. Wang, and N. Ahuja, "Svm for edge-preserving filtering," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 37

[71] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," *International Journal of Computer Vision*, vol. 81, no. 1, pp. 24–52, 2009. 37, 78

[72] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 257–266, 2002. 37, 78

[73] A. Hosni, M. Bleyer, and M. Gelautz, "Secrets of adaptive support weight techniques for local stereo matching," *Computer Vision and Image Understanding*, vol. 117, pp. 620–632, 2013. 37, 66, 78, 84, 85

[74] J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," in *Proceedings of the American Mathematical Society, 7*, 1956. 41

[75] E. W. Dijkstra, "A note on two problems in connexion with graphs," *NUMERISCHE MATHEMATIK*, vol. 1, no. 1, pp. 269–271, 1959. 43

[76] G. BORGEFORS, "Distance transformations in digital images," *Computer Vision, Graphics and Image Processing*, vol. 34, no. 3, 1986. 44

[77] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 629–639, 1990. 44

[78] D. Min and K. Sohn, "cost aggregation and occlusion handling with wls in stereo matching," *IEEE Transactions on Image Processing*, vol. 17, no. 8, 2008. 45

[79] I. S. K. Kuk-Jin Yoon, Yekeun Jeong, "Support aggregation via non-linear diffusion with disparity-dependent support-weights for stereo matching," in *Proceedings of Asian Conference on Computer Vision*, 2009. 45

[80] K. He, J. Sun, and X. Tang, "Guided image filtering," in *Proceedings of European Conference on Computer Vision*, 2010, pp. 1–14. 47, 52

[81] L. De-Maeztu, S. Mattoccia, A. Villanueva, and R. Cabeza, "linear stereo matching," in *Proceedings of IEEE International Conference on Computer Vision*, 2011. 48

[82] R. B. Potts, "Some generalized order-disorder transformations," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 48, no. 01, pp. 106–109, Jan. 1952. 50

[83] J. Besag, "On the statistical analysis of dirty pictures," *Journal of the Royal Statistical Society. Series B*, vol. 48, no. 3, pp. 259–302, 1986. 50

[84] S. Barnard, "Stochastic stereo matching over scale," *International Journal of Computer Vision*, vol. 3, no. 1, pp. 17–32, 1989. 50

[85] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, 2004. 50

[86] T. J. M. Wainwright and A. Willsky, "Map estimation via agreement on trees: Message-passing and linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 11, pp. 3697–3717, 2005. 50

[87] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1568–1583, 2006. 50

[88] N. Komodakis, N. Paragios, and G. Tziritas, "Mrf energy miniimization and beyond via dual decomposition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, 2011. 50

[89] 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," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 6, pp. 1068–1080, 2008. 50

[90] 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. 50

[91] D. Cline, K. White, and P. Egbert, "Fast 8-bit median filtering based on separability," in *Proceedings of IEEE International Conference on Image Processing*, 2007. 51

[92] S. Perreault and P. Hebert, "Median filtering in constant time," in *IEEE Transactions on Image Processing*, 2007. 51

[93] F. Crow, "Summed-area tables for texture mapping," in *Proceedings of Siggraph*, 1984. 51

[94] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2001. 51

[95] Z. Ma, K. He, Y. Wei, J. Sun, and E. Wu, "Constant time weighted median filtering for stereo matching abd beyond," in *Proceedings of IEEE International Conference on Computer Vison*, 2013. 52, 106

[96] E. Gastal and M. Oliveira, "Domian transform for edge-aware image and video processing," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 69:1–69:12, 2011. 52, 85

[97] D. Scharstein and R. Szeliski, "Middlebury stereo evaluation - version 2," http://vision.middlebury.edu/stereo/. 53, 54, 71, 84, 106

[98] D. Scharstein and R.Szeliski, "High-accuracy stereo depth maps using structured light," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2003, pp. 195–202. 55, 84

[99] D. Scharstein and C. Pal, "Learning conditional random fields for stereo," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 55, 84

