http://researchspace.auckland.ac.nz

ResearchSpace@Auckland

Copyright Statement

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

This thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author's right to be identified as the author of this thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from their thesis.

To request permissions please use the Feedback form on our webpage.
http://researchspace.auckland.ac.nz/feedback

General copyright and disclaimer

In addition to the above conditions, authors give their consent for the digital copy of their work to be used subject to the conditions specified on the Library Thesis Consent Form and Deposit Licence.
Improving the Usability of Real-Time Passive Stereo Sensors

Muhammad Tariq Ali Khan

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Department of Electrical and Computer Engineering
The University of Auckland
New Zealand
June 2014
I dedicate this work

to my parents
who are my best teachers, best guides and main source of motivation,

to my siblings
who always love, support, encourage and motivate me,

to my wife and son Nangial Khan
who bring love and colours to my life.
Abstract

There is an increasing industrial demand for robust real-time intelligent vision systems. High resolution stereo can play an important role but the high computational complexity for generating a depth map is the main concern. Hardware acceleration (using GPU or FPGA) can help satisfy real-time requirements. Symmetric Dynamic Programming Stereo (SDPS) is the highest throughput algorithm available on hardware (both in GPU and FPGA). However, the depth maps are generally noisy which reduces its usability. Therefore, the target here is to design and propose fast post-processing algorithms for the data generated by SDPS hardware which will improve its usability and will assist high level vision tasks.

This thesis makes three main contributions. The first one is the fast depth contour generation algorithm. Depth contours represent object shapes which is the key feature for object modelling, detection, pose estimation and recognition. The second contribution is the Points-of-Interest detection algorithm; it is fast and robust to depth noise due to its use of left and right image pair. Both contour generation and Points-of-Interest detection take one pass which reduces the search space and then uses the reduced search space for subsequent processing. The third contribution are the two depth contour refinement algorithms. To improve depth contours, conventionally depth maps are refined before generating contours which can remove some fine details present in depth map. Two novel contour refinement algorithms are presented in this thesis which improve depth contours and reduce the number of streaks in the contour while preserving fine detail.

**Keywords**: Real-time stereo, symmetric dynamic programming stereo (SDPS), depth contours, points of interest detection, key points detection, depth features, depth contour refinement, streak points count, contour mapping measure.
Acknowledgements

First and foremost all praises and deepest gratitude to ALLAH almighty the Most Gracious and Most Merciful, who gave me life, quest for knowledge, power to complete this thesis and for giving me outstanding vision system which is too hard to mimic.

I want to express my warmest gratitude to my supervisors Associate Professor John Morris and Dr Morteza Biglari-Abhari for their thought provoking suggestions, superb guidance, support and encouragement. I would also like to express my respect and gratitude to Professor Georgy Gimel’farb for all the time which he gave me from his busy life and for his enormous support, guidance and feedback.

I am thankful to the Photogrammetery lab colleagues Khurram Jawed who made FPGA and to Ratheesh Kalarot who made GPU SDPS systems available for my research and for their ongoing cooperation. I am also thankful to Waqar Khan and Usman Butt who helped me with data capturing and for sharing their experience with me. I am grateful to NB friends for their support, organizing Friday cricket and for arranging nice social gatherings with great food. Thanks to the PSA friends for their help and support.

I am also thankful to the Department of Computer Science for giving me a work place and access to different facilities in Tamaki campus. Thanks to Professor Reinhard Klette and ‘enpeda.’ group students for weekly talks, support and for the social gatherings. Thanks to Yeojin Jo, Dylan Rogers and Dr S. Manoharan for their help and cooperation.

I would also like to thank Associate Professor John Morris for all the sailing and being a part of the ‘Iolanthe II’ crew was much needed experience in the City of Sails. Thanks to Helen van der Peyl, Jack van der Peyl and sister Emily for their support. Lastly, special thanks to the Higher Education Commission of Pakistan for supporting this research.
# Contents

Abstract v  
Acknowledgements vii  

## 1 Introduction 1  
1.1 Motivation .............................................. 1  
1.1.1 Active Sensor ........................................... 2  
1.1.2 Passive Sensor .......................................... 2  
1.1.3 Advantages of Passive Sensors ......................... 3  
1.2 Aim and Objectives ....................................... 7  
1.3 Contribution ............................................. 8  
1.4 Applications ........................................... 9  
1.5 Structure of the Thesis .................................. 11  

## 2 Real-Time Stereo Vision 13  
2.1 Introduction ............................................. 13  
2.2 Pinhole Camera Model ...................................... 14  
2.3 Image Representation ...................................... 17  
2.4 Stereo System ........................................... 18
2.4.1 Epipolar Geometry .................................. 22
2.5 Calibration and Rectification ............................. 23
  2.5.1 Camera Calibration .................................. 25
  2.5.2 Distortion Removal .................................. 25
  2.5.3 Stereo Rectification .................................. 27
2.6 Stereo Matching ............................................ 29
2.7 Stereo Matching Algorithms .............................. 33
  2.7.1 Correlation Matching ................................. 33
  2.7.2 Symmetric Dynamic Programming Stereo (SDPS) ... 33
  2.7.3 Belief Propagation .................................... 37
  2.7.4 Semi-Global Matching ................................. 37
2.8 Evaluation Metrics for Stereo Matching ................. 38
2.9 Evaluation of Real-Time Stereo Matching ............... 40
2.10 Hardware Implementation of SDPS ................. 44
  2.10.1 FPGA Implementation ............................. 44
  2.10.2 GPU Implementation ............................... 49
2.11 Summary .................................................. 50

3 Depth Contours .............................................. 53
3.1 Introduction .............................................. 53
3.2 Contour Generation .................................. 56
  3.2.1 State Assignment .................................. 56
  3.2.2 Contour Extraction Algorithm ..................... 57
  3.2.3 Salmon Operations .................................. 61
3.3 Evaluation in Challenging Regions ...................... 61
3.4 Experimental Results .................................... 65
  3.4.1 Points Statistics .................................. 66
  3.4.2 Ground Truth Contours ............................ 68
  3.4.3 Accuracy ............................................ 69
  3.4.4 Contour Length Reduction ......................... 71
### 3.4.5 Performance ............................................. 72

### 3.5 Summary .................................................. 74

#### 4 Point-of-Interest Detection .................................. 77

- **4.1 Introduction** ........................................ 77
- **4.2 Related Work** ......................................... 79
- **4.3 Triple Edge Points Detection** .............................. 80
  - 4.3.1 Disparity State Assignment ............................... 80
  - 4.3.2 Edge Detection in Left and Right Images ................. 81
  - 4.3.3 Triple Edge Points Detection Algorithm .................. 81
- **4.4 Object Based Ground Truth Generation** .................... 91
- **4.5 Experimental Results** .................................. 93
  - 4.5.1 Significant Triple Edge Points ......................... 96
  - 4.5.2 Triple Edge Points Count ............................... 98
  - 4.5.3 Performance ........................................... 99
- **4.6 Summary** ................................................ 101

#### 5 Contour Refinement .......................................... 103

- **5.1 Introduction** .......................................... 103
- **5.2 Contour Refinement** .................................... 105
- **5.3 Proposed Algorithms** ................................... 107
  - 5.3.1 The Mn-Mx Algorithm .................................. 108
  - 5.3.2 The Triple Edge Mn-Mx Algorithm ...................... 110
- **5.4 Evaluation Metrics** .................................... 113
  - 5.4.1 Streak Points Count ................................. 113
  - 5.4.2 Contour Mapping Measure .............................. 115
- **5.5 Ground Truth** ........................................ 115
- **5.6 Experimental Results** .................................. 118
  - 5.6.1 Streak Points Count ................................. 120
  - 5.6.2 Contour Mapping Measure .............................. 121


<table>
<thead>
<tr>
<th>5.6.3</th>
<th>Analysis</th>
<th>123</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.7</td>
<td>Summary</td>
<td>124</td>
</tr>
</tbody>
</table>

6 Conclusion and Future Work 127
6.1 Directions for Future Research 129

A Appendix A: Stereo Configuration 131
A.1 Cameras used with SDPS Hardware 131
A.2 Stereo Configuration used for Experiments 132

List of Figures 143
List of Tables 150
Bibliography 153
Chapter 1

Introduction

Our brain uses different cues obtained from visual data to detect depth. Monocular vision cues are shadows, comparative size and motion. However, binocular vision cues allow for a more accurate perception of depth [152]. For binocular cues, our brain uses images of the scene from left and right eye to perceive depth [118]. It is not a new phenomenon: Charles Weatstone reported it in 1838 [157]. However, its computational complexity and poor quality prevented many systems from using it, especially at high resolution.

1.1 Motivation

Humans are the most intelligent creatures on this planet and are gifted with different senses - sight is the dominant one among these. We humans live in a visual world and most of the time we use visual data to observe, to operate, to make decisions and to learn. Sight is thus the main source of our knowledge. Seideman et al. state that 80% of what we learn come through sight [135].

In order to operate in an unknown and uncontrolled environment, the foremost thing we need is to analyse the scene and sense the depth of surrounding objects to avoid collisions. Humans mainly use vision for this purpose. If a machine needs to analyse the scene, it can use techniques based on active
sensing or passive sensing, or both.

1.1.1 Active Sensor

In active sensing techniques, the system transmits a pulse of energy to the environment and the response is used to calculate the depth. Many active vision techniques are either based on time of flight principles or on structured light.

In time of flight based techniques, the system consists of two parts: emitter and detector. The emitter emits a pulse and the detector measures the elapsed time until the reflected or scattered pulse is received. Common times of flight systems are laser range finders, Light Detection and Ranging (LIDAR or LADAR), active Sound Navigation and Ranging (SONAR) and time-of-flight cameras.

Structured light systems project a known pattern onto the environment and sense the reflected or scattered pattern to compute depth. These systems use a projector to transmit the pattern and a camera to sense reflected pattern. Microsoft Kinect uses this technique for depth perception [110, 143]. Kinect uses a pseudo random near IR (i.e. invisible to human eyes) pattern and an IR camera to observe the scene. The transmitted pattern can be either a visible or an invisible pattern.

1.1.2 Passive Sensor

Passive sensor detects the radiation that is emitted, reflected or scattered by the object to compute depth. The most common radiation is the reflected sunlight measured by the passive sensor to analyse the scene. Cameras are the most common passive sensor. Passive SONAR is also available as it can listen to sound to compute depth.
1.1. Motivation

1.1.3 Advantages of Passive Sensors

The heart of the Google autonomous car is a laser range finder but it costs $75,000 [117]! Cost is most likely one of the main issues which can detract from its widespread use. Pinto said that Google used a great deal of technology, but this technology currently is far from perfect [117]. In dense environments, where multiple systems are used to analyse the scene, active sensing techniques will fail due to interference. For example, for reliable automated vehicles in dense traffic, there will be interference and an active vision based approach may fail.

Passive sensing based techniques are also ‘environmental friendly’ because they do not transmit anything into the environment. While using active vision, safety issues need to be considered. For example, when using laser based sensing, the laser’s effect on the environment, human eyes and other electronics should be considered before its implementation. Another problem of active sensing based approaches can be unwanted detection by others due to transmission. However, passive sensing does not have this problem [63] because the passive sensor is only sensing the environment.

On the other hand, structured light approaches are usually only better for indoor applications. For example, if Kinect transmits an IR pattern in scene, this pattern may be distorted by the sun light in outdoor environment [2]. Passive sensing approaches are generally a better choice for outdoor environments.

Passive vision is also a better choice if power consumption is a concern, because active sensing uses power for the transmission of a probe pulse.

Stereo sensor

A scene can be analyzed by a single camera (monocular vision) or by a multiple camera system. If two cameras (e.g. our eyes) are used this is called
Figure 1.1: An example to indicate a possible high probability that a monocular system will interpret the scene differently from each image.
1.1. Motivation

stereo or binocular vision. More than two cameras can also be used but such systems are outside the scope of this thesis.

A stereo vision system takes two images of a scene from different perspectives, usually referred to as left and right images. These images are slightly different. This difference is the key to depth calculation. Matching the left image to the right one to perceive depth is called stereo matching. Traditionally, vision techniques have mostly used monocular vision. However, it is now increasingly being recognized that stereo data is richer and contains more useful information [95]. Especially in unknown uncontrolled environments, stereo is a better choice because monocular vision based technique cannot easily analyse the scene. A monocular system can more easily draw a false conclusion than a stereo one. For example, Figure 1.1 shows some images of almost the same scene with slight variations. The probability is high that a monocular system will interpret the scene differently from each image. From Figure 1.1(a) and (b) the system may conclude that someone is picking a small statue or chess king or pawn. But for Figure 1.1 (c), (d) and (e) the conclusion will be different. However, apart from monocular data (from left or right image) a stereo system also uses a second perspective to measure depth and should most likely draw the same conclusion (it is the same statue) in all cases.

Figure 1.2 shows another scenario. In this case the system is used to count the number of actual faces. Figure 1.2(a) shows a raw image 1, (b) shows a result generated by a monocular system and (c) shows the result generated by a stereo system. The stereo system easily identifies the actual faces due to the second perspective.

For humans too, binocular vision has always been preferred over monocular vision (looking only with one eye) 2 when trying to accomplish complex

---

1 The faces on the board were taken from NZ Herald http://www.nzherald.co.nz
2 To analyse a scene though we use some other cues like shadow, comparative size, motion and known objects.
1. Introduction

(a) Raw image

(b) Monocular based vision  
(c) Stereo based vision

Figure 1.2: Face detection by monocular and stereo vision based system. The aim is to identify actual faces.

tasks [152]. Recently Melmoth and Simon [101] provided empirical support for this view by carrying out an experiment on reaching and grasping. Chapman et al. [21] studied the importance of foot placement accuracy with binocular vision. These studies favour binocular vision over monocular vision for accuracy.
1.2 Aim and Objectives

Real-time stereo sensor

One reason that human-like vision (stereo vision) has been ignored for real-world applications is the computational complexity of stereo matching. For example, approximately 3 billion calculations are required for stereo matching per second for $1024 \times 768$ images at 30fps. However, in the last four to five years, there has been a seemingly good progress in terms of computational speed. Now there are a number of algorithms which are either implemented on Field Programmable Gate Array (FPGA) or on Graphics processing unit (GPU) which can generate a depth map in real-time (discussed in Chapter 2).

Gimel’farb’s Symmetric Dynamic Programming Stereo (SDPS) [47, 46, 48] based hardware FPGA system was implemented in our laboratory in 2009 [108]. The systems generate depth maps for high resolution images ($1024 \times 768$) at 30fps with 128 disparity range ($\Delta$). This was the first system which could achieve 1% depth accuracy in real-time for high resolution images.

1.2 Aim and Objectives

There is an increasing industrial demand for robust real-time intelligent vision systems. They can improve performance, productivity and safety for autonomous and intelligent vehicles. They can be used in sports for augmented commentary and referee assistance and also for materials handling. Inspired by nature, binocular stereo can play an important role. However, current real-time stereo systems produce poor depth maps and the amount of generated data is large. So, the aim of this thesis is to improve - in a computationally efficient manner suitable for real-time use - the usability of depth maps obtained in real-time from high resolution images (typically $1024 \times 768$) using Symmetric Dynamic Programming Stereo (SDPS) for tasks such as scene analysis, object detection and object pose estimation.
The SDPS hardware generated depth maps exhibit many artefacts, e.g. dynamic programming ‘streaks’ and imprecise object boundaries, especially in low textured scenes. Also, processing high resolution intensity images with high depth accuracy is computationally expensive. These challenges constrain the type of tasks that can use real-time stereo systems.

Therefore, the focus is on reliable information extraction techniques from generated real-time depth maps. One approach developed a fast method for depth contours extraction and refinement and the other extracted reliable points of interest by considering all available sources of information. Depth contours and reliable points of interest can play a vital role in analyzing the scene, understanding the shape and pose of the object.

1.3 Contribution

The main contributions of this thesis are:

- A novel contour generation algorithm was proposed for depth contours extraction. The algorithm reduces the search space in one pass and then uses SDPS visibility constraints in the reduced search space to extract contours. It is computationally efficient and can be useful for high level vision tasks (discussed in Chapter 3).

- Depth contour generation algorithm and its preprocessing steps were parallelized to obtain real-time performance (discussed in Chapter 3).

- Conventional three dimensional (3-D) point of interest detection techniques consider depth maps for point of interest detection. These techniques are not effective for depth maps generated by real-time stereo systems due to noisy depth maps. A novel point of interest detection technique was developed which considers left and right image pairs
1.4. Applications

with depth maps to detect points of interest due to which it is robust to depth noise (discussed in Chapter 4).

• The object depth contours generated from a real-time depth map are noisy which reduces its use. Two contour refinement algorithms were proposed which reduce the amount of noise in the generated contours and preserve the fine details if present in the depth map (discussed in Chapter 5).

• Camera calibration and raw lookup table generation for distortion removal and image rectification modules are contributed to the development of the SDPS hardware (discussed in Chapter 2).

1.4 Applications

This research is not focused on any single application as the techniques described here can contribute to many applications that require high resolution real-time stereo with ~30fps. Some areas of application are: sports monitoring and augmented commentary, road safety and material handling.

Sports monitoring and augmented commentary

Sport is a good source of entertainment whether you are a player or a fan, but unfortunately the history of sport has recorded many violent events [156, 24, 158], that are often caused by referee errors. There is no single solution to violence in sport because causes are complex. Referee decisions are one of the main causes of violence [156, 24, 158]. For example, a crucial goal or victory is stolen [24] or the most popular player is sent out of the game in confusion. In FIFA world cup 2010 there were clear referee goal mistakes, as in the match between England and Germany which England lost by 3 goals:
one clear goal for England was not given but the replay showed that there was a clear goal [112].

Sport broadcasting is a big industry and commentary is the main part of it. Augmented commentary can increase viewer interest [78, 3]. Techniques introduced in this thesis can contribute to accurate referee calls and augmented commentary.

**Road safety**

Road accidents are one of the main causes of death. According to the World Health Organization (WHO), for the age group of 15 to 29, it is the top cause of death [114]. More than 90% of road accidents are attributed to human errors [114].

Autonomous and intelligent vehicles can reduce the death toll but most vehicles are still without vision; if the driver loses concentration for a fraction of a second then serious accidents can occur [49]. The technology used in the Google’s autonomous car is very expensive and still far from perfect [117]. Real-time stereo can play an important role in autonomous and intelligent vehicles.

**Material handling**

Handling of luggage and materials is always important, in places such as airports, warehouses, manufacturing companies and the fruit industry. Handling of non-uniform (size and shape) materials needs intelligent solutions. Reducing the supervision and cost and increasing the flexibility of material handlers can lead to increased productivity.

High resolution real-time stereo can play an important role. Material handling companies are now taking interest by considering it for their future equipment.
1.5 Structure of the Thesis

The structure of the thesis with interdependencies between chapters (1 to 6) are shown in Figure 1.3. Chapter 2 covers the basic theory of stereo vision followed by description of stereo matching algorithms and evaluation techniques of these algorithms. Real-time stereo systems are discussed and compared with particular focus on FPGA and GPU implementation of the SDPS hardware. The SDPS hardware was used to stream to the host the corrected left and right images, the depth map and the occlusion map.

Chapter 3 presents a depth contour generation algorithm (Salmon) using a depth map and an occlusion map. The algorithm uses the visibility constraints of SDPS, reduces the search space in one pass and then uses the reduced search space to generate dense depth contours. The algorithm is
parallelized for improved performance. The parts of this chapter have been published [80, 81].

Chapter 4 covers the Points-of-Interest (Triple Edge) detection technique for real-time stereo systems. The technique uses a depth map with corrected left and right images. The conventional depth map Points-of-Interest detection techniques use only a depth map which is not useful for current real-time stereo systems because the amount of noise is high. This chapter has been partly published [79].

Chapter 5 presents two depth contours refinement algorithms. The algorithms deal with inherited noise in the SDPS depth contours. The amount of noise is reduced by rules which are derived from the SDPS depth map. Error metrics streak points count (SPC) and contour mapping measure (CMM) were used for the experiments. Finally, Chapter 6 concludes the thesis and highlights some future directions for research.
Chapter 2

Real-Time Stereo Vision

Binocular stereo uses two images to generate the depth map. Generating depth maps is a computationally expensive process. However, now it is possible to generate high resolution depth maps in real-time. The hardware implementations of Symmetric Dynamic Programming Stereo (SDPS) developed in the Photogrammetry Laboratory at Auckland are believed to be the best performing real-time stereo systems and can generate high resolution depth maps in real-time.

2.1 Introduction

The first part of this chapter summarizes the known basic theory to provide some background and to define basic notations. For this purpose, the pinhole camera model is explained as it is an idealized model of a projective system that projects three dimensional (3-D) scenes to two dimensional (2-D) images. Real cameras deviate from the ideal pinhole camera model due to distortions. Therefore, camera calibration and stereo rectification are required, as described in Section 2.5.

Binocular stereo uses two cameras which enable depth measurement. To calculate depth using stereo, the key difficulty is to identify the corresponding points in image pairs, i.e. the points that are projections of the same scene points. Unfortunately, this matching (stereo matching) is an ambiguous and
2. Real-Time Stereo Vision

Computationally expensive process [32, 149]. Stereo matching algorithms are discussed in this chapter.

The final part discusses some real-time stereo matching algorithms which are the main focus of this thesis. Correlation Matching (CM) [52, 70, 85], SDPS [47, 46, 48, 108, 74], Belief Propagation (BP) [30, 161] and Semi-Global Matching (SGM) [57, 4, 39] are reviewed, with particular attention to SDPS (discussed in Section 2.7.2, Section 2.9 and Section 2.10).

2.2 Pinhole Camera Model

A pinhole camera is an ideal device realizing perspective projection. Perspective projection maps a 3-D scene onto 2-D images. In this model, the distorting effects of real lenses are not considered and a perfect assembly of all camera components is assumed [14, 153].

In the pinhole camera model, a light ray enters the camera through an infinitesimally small aperture. The intersection of the light ray with the image plane (\( \Pi \)) forms the image of the object as shown in Figure 2.1. The optical
2.2. Pinhole Camera Model

Figure 2.2: Projection using pinhole camera model in which a virtual image plane has been placed in front of the optical centre. This simplifies the mathematics by removing the inversion of the image on the real image plane.

centre (or centre of projection) is the point where light rays enter the camera. The distance between optical centre and image plane is the focal length ($f$) of the camera.