[100] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 55, 84

[101] F. Tombari, S. Mattoccia, and L. D. Stefano, "Segmentation-based adaptive support for accurate stereo correspondence," in *Proceedings of Pacific-Rim*

Symposium on Image and Video Technology, vol. 1, 2007, pp. 427–438. 66, 72, 91

[102] M. Morronea and R. Owens, "Feature detection from local energy," *Pattern Recognition Letters*, vol. 6, no. 5, Dec. 1987. 68

[103] L. Rosenthaler, F. Heitger, O. Kubler, and R. von der Heydt, "Detection of general edges and keypoints," in *Proceedings of European Conference on Computer Vision*, 1992. 68

[104] B. Robbins and R. A. Owens, "2d feature detection via local energy," *Image and Vision Computing*, vol. 15, no. 5, pp. 353–368, 1997. 68

[105] J. Malik and P. Perona, "Preattentive texture discrimination with early vision mechanisms," *Journal of the Optical Society of America A*, vol. 7, pp. 923–932, 1990. 69

[106] D. Gabor, "Theory of communication," *Journal of the Institution of Electrical Engineers*, vol. 93, pp. 429–457, 1946. 69

[107] J. G. Daugman, "Two-dimensional spectral analysis of cortical receptive field profiles," *Vision Research*, vol. 20, no. 10, pp. 847–856, 1980. 69

[108] ——, "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters." *Journal of the Optical Society of America. A, Optics and image science*, vol. 2, no. 7, pp. 1160–1169, 1985. 69

[109] G. H. Granlund and H. Knutsson, *Signal processing for computer vision.* Kluwer, 1995. 69

[110] T. Cour, F. Benezit, and J. Shi, "Spectral segmentation with multiscale graph decomposition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 1124–1131. 70, 115

[111] A. Oppenheim and J. Lim, "The importance of phase in signals," *Proceedings of the IEEE*, vol. 69, pp. 529–541, 1981. 70

[112] R. Owens, "Computer vision tutorial," http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT7/node2.html, accessed April 4, 2014. 70, 71

[113] S. Venkatesh and R. A. Owens, "An energy feature detection scheme," in *Proceedings of IEEE International Conference on Image Processing*, 1989. 70

[114] A. Al-Sarraf, T. Vaudrey, R. Klette, and Y. Woo, "An approach for evaluating robustness of edge operators using real-world driving scenes," in *Proceedings of the International Conference on Image and Vision Computing New Zealand*, 2008. 71

[115] S. Orfanidis, *Introduction to Signal Processing.* Upper Saddle River, NJ: Prentice Hall Inc., 1996. 79

[116] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 2, pp. 504–511, 2013. 84, 91, 96, 97

[117] P. Kovesi, "Matlab and octave functions for computer vision and image processing," http://people.csse.uwa.edu.au/pk/Research/MatlabFns/index.html, accessed April 4, 2014. 103

[118] D. Chen, M. Ardabilian, and L. Chen, "A Novel Trilateral Filter based Adaptive Support Weight Method for Stereo Matching," in *Proceedings of British Machine Vision Conference*, 2013. 106, 107, 109

[119] D. Nehab, A. Maximo, R. S. Lima, and H. Hoppe, "GPU-efficient recursive filtering and summed-area tables," *ACM Transactions on Graphics (Proceedings of the ACM SIGGRAPH Asia 2011)*, vol. 30, no. 6, p. 176, 2011. 112

[120] D. Ruijters and P. Thévenaz, "GPU prefilter for accurate cubic B-spline interpolation," *The Computer Journal*, vol. 55, no. 1, pp. 15–20, January 2012. 112

[121] T. H. Kim, K. M. Lee, and S. U. Lee, "Learning full pairwise affinities for spectral segmentation." in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2101–2108. 115

[122] H. Lu, R. Zhang, S. Li, and X. Li, "Spectral segmentation via midlevel cues integrating geodesic and intensity." *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 2170–2178, 2013. 115