To simplify the mathematics of a perspective projection, a virtual image plane is often placed in front of the camera as shown in Figure 2.2. The optical centre ($O_c$) is chosen as the origin of 3-D world coordinates. In Figure 2.2, $Z_c$ is the optical axis. The intersection of the optical axis with the image plane, $\Pi$ is the principal point, $[\mu_x \ \mu_y]^T$. The projection of point $Q$ with $|Z_q| > 0$ to point $q$ on image plane $\Pi$, is shown in Figure 2.2. The real world coordinates
of $Q$ are:

$$Q = \begin{bmatrix} X_q \\ Y_q \\ Z_q \end{bmatrix}$$  \hspace{1cm} (2.1)$$

and the camera matrix $C_p$ for (perfect) pinhole camera, maps from a 3-D to the 2-D image plane:

$$C_p = \begin{bmatrix} f & 0 & \mu_x \\ 0 & f & \mu_y \\ 0 & 0 & 1 \end{bmatrix}$$  \hspace{1cm} (2.2)$$

Homogeneous coordinates are usually used to compute $q$ from $Q$ and $C$ because they allow us to use matrix algebra. In homogeneous coordinates, in perspective space a point with dimension $n$ is expressed in $n+1$ dimensions [55]. $q$ in homogeneous coordinates expressed as :

$$q = \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$  \hspace{1cm} (2.3)$$

Now we can calculate $q$ from $C$ and $Q$ :

$$q = C_p Q$$  \hspace{1cm} (2.4)$$

Solving Equation 2.4 for $x$ and $y$ gives [14, 55]:

$$x = \left( \frac{f X_q}{Z_q} + \mu_x \right)$$

$$y = \left( \frac{f Y_q}{Z_q} + \mu_y \right)$$  \hspace{1cm} (2.5)$$
2.3. Image Representation

For a real camera, the camera matrix is:

\[
C = \begin{bmatrix}
  f_x & 0 & \mu_x \\
  0 & f_y & \mu_y \\
  0 & 0 & 1 \\
\end{bmatrix}
\]  

(2.6)

where \( f_x \) and \( f_y \) are the focal lengths on the x-axis and y-axis. Two values are introduced here to allow for slightly imperfect lenses and non-square pixels.

2.3 Image Representation

In a camera, light rays enter and hit the image plane which is a \( w \times h \) rectangular grid of sensors. These sensors are photosensors as they are sensitive to light and record its intensity.

The output of the camera is a digital image that can be represented as a matrix of numbers. The entries in this matrix are called pixels (picture el-
Figure 2.4: Stereo camera setup: the textured region shows the common field of view.

$P(x, y)$ represents one pixel, in which $y$ represents a scan line and $x$ is the column. The origin of the image coordinates is chosen as the top left corner as shown in Figure 2.3. The pixel value is the recorded intensity of light. If a pixel value is represented by 8 bits, then the pixel intensity has 256 distinct values or grey levels. Figure 2.3 shows a grey scale image; for the selected $12 \times 8$ pixels patch, the grey scale values are shown.

2.4 Stereo System

In binocular stereo, two cameras labeled $L_{\text{camera}}$ and $R_{\text{camera}}$, are used. The region visible to both cameras is the common field of view. Figure 2.4 shows the field of view of each camera with the textured region showing the common field of view. The depth of (or distance to) points in the common field of view can be calculated if their corresponding points can be correctly identified. $L_{\text{camera}}$ and $R_{\text{camera}}$ are pinhole cameras with the same optical configuration
(focal length, image size), separated by a distance $b$, the baseline. $O_l$ and $O_r$ are the optical centres of $L_{camera}$ and $R_{camera}$ as shown in Figure 2.5. An object in the common field of view projects to a different location on the image planes $\Pi_l$ and $\Pi_r$. The different position of the projection on each image plane enables us to calculate the depth $Z$, of an object.

Let $Q (X, Y, Z)$ is a point on a real world object as shown in Figure 2.5. $Q$ projects onto $\Pi_l$ as $q_l$ with coordinates $(x_l, y_l)$ and onto $\Pi_r$ as $q_r$ with $(x_r, y_r)$.
To calculate the depth $Z$ of $Q$ using $q_l$ and $q_r$, observe that, in Figure 2.6, line $QO_l$ intersects with $y = y_l$ at $x_l$ and $QO_r$ intersects with $y = y_r$ at $x_r$. From similar triangles $QO_lO_r$ and $Qx_lx_r$ (see Figure 2.6):

$$\frac{b}{Z} = \frac{b - (x_l - x_r)}{Z - f} \Rightarrow Z = \frac{fb}{x_l - x_r} \quad (2.7)$$

The difference, $d = x_l - x_r$, is called disparity. Then Equation 2.7 becomes:

$$Z = \frac{fb}{d} \quad (2.8)$$

In Equation 2.8, $f$ and $b$ are constant for any given configuration, so $Z$ is inversely proportional to $d$:

$$Z \propto \frac{1}{d} \quad (2.9)$$
Figure 2.7: Depth ($Z$) is inversely proportional to disparity ($d$) - two corresponding points at different depths (one near and one far) are marked on left (top) and right (bottom) images which shows that $Z$ is inversely proportional to $d$.

Figure 2.7 shows left and right images with two subjects at different depth $Z$ to show the inversely proportional relationship with disparity $d$. 
Figure 2.8: Epipolar geometry - showing epipolar plane, epipolar line and epipolar points for point, $Q$.

### 2.4.1 Epipolar Geometry

Every point in the common field of view lies in an epipolar plane. These points pass through the centres of projection ($O_l$ and $O_r$) of both cameras (Figure 2.8). The intersection of the epipolar plane with the image plane forms an epipolar line. The projection of $O_r$ on $\Pi_l$ is the left epipole $e_l$ and the projection of $O_l$ on $\Pi_r$ is the right epipole $e_r$.

The epipoles move to infinity if the optical axes of both cameras become parallel [14, 153]. Figure 2.8 shows a configuration in which the camera axes are verged toward each other to show the epipoles clearly. In Figure 2.8, $QO_lO_r$ is the epipolar plane, $q_l e_l$ and $q_r e_r$ are epipolar lines and $e_l$ and $e_r$ are epipoles.

In this work, images have been rectified and the epipolar lines lie along camera scan lines. Each point observed on $\Pi_l$ must be observed on a known epipolar line in $\Pi_r$. For example, if the projection of point $q_l$ is known,
2.5. Calibration and Rectification

In practice, real cameras deviate from the pinhole camera model and preprocessing is required for correction. Camera calibration procedures determine camera parameters for mathematical correction.

Two types of distortion are common in real cameras - radial and tangential distortion [14, 154]. Radial distortion arises from real (thick) lenses and tangential distortion is mainly due to overall assembly of the camera [166, 154]. There is almost no distortion near the principal point and the distortion increases toward the periphery of the image. Figure 2.10 shows radial distort-

then \( q_r \) must lie on the corresponding epipolar line as shown in Figure 2.9 [14, 153].

Figure 2.9: The epipolar constraint: points in one image are projected onto the epipolar line in the other image.
2. Real-Time Stereo Vision

Figure 2.10: Contour map of distortions from our laboratory’s CameraLink (STC-CL83A) camera. The central region has essentially zero distortion.
2.5. Calibration and Rectification

tion and tangential distortion. The central circle is the region with zero or very small distortion and distortion increases from circle to circle moving out from the centre. The images used in this thesis have been corrected for both distortions, before any further processing.

The two cameras are rarely in perfect alignment with coplanar image planes and parallel optical axes. Stereo rectification transforms each image so that they become coplanar and the corresponding pixels lie in the same scan line.

2.5.1 Camera Calibration

There are a number of ways to calibrate cameras [166, 14, 122, 129, 147, 154]. Here, the procedure described by Zhang [166, 14] was applied it uses a chequerboard as calibration target. For the experiments, an 8 × 7 chequerboard of 90mm squares (7 × 6 interior corners) was used. A number of calibration images $c_i$, where $10 \leq c_i \leq 30$, were captured of the chequerboard with different locations and rotations (but no $90^\circ$ rotation) to cover the common field of view. Then the captured images were processed by an off-line calibration procedure; Figure 2.11 shows a flowchart of the procedure.

The camera parameters can be grouped into intrinsic, distortion and extrinsic parameters. There are four intrinsic parameters - two for focal length - $f_x$ and $f_y$ and two for principal point $\mu_x$ and $\mu_y$; five distortion parameters - three for radial distortion ($k_1$, $k_2$ and $k_3$) and two for tangential distortion ($p_1$ and $p_2$) [14]; and six extrinsic parameters - three for translation ($T_X$, $T_Y$, $T_Z$) and three for rotation ($\psi_r$, $\varphi_r$, $\theta_r$) [166, 14].

2.5.2 Distortion Removal

Images were corrected for radial and tangential distortion using parameters $k_1$, $k_2$, $k_3$, $p_1$ and $p_2$. The radial distortion correction is:
2. Real-Time Stereo Vision

Figure 2.11: Camera calibration procedure flowchart.

\[ x_{d1} = x_{d2}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \]
\[ y_{d1} = y_{d2}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \]  \hspace{1cm} (2.10)

where \( x_{d1} \) and \( y_{d1} \) contained tangential distortion only and are corrected for radial distortion, while \( x_{d2} \) and \( y_{d2} \) contain both radial and tangential distortions (original coordinates) [166, 14].

For tangential distortion correction, the corrected pixel coordinates \( x \) and \( y \) are:

\[ x = x_{d1} + (2p_1 y_{d1} + p_2 (r^2 + 2x_{d1}^2)) \]
\[ y = y_{d1} + (p_1 (r^2 + 2y_{d1}^2) + 2p_1 x_{d1}) \]  \hspace{1cm} (2.11)

For derivation of Equation 2.10 and Equation 2.11, see Brown [15].
2.5. Calibration and Rectification

2.5.3 Stereo Rectification

Stereo rectification makes epipolar lines colinear and parallel to the baseline, which reduces the search space for stereo matching from 2-D to 1-D. There are a number of techniques for image rectification [12, 14, 122, 129, 154]. For the experiments in this study, Bouguet’s algorithm [12, 14] was used for rectification.

\( R_l \) is the rotation matrix and \( T_l \) is the translation vector for the left camera and \( R_r \) and \( T_r \) are for the right camera. \( R \) is the rotation matrix and \( T \) is the translation vector which relates the left camera to the right camera. \( R \) and \( T \) can be calculated from [12, 14]:

\[
\begin{align*}
R &= R_r (R_l)^T \\
T &= T_r - RT_l
\end{align*}
\] (2.12)

Given \( R \) and \( T \), Bouguet’s method for stereo alignment attempts to maximize the common viewing area and to minimize the amount of change the reprojection produces.

In Bouguet’ method \( R \) is split between \( L_{camera} \) and \( R_{camera} \) to minimize reprojection changes. The resulting half rotation matrices are: \( R_{hl} \) for \( L_{camera} \) and \( R_{hr} \) for \( R_{camera} \) [12, 14]. \( R_\gamma \) is used to take camera epipole to infinity and align the epipolar line horizontally:

\[
R_\gamma = \begin{bmatrix}
(r_1)^T \\
(r_2)^T \\
(r_3)^T
\end{bmatrix}
\] (2.13)

where

\[
r_1 = \frac{T}{||T||}
\] (2.14)

\( r_2 \) must be orthogonal to \( r_1 \):
\begin{equation} r_2 = \frac{-T_y \ T_x \ 0}{\sqrt{T_x^2 + T_y^2}} \end{equation}

and \( r_3 \) is orthogonal to both \( r_1 \) and \( r_2 \):

\begin{equation} r_3 = r_1 \times r_2 \end{equation}

Row alignment is then achieved for each camera:

\begin{equation}
\begin{align*}
R_{\gamma l} &= R_{\gamma} R_{hl} \\
R_{\gamma r} &= R_{\gamma} R_{hr}
\end{align*}
\end{equation}

**Reprojection**

The image point with coordinates \( x \) and \( y \) and disparity \( d \) can be reprojected into the 3-D coordinate system using the equation:

\begin{equation}
\begin{bmatrix}
X \\
Y \\
Z \\
W
\end{bmatrix} = Q \begin{bmatrix}
x \\
y \\
d \\
1
\end{bmatrix}
\end{equation}

where points are in homogeneous coordinates and \( Q \) is the reprojection matrix:

\begin{equation}
Q = \begin{bmatrix}
1 & 0 & 0 & -\mu_x \\
0 & 1 & 0 & -\mu_y \\
0 & 0 & f & 0 \\
0 & 0 & -\frac{1}{T_x} \frac{\mu_z - \mu_z'}{T_x'}
\end{bmatrix}
\end{equation}

where all parameters are from the left camera except \( \mu_z' \), which is from the right camera [14].
2.6. Stereo Matching

The aim of stereo matching is to estimate disparity $d$ from the left and right image pair. The resulting disparity map ($D$) is the disparity estimate at each pixel. Figure 2.12 shows a stereo image pair with an ideal (ground truth) disparity map [131].

Stereo matching is still an active area of research, because estimating disparity is an ill-posed problem [106]. The ill-posedness arises from:

- Multiple surfaces
- Uniform or repetitive textures
• Partial occlusion

• Optical signal distortion.

To solve the ill-posed problem, regularization assumptions are required [32]. Most stereo matching algorithms make some or all of the regularization assumptions, for instance [48]:

• Reconstructed scene is assumed to be a single surface.

• Large occlusions are constrained

• Depth gradients are constrained

• Simple noise models are assumed.

Aligning systems (either mechanically or by rectification) so that the epipolar constraint is met, simplifies the search for stereo matching from the whole 2-D image to a 1-D scan line. Stereo matching algorithms also use minimum, \( d_{min} \) and maximum \( d_{max} \) disparities determined by scene geometry to limit the search area.

Scharstein and Szeliski state that stereo matching algorithms generally have the following four steps [132]:

• Computation of matching cost

• Aggregation of cost

• Computation of disparity

• Refinement of disparity.

The sequence of these steps depend on each algorithm.
2.6. Stereo Matching

**Computation of matching cost**

Stereo matching methods rely on a matching cost function for computing similarities between pixels (or small windows or neighbourhoods of pixels). The matching cost is computed for all pixels for each disparity between $d_{\text{min}}$ and $d_{\text{max}}$. Hirschmuller and Scharstein note that the performance of a cost function varies depending on the stereo method using it [59, 60]. Common cost functions are square of intensity difference (SD) [53, 142] and absolute difference (AD) [75]. Other cost functions include normalized cross-correlation (NCC) [53, 127], binary matching and gradient-based measures [136]. Marr et al. used a filter bank response for matching cost [97]. Brichfield and Tomasi proposed a cost function which is insensitive to image sampling [10]. More complex cost functions are Mutual Information (MI) [27, 57, 82] and approximative segment-wise MI [167].

**Aggregation of cost**

Cost aggregation is the calculation of aggregate cost in support regions. For local and windows based stereo matching approaches, this is the main step [132]. Gong et al. evaluated the performance of six different cost aggregation approaches [50]. Their evaluation suggests that properly designed cost aggregation can significantly improve the quality of the depth maps without introducing too much computation cost. Several new cost aggregations have been proposed recently [119, 69, 104, 132] with improved results.

**Disparity computation**

For local methods, the main steps are cost calculation and cost aggregation. Disparity computation simply means choosing the disparity with minimum cost value using winner-takes-all (WTA) basis [132]. In local methods, one limitation is that uniqueness of match is enforced on only one image.
2. Real-Time Stereo Vision

Figure 2.13: Left image of the scene and its reconstruction using SDPS from side view. Reconstruction does not consider sub-pixel disparities.

For global methods, disparity computation is the main step and its objective is to find minimum global energy [132]. A number of algorithms, including simulated annealing [45, 5], high confidence first [23], mean field annealing [44], and graph cut [13, 84] have then been used for the global energy minimization.

Dynamic programming algorithms are ‘global’ optimization algorithms which find the minimum for independent scan lines. It was first used by Gimel’farb in 1979 [47]. These approaches effectively find a minimum cost path through the matrix of all matches between the corresponding scan lines [132].

Disparity refinement

Most stereo matching algorithms compute disparity in a discrete space. The scene reconstruction from such algorithms appears as many discrete thin layers. Figure 2.13 (a) shows the original synthetic scene and Figure 2.13 (b) shows its reconstruction using SDPS. If we need to build an accurate 3-D model, then we need sub-pixel (non-integral) disparities. Sub-pixel disparities can be computed by iterative gradient decent and by fitting a curve to
the matching costs at discrete disparity levels [127, 94, 151, 98, 76]. Besides sub-pixel computations, another technique is by detecting occluded region cross checking disparities refinement [25, 35].

2.7 Stereo Matching Algorithms

This section introduces some stereo matching algorithms which have real-time implementation and are important for the following sections.

2.7.1 Correlation Matching

Correlation matching is a local window based method which is easy to implement and produces dense depth maps [85]. In this type of matching, a window from one image is correlated with a window from the other image. The question is how to choose the window size. Since a smaller window cannot capture enough structure, it will produce many false matches which will result in a noisy depth map. On the other hand, a larger window will be less sensitive to noise but will blur the disparity map and increase computation requirements [36, 77]. Correlation method is one of the best candidates for real-time implementation in terms of speed (see Section 2.9).

2.7.2 Symmetric Dynamic Programming Stereo (SDPS)

Symmetric dynamic programming stereo (SDPS) [47, 46, 48] is a dynamic programming stereo matching algorithm. It was originally proposed in the late 1970s by Gimel’farb [47]. A key difference of this approach is that, rather than creating a disparity map corresponding to the left or right image as the reference image, it creates the disparity map that would be seen by a virtual Cyclopæan camera placed midway between the two cameras. The Cyclopæan camera ‘sees’ twice as many points on each scan line as the orig-
Real-Time Stereo Vision

These points in Cyclopæan view are assigned one of the three visibility states [46, 48]:

- Monocularly visible to left camera (ML)
- Binocularly visible (B)
- Monocularly visible to right camera (MR)

Figure 2.14 shows that, in the Cyclopæan image, a transition between two disparity levels \(d_1\) and \(d_2\) should result in exactly \(|d_1 - d_2|\) occluded pixels marked ML or MR in the Cyclopæan view.

For simplicity the following three assumptions are made [107, 108]:

- Disparity \(d\) lies in the fixed range \(0, 1, \ldots, \Delta - 1\)
- Images are undistorted and are rectified
- Cameras are electrically and optically matched

SDPS matches scan line by scan line, so only one line is processed at a time. Thus, one dimensional array of pixels from the left \((G^l)\) and right \((G^r)\) images is processed at a time:

\[
G^l = \{G^l_x | x_l = 0, 1, \ldots, w - 1\} \\
G^r = \{G^r_x | x_r = 0, 1, \ldots, w - 1\}
\]  

(2.20)

\(w\) is the image width (or length of the scan line). \(G^l\) and \(G^r\) are used to construct a Cyclopæan image line \(G^\varsigma\):

\[
G^\varsigma = \{G^\varsigma x | x = 0, 1, 2 \ldots 2w - 1\}
\]  

(2.21)

\(G^\varsigma\) has twice as many points: Giml’farb [46, 48] used half integral indices rather than integral ones. In this thesis, integral indices are used which generate \(G^\varsigma\) of length \(2w\) [107, 108]. Here, integral indices are used rather than the half-integral ones of Gimel’farb [46, 48]
2.7. Stereo Matching Algorithms

Figure 2.14: Camera configuration, showing a part of Cyclopæan image seen by a virtual Cyclopæan camera (centre) and a scene object profile. Visibility states (ML, B or MR) are marked on profile from Morris et al. [108].

Corresponding L|R image points for a point \( x \), in Cyclopæan image at disparity \( d \) are:

\[
\begin{align*}
x_l &= \frac{x + d}{2} \\
x_r &= \frac{x - d}{2}
\end{align*}
\]  

(2.22)
Points in the Cyclopæan image, with even indices \( x = 0, 2, 4 \ldots \) represent points with even disparities and those with odd indices \( x = 1, 3, 5 \ldots \) represent points with odd disparities \[46\]. Each major step has two phases: first even indices in \( G^s \) are considered and then odd indices in \( G^s \) are computed.

The cost array, \( C \), has elements \( c_{x,d,s} \) which represent the best cost of the path from \( x = 0 \) in \( G^s \) ending at disparity \( d \) with visibility state \( s \in \{ MR, B, ML \} \).

A predecessor array \( \pi^p \) holds a reference to the predecessor of each state in \( C \).

The final module is the back-track module which works backwards through the predecessor array to build the list of disparities.

Any dissimilarity function \[59, 60\] between corresponding pixels can be used. Here, absolute difference is used because of its simplicity, speed and demonstrated performance with SDPS \[107\]. The costs to reach each point, for each state, \( ML, B \) and \( MR \) are:

\[
\begin{align*}
c_{x,d,ML} &= o_t + \min(c_{x-1,d-1,ML}, c_{x-2,d,B}, c_{x-2,d,MR}) \\
c_{x,d,B} &= dI(x, d) + \min(c_{x-1,d-1,ML}, c_{x-2,d,B}, c_{x-2,d,MR}) \\
c_{x,d,MR} &= o_t + \min(c_{x-1,d+1,B}, c_{x-1,d+1,MR})
\end{align*}
\]

where \( o_t \) is the occlusion cost associated with a disparity change which acts as smoothness constraint and \( dI(x, d) \) is the pixel dissimilarity function between \( \frac{x+d}{2} \) and \( \frac{x-d}{2} \) in \( G^l \) and \( G^r \). The predecessors, \( \pi^p_{x,d,s} \), are:

\[
\begin{align*}
\pi^p_{x,d,ML} &= \arg\min(c_{x-1,d-1,ML}, c_{x-2,d,B}, c_{x-2,d,MR}) \\
\pi^p_{x,d,B} &= \arg\min(c_{x-1,d-1,ML}, c_{x-2,d,B}, c_{x-2,d,MR}) \\
\pi^p_{x,d,MR} &= \arg\min(c_{x-1,d+1,B}, c_{x-1,d+1,MR})
\end{align*}
\]

More details can be found in Gimel’farb \[47, 46, 48\] and Morris \textit{et al.} \[107, 108\].
2.7. Stereo Matching Algorithms

2.7.3 Belief Propagation

Belief propagation is an iterative process in which neighbours talk to each other in order to solve a Markov random field. This approach was first used by Felzenwalb and Huttenlocher for stereo matching [30]. For stereo matching, the neighbours of pixel $p$ specify the depth they believe $p$ lies at. In each iteration, neighbours improve their belief. Neighbour beliefs should vary smoothly except at some edges. Belief propagation can be easily implemented and parallelized but is memory intensive and slow (discussed in Section 2.9).

2.7.4 Semi-Global Matching

Semi-Global matching was introduced by Hirschmüller [57] and combines local and global methods. It is based on pixel-wise matching of mutual information and enforces an approximate global smoothness constraint by combining a number of 1-D constraints. The main steps are:

- Pixel wise cost calculation: in the original implementation two cost functions are considered. One is Birchfield and Tomasi’s sample intensity measure [10] and the second is mutual information [57].

- Aggregation of costs: a smoothness constraint is added which penalizes changes of neighbouring disparities.

- Disparity computation, which determines disparity with sub-pixel accuracy and occlusion detection.

A drawback of this type of matching is the large memory requirement compared to local methods.
2.8 Evaluation Metrics for Stereo Matching

Stereo matching evaluation metrics are important to check the performance of stereo matching algorithms. Scharstein and Szeliski used the metrics: root mean square (RMS) error and percentage of bad pixels [132]. They calculated these metrics by using ground truth images. RMS error was calculated:

\[
\epsilon_{RMS} = \left( \frac{1}{N} \sum_{(x,y)} |D(x,y) - D_{GT}(x,y)|^2 \right)^{\frac{1}{2}}
\]

where \( N \) is the number of pixels, \( D \) is the computed depth map and \( D_{GT} \) is the ground truth depth map.

Percentage of bad pixels was calculated:

\[
\epsilon_B = \frac{1}{N} \sum_{(x,y)} (|D(x,y) - D_{GT}(x,y)| > \tau)
\]

where \( \tau \) is the bad pixel threshold.

Beside calculating these metrics (Equation 2.25 and Equation 2.26) for the whole image, Scharstein and Szeliski considered three different regions - textureless, occluded and depth discontinuity regions - in their evaluation. The Middlebury College website ranks many algorithms [132, 131].

The ground truth dataset used by Scharstein and Szeliski is small. Geiger et al. [41] recently introduced a larger ground truth dataset. They used the same metrics as Scharstein and Szeliski [132] and, for each method, they showed:

- Percentage of erroneous pixels in non-occluded regions
- Percentage of erroneous pixels in total
- Percentage of pixels for which ground truth is provided by the method
- Average disparity error in non-occluded area and
### Evaluation Metrics for Stereo Matching

#### Table 2.1: KITTI stereo matching algorithms ranking [40]. Only matching accuracy is used for ranking. Time is given in seconds $s$ for $1242 \times 375$ images. High rank algorithms are too slow (300s and 8s top two).

<table>
<thead>
<tr>
<th>Rank</th>
<th>Method</th>
<th>Timing (s)</th>
<th>Cores</th>
<th>Clock</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PCBP[160]</td>
<td>300</td>
<td>4</td>
<td>2.5 Ghz</td>
<td>Matlab + C/C++</td>
</tr>
<tr>
<td>2</td>
<td>iSGM[56]</td>
<td>8</td>
<td>2</td>
<td>2.5 Ghz</td>
<td>C/C++</td>
</tr>
<tr>
<td>3</td>
<td>SGM [58]</td>
<td>3.7</td>
<td>1</td>
<td>3.0 Ghz</td>
<td>C/C++</td>
</tr>
<tr>
<td>4</td>
<td>SNCC[28]</td>
<td>0.27</td>
<td>1</td>
<td>3.0 Ghz</td>
<td>C/C++</td>
</tr>
<tr>
<td>5</td>
<td>ITGV[120]</td>
<td>7</td>
<td>1</td>
<td>3.0 Ghz</td>
<td>Matlab + C/C++</td>
</tr>
<tr>
<td>6</td>
<td>LDE[40]</td>
<td>14</td>
<td>2</td>
<td>2.5 Ghz</td>
<td>C/C++</td>
</tr>
<tr>
<td>7</td>
<td>BSSM[40]</td>
<td>20.7</td>
<td>1</td>
<td>3.5 Ghz</td>
<td>C/C++</td>
</tr>
<tr>
<td>8</td>
<td>OCV-SGBM[14]</td>
<td>1.1</td>
<td>1</td>
<td>2.5 Ghz</td>
<td>C/C++</td>
</tr>
<tr>
<td>9</td>
<td>ELAS[42]</td>
<td>0.3</td>
<td>1</td>
<td>2.5 Ghz</td>
<td>C/C++</td>
</tr>
<tr>
<td>10</td>
<td>MS-DSI[40]</td>
<td>10.3</td>
<td>&gt;8</td>
<td>2.5 Ghz</td>
<td>C/C++</td>
</tr>
<tr>
<td>11</td>
<td>SDM[86]</td>
<td>60</td>
<td>1</td>
<td>2.5 Ghz</td>
<td>C/C++</td>
</tr>
<tr>
<td>12</td>
<td>GCSF[18]</td>
<td>2.4</td>
<td>1</td>
<td>2.5 Ghz</td>
<td>C/C++</td>
</tr>
<tr>
<td>13</td>
<td>GCS[19]</td>
<td>2.2</td>
<td>1</td>
<td>2.5 Ghz</td>
<td>C/C++</td>
</tr>
<tr>
<td>14</td>
<td>CostFilter[62]</td>
<td>4</td>
<td>1</td>
<td>2.5 Ghz</td>
<td>Matlab</td>
</tr>
<tr>
<td>15</td>
<td>OCV-BM[14]</td>
<td>0.1</td>
<td>1</td>
<td>2.5 Ghz</td>
<td>C/C++</td>
</tr>
<tr>
<td>16</td>
<td>GC+OCC[84]</td>
<td>360</td>
<td>1</td>
<td>2.5 Ghz</td>
<td>C/C++</td>
</tr>
</tbody>
</table>

- Average disparity error in total.

The ranking of algorithms is given in Table 2.1 (updated ranking can be found on the KITTI website [41, 40]). The Middlebury and KITTI rankings use only matching accuracy for ranking and the algorithms performing well on one dataset, sometimes perform badly on other datasets. Geiger et al.
found that algorithms ranking highly on Middlebury, perform particularly badly on their dataset [41]. Therefore, even though an algorithm tops a ranking, it does not always mean that it is the best available stereo matching algorithm.

### 2.9 Evaluation of Real-Time Stereo Matching

The Middlebury and KITTI rankings consider matching performance only. However, for real-time applications, a critical factor is speed. Table 2.1 shows computation times on the KITTI dataset. The best matching algorithm takes 300s for a KITTI dataset image pair (1241 × 375). These algorithms are too slow for real-time applications. A number of real-time stereo systems are
2.9. Evaluation of Real-Time Stereo Matching

<table>
<thead>
<tr>
<th>Disparity range (Δ)</th>
<th>32</th>
<th>64</th>
<th>96</th>
<th>128</th>
<th>160</th>
<th>192</th>
<th>244</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDPS</td>
<td>325</td>
<td>299</td>
<td>268</td>
<td>267</td>
<td>268</td>
<td>287</td>
<td>278</td>
<td>270</td>
</tr>
<tr>
<td>CM</td>
<td>627</td>
<td>466</td>
<td>358</td>
<td>356</td>
<td>325</td>
<td>294</td>
<td>253</td>
<td>252</td>
</tr>
<tr>
<td>SGM</td>
<td>31</td>
<td>24</td>
<td>20</td>
<td>16</td>
<td>14.9</td>
<td>12.8</td>
<td>12.7</td>
<td>12.2</td>
</tr>
<tr>
<td>BP</td>
<td>11.5</td>
<td>6</td>
<td>4.1</td>
<td>3.1</td>
<td>2.5</td>
<td>2.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BP-CS</td>
<td>14.6</td>
<td>14.5</td>
<td>14.2</td>
<td>14.1</td>
<td>14</td>
<td>13.7</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2.3: Frames per second (fps) vs disparity range for 512×512 pixel images [73]. The top row (in bold) is disparity range.

Now available and Table 2.2 lists some of these. For all systems, the frame rate $\geq 16$, frame size and disparity range ($\Delta$) are given. Notable techniques in terms of speed are:

- Correlation matching (CM) - Greisen et al., 30fps for 1920×1080 images with $\Delta = 256$ [52].
- SDPS, which gives 30fps for 1024×768 with $\Delta = 256$ [108, 74].
- Semi-Global matching (SGM) developed by Banz et al. 25fps for 1024×768 with $\Delta = 128$ [4].

Kalarot et al. analysed and compared the following algorithms capable of real-time use [73]:

- Symmetric Dynamic Programming Stereo (SDPS)
- Semi-Global Matching (SGM)
- Correlation Matching (CM)
- Belief Propagation (BP)
2. Real-Time Stereo Vision

Figure 2.15: Left images for real-time stereo matching [131, 71].

- Belief Propagation with Constant Space (BP-CS)
- Semi-Global Block Matching (SGBM).

SGBM was evaluated on a CPU whereas all other algorithms were evaluated on a GPU. Therefore, SGBM is excluded here. First, they measured the speed with different disparity ranges (Δ) as shown in Table 2.3 [73]. They also found that CM, SDPS and SGM were the best stereo matching algorithms for real-time use. Table 2.3 also shows similar results to those in Table 2.2, namely that CM is the fastest.

Kalarot et al. also considered the trade-off between the speed and performance measures. They used the Middlebury images Cone, Teddy, Tsukuba...
2.9. Evaluation of Real-Time Stereo Matching

and Venus [73, 131] and a synthetic image set called Fat boy [71]. The left image of each pair is shown in Figure 2.15.

They used RMS error (see Equation 2.25) and percentage of bad pixels metrics (see Equation 2.26). Figure 2.16 shows selected results from Kalarot et al. for RMS error and Figure 2.17 shows percentage of bad pixels [73].

Figure 2.16 and Figure 2.17 show that CM is the worst on all images. For Cones and Teddy, SGM performs much better than SDPS, while on Tsukuba and Venus SGM and SDPS are nearly the same, but on Fatboy SDPS performs much better than SGM.

An additional feature of the SDPS algorithm is that it generates occlusion maps at no additional cost, which can speed up scene interpretation [107]. SPDS operates on corresponding scan lines, while CM with $7 \times 7$ windows matching has to wait for seven corresponding scan lines. Considering both speed (see Table 2.2 and Table 2.3) and matching performance (see Figure 2.16 and Figure 2.17), SDPS generally performs better than any other algorithm.
2. Real-Time Stereo Vision

Figure 2.17: Percent bad pixels for SDPS, CM, SGM, BP and BP-CS.

2.10 Hardware Implementation of SDPS

Evaluation of real-time stereo (Section 2.9) showed the importance of SDPS. Here the main focus is on improving and interpreting depth maps produced by two SDPS hardware implementations - one using FPGA and the other on a GPU.

2.10.1 FPGA Implementation

The FPGA SDPS system was implemented on a Gidel ProcStar III card, containing an Altera Statix III FPGA with Cameralink camera interface. The card interfaced to a host PC by an 8 lane PCI express bus. The system can generate 128 disparity levels [107, 108].

The FPGA implementation generates four outputs and consists of three main blocks:

- Distortion removal and rectification
- Disparity calculators
Figure 2.18: FPGA implementation block diagram (from Morris et al. [108]).

- Predecessor array and backtrack module.

Figure 2.18 shows the main modules with generated output.

**Distortion removal and rectification**

Conventional procedures solve equations for each pixel to remove distortion and to rectify images. Solving equations is simple but slow iterative approach is not suitable for real-time use [67] as around 19 addition and 27 multiplications are required per pixel.

The fastest approach is to use lookup tables for distortion removal and rectification [67]. The contribution made here was to build single lookup tables for removing distortion and rectifying images in one operation. To generate
lookup tables, camera parameters (discussed in Section 2.5.1) are loaded for left and right cameras as shown in Figure 2.19. To generate single $x-$ and $y-$ lookup tables, the $x-$ ($y-$) displacement is calculated for distortion and added to the $x-$ ($y-$) displacement for rectification. However, for larger images ($1024 \times 768$), the raw lookup tables require more memory than available on the Altera Stratix III FPGA [26]. Therefore, the lookup tables are reduced iteratively until the maximum error introduced by interpolation exceeds an acceptable maximum error [67]. For typical configurations used in the lab, a $65 \times 65$ entry lookup table suffices for $1024 \times 768$ pixel images. The lookup tables are saved as VHDL code and synthesized with the rest of the distortion removal and rectification module as shown in Figure 2.18.
Disparity calculator

One disparity block is instantiated for every pair (odd, even) of disparities. Pixels intensities are stored in the pixel shift registers: the current value from the left camera $G^l_x$ up to $G^l_x + \Delta$ and values from right camera $G^r_x$ up to $G^r_x - \Delta$ are required. The disparity calculation block ensures that the correct pixels are available to each disparity calculation block in each cycle. The block calculates six costs - one for each visibility state for each disparity using Equation 2.23. The predecessor values are determined using Equation 2.24 and are stored in the predecessor array $\pi^p$ (More details can be found in Morris et al. [107, 108]).

Predecessor array and backtrack

$\pi^p$ is a large store and a total of $3(w - 1)\Delta$ values are needed. However, because of SDPS transition rules, the values need only 2 bits to encode. If $w = 1024$ and $\Delta = 128$, then the predecessor array requires $8 \times 10^5$ bits - which is easily within the capacity of a modern FPGA.

At the end of each scan line, the minimum cost is identified and its index sent as an address into the backtrack module. The backtrack module works backward along the scan line to build a list of disparities for this scan line.

Output

The FPGA implementation of SDPS produces corrected left and right images, depth maps and occlusion maps. These maps are streamed to the host through PCI express bus. Three frames from the CITR-Bike video sequence are shown in Figure 2.20. Figure 2.20(b) shows that the SDPS generated depth maps suffer from streaks. Since SDPS is dynamic programming (DP) stereo matching algorithm, it is well known that DP based stereo algorithms have streaks in the resulting disparity map. Following are quotes
2. Real-Time Stereo Vision

Figure 2.20: Three frames from a video sequence. Note that, in the Cyclopean view ‘images’ have twice as many pixels per scan line. In the false colour depth map purple represents closer and blue represents farther away. In the occlusion map white represent binocular points.

from Scharstein and Szeliski [132] about dynamic programming disparity maps:

- “Both DP and SO\(^1\) algorithms suffer from the well known difficulty of enforcing inter-scanline consistency, resulting in horizontal streaks in the computed disparity map.”

- “The disparity maps created by the scanline-based algorithms (DP and

\(^1\)SO stand for scanline optimization
2.10. Hardware Implementation of SDPS

SO) are promising and show a lot of detail, but the larger quantitative errors are clearly a result of the streaking due to the lack of inter-scanline consistency.”

2.10.2 GPU Implementation

The GPU implementation of SDPS is based on an nVidia GeForce GTX 280 card, connected to a host CPU via PCI express [72]. The GPU (GTX 280) has 30 multiprocessors with a processor clock of 1296 MHz and 2 GB global memory. This system is described by Kalarot and Morris [74]. Compute Unified Device Architecture (CUDA) was used to program the GPU. The overall structure is similar to the FPGA implementation with rectification, forward pass and back tracking phases [72]. Lookup tables are also used for rectification. The processed scan lines are input to forward pass which continue with same thread allocation and this phase generates predecessor and final cost arrays, as shown in Figure 2.21. The maximum multiprocessor use is achieved by assigning one multiprocessor to one scan line with Δ threads per block. The back tracking module takes inputs from the forward pass and outputs the disparity and occlusion maps [74]. Both GPU and FPGA systems have similar performance; a few minor differences are [72]:

- The cameras are not connected directly to the GPU system whereas in the FPGA system cameras are directly connected to FPGA

- Full lookup tables are used in the GPU system, whereas in the FPGA system reduced lookup tables are used. The use of full lookup tables takes more memory but interpolation computations are not required.

Stereo video sequences used here were generated by our laboratory’s hardware systems. Since they use the same basic algorithm, the outputs from
both systems are essentially identical - showing the same artefacts that this study aims to reduce.

2.11 Summary

This chapter presents the theoretical background of stereo vision systems followed by analysis of real-time stereo, focused on both speed and matching performance. The analysis shows that SDPS generally performs better than other algorithms. There are two hardware implementations of SDPS one on FPGA and the other on GPU. The FPGA SDPS was implemented on a Gidel ProcStar III card, using Altera Statix III FPGA with Cameralink camera inter-
face while the GPU implementation of SDPS is based on an nVidia GeForce GTX 280 card; both are connected to the host CPU using PCI express bus. The SPDS hardware systems stream corrected left and right images depth map and occlusion map. SDPS generate occlusion map at no additional cost and the outputs of both systems are identical.
Chapter 3

Depth Contours

The aim of many computer vision techniques is to outline the shape of an object and reduce visual clutter. Depth contours can play an important role. This chapter proposes algorithm for generating depth contours for SDPS depth maps (produced in real-time). The algorithm has two main steps: (a) reduce search space by marking irrelevant pixels in the centre of regions and (b) generate contours in the reduced search space. Results for high resolution images (approximately 1 M pixels) show that the algorithm is accurate and fast. The algorithm is parallelized to increase the number of contours that can be generated.

3.1 Introduction

Shotton has demonstrated that contours alone suffice for visual object recognition [140, 141]. Contours from texture (intensity or colour) images are used for various tasks, such as object detection, recognition [31, 37, 141] and pose estimation [1, 115]. However, complex textures, illumination changes, shadows and background clutter can create problems for contour extraction from textured images [140].

With stereo systems, object contours can be determined from depth maps. In principle, depth determination is not affected by texture, shadows, background clutter or illumination changes (providing the left and right cameras
Figure 3.1: Intensity contours \textit{vs} depth contours for the Middleburry data set Baby [131]: (a) Left image; (b) Depth map (ground truth) where lighter colour represent near to camera; (c) Detected contours of left image; (d) Detected depth contours.
are synchronized properly). In fact, the additional texture created by shadows, etc. can assist matching. Figure 3.1 shows an example of textured image contours and ideal depth contours for the Middlebury data set Baby [131]. The intensity contours were generated with a Canny edge detector [17] and depth contours were generated by a border following algorithm [14, 148]. In practice depth contours generated from a stereo depth map are noisy, but recently depth contours were used by McCarthy and Barnes for surface extraction [99], by Butt and Morris for object tracking [16], Maldeni et al. for gesture recognition [96], and by Jager for depth map compression [64]. An efficient depth contour extraction algorithm would improve the usability of these techniques.

The problem with both the Suzuki and Abe border following algorithms [148] (available in OpenCV [14]) and the Moore neighbour tracing algorithm (available in Matlab) is the processing time because these algorithms operate on binary images [51]. In order to build contours for \( \Delta \) disparities \( \Delta \) passes over disparity map are required. However, hardware SDPS generates occlusion maps (discussed in Section 2.10) in which every depth value marked as ML, MR or B can be used to reduce (\( \Delta \)) passes over the disparity map. The scan line profiles of SDPS generated depth maps in which for each disparity \( d_x \) neighbouring disparities \( d_{x-1} \) and \( d_{x+1} \) (left and right) are predictable (see Equation 3.1) can help to formulate rules to extract contours.

\[
d_{x+1} = \begin{cases} 
    d_x & \text{if } B \\
    d_x + 1 & \text{if } ML \\
    d_x - 1 & \text{if } MR 
\end{cases} \quad (3.1)
\]

Therefore, this chapter proposes an algorithm for contour extraction from SDPS generated depth maps. For \( \Delta \) disparities the proposed algorithm takes one pass over disparity map to assign a state to each disparity (discussed in Section 3.2.1) which is then used with the disparity map to extract contours.
3. Depth Contours

3.2 Contour Generation

The SDPS based hardware produces two maps: (a) a disparity map and (b) an occlusion map. In the occlusion map, each pixel is marked as:

- ML - visible only by the $L_{camera}$
- B - Binocularly visible
- MR - visible only by the $R_{camera}$

The main steps of the contour generation procedure are:

- State assignment: inspired by the occlusion map to reduce search space for contour generation.
- Contour extraction: based on the principles of local neighbours search to extract contours and uses the rules derived from properties of the SDPS depth map. In a SDPS depth map for each disparity the neighbouring (left and right) disparities are predictable.

3.2.1 State Assignment

This section can be also called C state assignment. Pixels in the centre of regions of uniform disparity do not participate in contours and are marked as ‘central’. A disparity, $D(x, y)$, is already marked in the occlusion map as MR, B or ML [107, 108]. Another state, C, is added to assist in the contour generation, giving four states:

- ML - monocular left points
- MR - monocular right points
3.2. Contour Generation

- B - binocularly visible points
- C - binocularly visible, located in region centres and not needed for contour generation.

The basic idea is that central pixels have the same disparity as their neighbours. Each B, \( D(x, y) \), with disparity \( d \) is examined and classified according to its 8-neighbourhood pixels. Since the SDPS algorithm has already considered the points on the same scan line, \( D(x - 1, y) \) and \( D(x + 1, y) \), six neighbours are checked as follows: if the central point disparity, \( D(x, y) \) is greater than any of its neighbours, \( D(x - 1, y - 1) \), \( D(x, y - 1) \), \( D(x + 1, y - 1) \), \( D(x - 1, y + 1) \), \( D(x, y + 1) \) or \( D(x + 1, y + 1) \) then retain B for \( D(x, y) \), else label it C. The search space is reduced considerably by ignoring C points. Statistics of points in each state are discussed in Section 3.4.1.

3.2.2 Contour Extraction Algorithm

As the C points are marked in the previous step, lists of ML and MR points are generated for disparity \( d \). This facilitates fast location of potential contour points. Then a virtual ‘salmon’ is used to generate contours\(^1\); the Salmon leaves a trail of points behind it as it traverses its path. If \( d_s \) is the disparity for which the Salmon is constructing a contour. The Salmon chooses the first ML point from a list with a disparity \( d_s \) (any ML or MR point with disparity \( d_s \) can be chosen).

If \( p_c \) is the current location of the Salmon and \( d_c \) is the disparity at \( p_c \), as the contour is generated, the Salmon states can be ON – EDGE, INSIDE or OUTSIDE. Table 3.1 defines the states.

The Salmon’s state is a 4-tuple:

\(^1\)“Salmon make an incredible journey downstream from the fresh water where they are born, to the ocean, and then back upstream again as adults, finding the exact location where they began several years earlier.” [113]
\((s, d_s, up, r_{\text{max}})\)

where \(s \in \{\text{ON-EDGE}, \text{INSIDE}, \text{OUTSIDE}\}\) is the Salmon state, \(up\) is either true (tracing MR contour) or false (tracing ML contour) and \(r_{\text{max}}\) is the maximum extent of the Salmon’s sight - a measure of how many neighbouring pixels the Salmon will examine in this iteration. For sight, \(j|j = 1, ..., r_{\text{max}}\), the Salmon will examine \(N = 4j + 1\) neighbours. The direction of each neighbour is:

\[
\phi_0 = \begin{cases} \text{when (up) 0} & \text{else } \pi \\ \phi_i = \phi_{i-1} + \frac{\pi}{4j}, i = 1, 2, \ldots, N \end{cases}
\]

where \(\phi = 0\) represents the direction along the \(x\) axis and \(\phi\) increases in an anti-clockwise direction.

The Salmon’s initial state will be \textbf{ON-EDGE}. The Salmon then follows the \(d_s\) contour and returns to where it started; it uses rules derived from the visibility constraints of SDPS [46, 48]. In the MR state, the Salmon moves up so \(0 \leq \phi \leq \pi\). In the ML state, the Salmon moves down so that \(\pi \leq \phi \leq 2\pi\).

The Salmon uses steps \(j = 1, ..., r_{\text{max}}\) to decide which direction to take. If it can not decide at step \(j\) then it tries step \(j + 1\).

**General rules**

1. Any point in C state or on single line streak is ignored. A point \(p_{x,y}\) is a single line streak point if either \(p_{x,y-1}\) and \(p_{x,y+1}\) or \(p_{x-1,y}\) and \(p_{x+1,y}\) have a disparity less than the disparity at \(p_{x,y}\).

2. A neighbour at \(j\) always has higher priority than neighbours at \(j + 1\).

3. Transition to an \textbf{ON-EDGE} state always has the highest priority.

4. Transition to an \textbf{INSIDE} state always has a priority over transition to an \textbf{OUTSIDE} state.
3.2. Contour Generation

<table>
<thead>
<tr>
<th>State</th>
<th>Definition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON-EDGE</td>
<td>on an edge: $p_c$ is in ML or MR state with $d_c = d_s$</td>
<td>continue along the edge</td>
</tr>
<tr>
<td>INSIDE</td>
<td>inside a contour: $p_c$ is in B state with disparity equal to $d_s$ or $d_c &gt; d_s$</td>
<td>when going (down</td>
</tr>
<tr>
<td>OUTSIDE</td>
<td>outside: $d_c &lt; d_s$</td>
<td>when going (down</td>
</tr>
</tbody>
</table>

Table 3.1: Salmon state definitions.

5. If there are two or more possible transitions to INSIDE neighbours then the neighbour with the closest disparity is chosen.

6. If there are two or more possible transitions to OUTSIDE neighbours then the neighbour with the closest disparity is chosen.

7. If the current state is ON-EDGE or INSIDE then ML or MR is chosen over B with the same disparity.

8. If the current state is OUTSIDE then B is chosen over ML or MR with the same disparity.

Search priority

In each state $s$ the order in which the neighbours are searched is different; the priorities for the Salmon moving down and for $j = 1$ are shown in Figure 3.2.
Figure 3.2: Order in which the neighbours are visited for each of the three Salmon states for a Salmon ‘descending’ \((\pi \leq \phi \leq 2\pi)\) an ML edge. The order is reversed for a Salmon ‘climbing’ \((0 \leq \phi \leq \pi)\) an MR edge.
Expanding the search region

If all the neighbours of a point are marked C or have been visited, then it is necessary to increment $j$ and search in a wider region. To prevent the Salmon from selecting the wrong neighbour, a maximum iteration count is specified. If this count reaches a predefined limit or the Salmon returns to its starting point, the contour will be terminated and added to the contour list.

3.2.3 Salmon Operations

An example run of the Salmon is shown in Figure 3.3. The Salmon starts in the (7, ML) pixel at the top and works its way down, trying to stay on the (7, ML) ‘edge’.

However, this is not always possible: in row 2, the Salmon is ON-EDGE, but there is no (7, ML) neighbour, so the Salmon will use the priority scheme shown in Figure 3.2 and go INSIDE to (8, B) attempting to find another (7, ML) point.

In row 4 at first (7, B) from the right, there is a single line streak so the Salmon overcomes and selects (7, ML) on row 5 instead of (7, B) on the left. Then a sequence of (7, ML) pixels are followed until at row 9 the Salmon is facing a continuous wall of C pixels. It is forced to go INSIDE and swim through a sequence of (7, B) pixels (a horizontal edge), eventually finding a (7, MR) one. At this point, it reverses the direction and ‘climbs’ the (7, MR) edge back to the starting point.

3.3 Evaluation in Challenging Regions

A Salmon can find a number of challenging regions during its journey. Two crucial regions are:

Pixels shared by multiple contours: a region where multiple contours share
Figure 3.3: Example Salmon ‘run’: boxes represent pixels in the disparity and occlusion maps; they are labeled with the disparity and the visibility state after C states have been assigned. The background pattern for each pixel shows the Salmon state as it visits a pixel - see the legend. The Salmon starts with the highlighted (7, ML) pixel at the top, ‘descends’ through ML pixels and climbs back (not shown) through MR pixels to reach its starting point again.
3.3. Evaluation in Challenging Regions

Figure 3.4: Disparity map for scene object with horizontal edges and locations where multiple contours share pixels: the marked regions (a-h) have been expanded in Figure 3.5 to show details.

pixels. For example, for large disparity changes in the vertical direction, multiple contours must pass through a single pixel.

**Horizontal edges:** Objects with horizontal edges also challenge the Salmon algorithm; they show large disparity jumps between monocular pixels and can have large regions of B pixels delineating the contour.

A synthetic disparity map is shown in Figure 3.4 which has multiple regions representing these challenges; the regions are marked in Figure 3.4 with (a)-(h) and Figure 3.5 (a)-(h) shows in detail the behaviour at critical points. In Figure 3.5(a), the $d = 6$ contour enters the **INSIDE** state and then turns left, preferring the (6, B) pixel over the (6, C) pixel (which constitute the centre of
Figure 3.5: Salmon traversal for expanded regions (a-h) marked in Figure 3.4.
3.4 Experimental Results

The experiments were conducted on 2.4 GHz quad core processor with 4 GB of main memory. The code was compiled using Visual C++’s optimizers. For

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Frame count</th>
<th>Disparity range $\Delta_c$</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ball</td>
<td>195</td>
<td>20-50</td>
<td>$2048 \times 768$</td>
</tr>
<tr>
<td>Gujral</td>
<td>115</td>
<td>75-127</td>
<td>$2048 \times 768$</td>
</tr>
<tr>
<td>Bob</td>
<td>101</td>
<td>75-127</td>
<td>$2048 \times 768$</td>
</tr>
<tr>
<td>Dolls</td>
<td>1</td>
<td>33-105</td>
<td>$1390 \times 555$</td>
</tr>
</tbody>
</table>

Table 3.2: Experimental data overview: $\Delta_c$ is the range of disparities for critical contours which are used to outline and model the objects of interest in the scene.
efficient use of a multi-core processor, contour generation was parallelized. In the C state assignment step, the disparity map was divided into \( nc - 1 \) equal vertical regions, where \( nc \) is the number of cores; each core then assigns C states for its allocated region. In the contour generation step, the full disparity range \( \Delta_c \) was divided into equal sub ranges \( \Delta_1, \Delta_2 \ldots \Delta_{nc-1} \) so that core \( i = 1 \ldots nc - 1 \), generated contours for sub range \( \Delta_i \). One master core assigned the work load and synchronized each step.

Performance was analysed using the Ball, Gujral and Bob sequences captured by the SDPS FPGA hardware [107, 108]. The images have \( 1024 \times 768 \) pixels with disparity maps of \( 2048 \times 768 \) resolution (a consequence of the Cyclopæan view discussed in Section 2.7.2). One Middlebury image pair, Dolls [131], was also used. For each sequence, Table 3.2 shows \( \Delta_c \), frame count and resolution of the disparity map and Figure 3.6 shows selected rectified left images from each sequence with false colour disparity maps. Figure 3.7 shows a set of generated Salmon contours for a frame of the Bob sequence with a 3-D projection to show that the generated contours represent the shape of the object.

### 3.4.1 Points Statistics

Table 3.3 shows the number of points for each state MR, ML, B and C (discussed in Section 3.2.1) for Middlebury Dolls and selected frames of Ball,

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Frame #</th>
<th>% C</th>
<th>% B</th>
<th>% ML</th>
<th>% MR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ball</td>
<td>122</td>
<td>98.4</td>
<td>1.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Gujral</td>
<td>97</td>
<td>84.8</td>
<td>11.6</td>
<td>1.8</td>
<td>1.8</td>
</tr>
<tr>
<td>Bob</td>
<td>43</td>
<td>88.5</td>
<td>7.7</td>
<td>1.9</td>
<td>1.9</td>
</tr>
<tr>
<td>Dolls</td>
<td>1</td>
<td>77.7</td>
<td>17.2</td>
<td>2.7</td>
<td>2.4</td>
</tr>
</tbody>
</table>

Table 3.3: Number of points in each state.
3.4. Experimental Results

Figure 3.6: Selected frames from each sequence: rectified left images and false colour disparity maps (purple represents closer (larger $d$) and blue represents farther away (smaller $d$). (a) **Ball** frame # 122; (b) **Gujral** frame # 97; (c) **Bob** frame # 43; (d) **Dolls**.
3. Depth Contours

(a) 2-D projection
(b) 3-D projection

Figure 3.7: Salmon contours for frame # 43 from Bob sequence.

Gujral and Bob sequences. The results show that more than 95% of the points are marked C and ignored in subsequent processing.

3.4.2 Ground Truth Contours

For quantitative validation of generated Salmon contours two types of ground truth contour maps, $D^{ga}$ (all contours) and $D^{gm}$ (maximum length contours), were generated using the Suzuki and Abe border following algorithm [14, 148]. The steps to generate $D^{ga}$ are given in Algorithm 1. $D^{ga}$ was generated for all frames from the sequences (Ball, Gujral, Bob and Dolls) for the disparity range in Table 3.2. $D^{ga}$ is a set of ground truth contours which was used to validate Salmon generated contours.

However, if a ground truth contour is generated as multiple broken contours then points validation only is not enough. Therefore, to further validate selected ground truth contours (as described below) were added to $D^{gm}$, to
3.4. Experimental Results

Algorithm 1 Generate ground truth contour map $D^{ga}$

for all $d$ in disparity range $\Delta$ do

  Threshold disparity map for $d \rightarrow D^t_d$
  Generate contours for $D^t_d$ by border following
  Add generated contours to $D^{ga}$

end for

check that the generated contours match their corresponding ground truth contours. The steps to generate $D^{gm}$ are given in Algorithm 2. $D^{gm}$ was generated for the Gujral and Bob sequences because there is a single large closed contour at the middle of the disparity map for each disparity, which was easily detected. Contours at the middle of the disparity map were selected because contours which cross disparity map borders are closed in Suzuki and Abe border following algorithm [14, 148] whereas the Salmon stops when it reaches the border.

Algorithm 2 Ground truth contour map $D^{gm}$

for all $d$ in disparity range $\Delta$ do

  Threshold disparity map for $d \rightarrow D^t_d$
  Generate contours for $D^t_d$ by border following
  Find the largest closed contour and add it to $D^{gm}$

end for

3.4.3 Accuracy

The accuracy of generated Salmon contours was checked using ground truths $D^{gm}$ and $D^{ga}$. For both ground truths, true positives and false positives were counted using Algorithm 3 and false negatives were counted using Algorithm 4. For each frame from the Ball, Gujral, Bob and Dolls sequences
Table 3.4: Salmon contour validation for all frames in each sequence using ground truth $D^{gt}$. True positive and false positive points were calculated using Algorithm 3 no false negative points were detected using Algorithm 4.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Number of frames</th>
<th>True positive</th>
<th>False positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ball</td>
<td>195</td>
<td>99.8</td>
<td>0.2</td>
</tr>
<tr>
<td>Gujral</td>
<td>115</td>
<td>99.6</td>
<td>0.4</td>
</tr>
<tr>
<td>Bob</td>
<td>101</td>
<td>99.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Dolls</td>
<td>1</td>
<td>99.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

(Table 3.2), $D^{gt}$ was used to check the accuracy of Salmon contours. The percentage of true positives, false positives and false negatives over the whole sequence are presented in Table 3.4 which shows that in these sequences 99% Salmon points are correct. For all sequences false positive points were assessed through Algorithm 3 with condition for four neighbours ($c_i$ matches ground truth contour point or $c_i$ is in 4 neighbours of ground truth contour point); it was found that these points were in four neighbours of ground truth contour points.

**Algorithm 3** True positive and false positive detection

```plaintext
for all Salmon contour points $c_i$ do
    if $c_i$ matches ground truth contour point then
        Increment true positive
    else
        Increment false positive
    end if
end for
```
3.4. Experimental Results

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Number of contours</th>
<th>True positive</th>
<th>False positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gujral</td>
<td>6095</td>
<td>99.2</td>
<td>0.8</td>
</tr>
<tr>
<td>Bob</td>
<td>5353</td>
<td>98.6</td>
<td>1.4</td>
</tr>
</tbody>
</table>

Table 3.5: Contour validation using ground truth $D_{gm}$ for each frame of the Bob and Gujral sequences. No false negative points were detected using Algorithm 4.

To validate the corresponding contours for each frame in Bob and Gujral sequences, the largest close Salmon contours were extracted for each disparity. For the Bob sequence 5353 contours were extracted with more than 12 millions points and 6095 with more than 18 millions points were extracted for the Gujral sequence. These contours were checked against ground truth $D_{gm}$ using Algorithms 3 and 4. Table 3.5 shows that, there are 0.8% percent false positives for Gujral and 1.4% for Bob sequence. These points were assessed and it was found that 0.5% points of Gujral and 1.1% of Bob sequence were in four neighbour of ground truth contour points. The remaining 0.3% points for both sequences was examined for selected frames by matching with the disparity map; it was found that they were due to the single line closed contour connected to a main contour for which the border following algorithm generates two contours while Salmon generates a single contour. No false negative points were detected.

3.4.4 Contour Length Reduction

Salmon overcomes single line streaks which results in shorter contours. To validate this, a large closed contour for each disparity was selected for each frame of the Bob and Gujral sequences (for the same reason mentioned in Section 3.4.3). For each contour, the ‘raw’ length was estimated by counting
3. Depth Contours

Algorithm 4 False negative detection

for all Ground truth contour points $g_i$ do
  if $g_i$ is not matching with Salmon contour point then
    if $g_i$ is not on a single line streak or on a border of contour map then
      if $g_i$ is not in 4 neighbours of Salmon contour point then
        Increment false negative
      end if
    end if
  end if
end for

points generated using the border following algorithm [14, 148]. Figure 3.8 shows the percent of contour length reduction for each frame of the Gujral and Bob sequences. The overall mean reduction was 30% (standard deviation, $\sigma = 1.2$) for the Gujral sequence and, for Bob, the overall mean reduction was 33% ($\sigma = 0.8$). The small differences may be attributed to two factors: in Gujral, there are 3% fewer single line streaks than in Bob but the standard deviation is higher due to different face poses and expressions.

3.4.5 Performance

The performance of the Salmon algorithm was measured by execution time. The times taken by the C state assignment and contour generation using different numbers of cores are shown in Table 3.6 for selected frames. C state assignment is $O(wh)$ for a $w \times h$ image. However, contour generation time is $O(\Delta n)$ where $\Delta$ is the disparity range and $n$ is the number of points in the longest contour. For selected frames (Figure 3.6) the time measured is shown in Table 3.6; the maximum time was 38 ms for the Gujral frame.

For comparison, OpenCV’s implementation of the border following algorithm of Suzuki and Abe [14, 148] was used which uses all available cores.
Figure 3.8: Percent reduction in contour points using Salmon for each frame of the Gujral and Bob sequences. The mean reduction is 30% for Gujral and 33% for Bob.
Table 3.6: Salmon execution time: all times are given in milliseconds on a 2.4 GHz quad core processor with 4 GB of memory. One core was used for task assignment and synchronization.

Table 3.7 compares Salmon with the border following algorithm for selected frames. Salmon is more than six times faster on all frames.

### 3.5 Summary

In this chapter, a novel fast depth contour generation algorithm, Salmon, is proposed which works effectively on noisy depth maps generated by hardware SDPS. The algorithm in the first step reduces the search space and then, in the reduced search space, generates contours. The experimental results show that the algorithm is as accurate as a border following algorithm operating on binary images and at the same time reduces single line streaks. The time to generate contours is proportional to the length of the contour. Salmon generates each contour independently. Therefore, higher throughput was achieved with a parallel implementation using multiple cores. For
3.5. Summary

Table 3.7: Comparison of the Salmon algorithm with border following algorithm. All times in milliseconds.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Frame #</th>
<th>Disparity range (Δ)</th>
<th>Salmon with C state assignment</th>
<th>Border following</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ball</td>
<td>122</td>
<td>20-50</td>
<td>10</td>
<td>181</td>
</tr>
<tr>
<td>Gujral</td>
<td>97</td>
<td>75-127</td>
<td>42</td>
<td>306</td>
</tr>
<tr>
<td>Bob</td>
<td>43</td>
<td>75-127</td>
<td>28</td>
<td>296</td>
</tr>
<tr>
<td>Doll</td>
<td>1</td>
<td>33-105</td>
<td>31</td>
<td>202</td>
</tr>
</tbody>
</table>

a 2.4 GHz quad core processor, Salmon was able to generate the critical contours of a typical image sequence at 24 fps which is six time faster than border following algorithm.

Further improvement of speed could be achieved by incorporating C state assignment directly into the SDPS hardware. Salmon contours are generated independently in parallel providing typically more than 40-fold inherent parallelism - readily exploited in either FPGA or GPU hardware.
Chapter 4

Point-of-Interest Detection

The use of Points-of-Interest can significantly reduce the computational complexity and the memory required for high-level vision tasks. This chapter covers a reliable and effective technique for detecting Points-of-Interest (Triple Edge points) from noisy depth maps generated by a real-time stereo system, which considers all sources of information, i.e. the depth map, the left and right images of a stereo pair and the occlusion map generated by SDPS hardware systems (if available). Experimental results show that the Triple Edge points are a reliable source of information for identifying multiple objects in low-texture scenes with noisy depth maps generated by a real-time stereo system.

4.1 Introduction

Recent developments in real-time stereo vision processing allow us to capture dense depth maps at 30fps for high resolution images (1Mpixel or more), as described in Chapter 2. Real-time depth information in addition to intensity images can be used for a wide range of tasks such as segmentation, object extraction and scene reconstruction.

This chapter focuses on quickly extracting reliable Points-of-Interest in a computationally efficient manner and using the Points-of-Interest rather than the full maps or images. Figure 4.1 shows examples of the Points-of-Interest
4. Point-of-Interest Detection

Figure 4.1: This frame (from CITR2 Sequence discussed in Section 4.5) contains two subjects with bounding boxes. A false colour disparity map, a disparity state map and **Triple Edge** points are shown only for regions inside the bounding boxes for ground truth subject edges.

(fully described later) for selected locations of a frame.

The Points-of-Interest described here are those disparity transition points which coincide with the corresponding left and right image pair edges points. These points are also called **Triple Edge** points because they are the points where three edges coincide (left and right image pair edges with the disparity map edges).
4.2 Related Work

In contrast to the extensive literature on 2-D monocular image Points-of-Interest detection, there has been little research focus on Points-of-Interest detection from 3-D data. The two popular 2-D images Points-of-Interest extracting techniques are Scale Invariant Feature Transform (SIFT) [91, 92] and Speeded Up Robust Features (SURF) [9, 8]. These techniques use local gradients and unique orientation for Points-of-Interest detection.

For 3D data, Mian et al. discuss a Points-of-Interest detection technique using depth maps from range images only [103, 102]. Viksten et al. also use range data to detect Points-of-Interest with an extended Harris corner detector [155]. Steder et al. [144] considered only range images and they used a Laplacian of Gaussian based method for Points-of-Interest detection. However, this technique is not suitable for real-time performance, due to its computational cost. Steder et al. [145] used a technique which first classifies object borders and then locates Points-of-Interest. However, it is designed for range data and is computationally expensive (Steder et al. [144, 145] techniques computation time is compared with the computation time of technique discussed in this chapter in Section 4.5.3). Flint et al. [33] described distinctive scene features and Rusu et al. [126] identify rigid objects in indoor environment using a laser range finder. Holzer et al. used Point-of-Interest detection technique from Kinect data [61], Rosten and Tom [123, 124], Rosten et al. [125], Lepetit et al. [88], Stuckler and Sven [146] and Shotton et al. [139] used learning based techniques, which need training and these techniques are not used for full stereo depth maps. Blum et al.[11] processed RGB images with Kinect depth maps for feature descriptors only; they relied on other Point-of-Interest extraction methods such as that of Steder et al. [145].

However, all of these techniques are either for laser range finder or depth maps from Kinect. As pointed out in Chapter 1, laser range finders are ex-
4.3 Triple Edge Points Detection

To detect Triple Edge points (introduced in Section 4.1), a rectified left image \((I_L)\), right image \((I_R)\) and disparity map \((D)\) are required. The first step is to assign states to \(D\) pixels - leading to the disparity state map, \(D^S\). Then edges in \(I_L\) and \(I_R\) are detected to generate the left and right edge maps, \(I_{Le}^L\) and \(I_{Re}^R\). Finally \(D, D^S, I_{Le}^L\) and \(I_{Re}^R\) are used to detect Triple Edge points - see Figure 4.2. State assignment and edge marking are independent and can run in parallel.

4.3.1 Disparity State Assignment

The hardware implementation of SDPS (discussed in Section 2.10) generates an occlusion map in the hardware; in this case we do not need to scan \(D\) again to generate \(D^S\). However, when it is necessary to generate the state
4.3. **Triple Edge Points Detection**

map $D^S$ from $D$, adjacent horizontal pixels are considered. $d_x$ represents a disparity at position $x$, in an arbitrary row of the disparity map $D$: adjacent disparities, $d_{x-1}$ and $d_{x+1}$, on the same scan line are considered. $s_x$ represents the state in $D^S$, chosen from the set $\{B, M\}$, where $B$ denotes binocularly visible and $M$ monocular points.

The state assignment rule is:

$$s_x = \begin{cases} B & \text{if } d_x = d_{x-1} = d_{x+1} \\ M & \text{otherwise} \end{cases} \quad (4.1)$$

The hardware implementation of SDPS generates disparity states $\{B, ML, MR\}$ in the occlusion map [108, 74]: map $ML \rightarrow M$ and $MR \rightarrow M$.

In the next step of **Triple Edge** point marking, all $B$ points are ignored (Table 4.5 shows the number of $M$ points for four sequences discussed in Section 4.5).

4.3.2 **Edge Detection in Left and Right Images**

For edge detection ‘raw’ $I^L$ and $I^R$ were used without any smoothing using the $1 \times 3$ convolution kernel,

$$K = \begin{bmatrix} +1 & 0 & -1 \end{bmatrix}$$

which responds to vertical edges with an edge threshold $g^e$. The algorithm used to compute $I^{Le}$ and $I^{Re}$ is not critical: several algorithms such as Prewitt, Sobel, Roberts, Laplacian of Gaussian, Canny and other [51, 121] could be used to detect edges.

4.3.3 **Triple Edge Points Detection Algorithm**

The **Triple Edge** point detection algorithm operates on $D^S$. It ignores $B$ points, but if the point is $M$ then the disparity information is used to cal-
4. Point-of-Interest Detection

Figure 4.3: Lens blur estimation using calibration image: (a) centre of square detection; (b) blur estimation using detected centres.

calculate the expected locations of the edges in $I_{Le}$ and $I_{Re}$. The calculated locations are searched over a horizontal window of $s^\epsilon$ pixels in both directions. This allows for:

a) the lens blur that all real cameras exhibit,

b) motion blur and

c) the displacement of the $D$ edge due to the smoothing component of stereo matching algorithm.

The value of $s^\epsilon$ should be guided by factors (a), (b) and (c). If the edge found before searching $s^\epsilon$ pixels then search stop. Therefore, we need to adjust $s^\epsilon$ to maximum blur. For experiments $s^\epsilon = 4$ was used which was estimated as follow:

a) **Lens blur estimation**

Even at the 'best focus' point in the static scene, a strong edge is blurred over several pixels generating a sigmoid-like intensity profile along a scan line
4.3. **Triple Edge Points Detection**

<table>
<thead>
<tr>
<th>Frame number</th>
<th>Left camera</th>
<th></th>
<th>Right camera</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Standard deviation</td>
<td>Mean</td>
<td>Standard deviation</td>
</tr>
<tr>
<td>1</td>
<td>4.7</td>
<td>0.6</td>
<td>4.6</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>4.3</td>
<td>0.5</td>
<td>4.6</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>4.2</td>
<td>0.5</td>
<td>4.5</td>
<td>0.6</td>
</tr>
<tr>
<td>4</td>
<td>4.0</td>
<td>0.5</td>
<td>4.3</td>
<td>0.5</td>
</tr>
<tr>
<td>5</td>
<td>4.2</td>
<td>0.6</td>
<td>4.7</td>
<td>0.6</td>
</tr>
<tr>
<td>6</td>
<td>4.7</td>
<td>0.7</td>
<td>4.7</td>
<td>0.5</td>
</tr>
<tr>
<td>7</td>
<td>4.8</td>
<td>0.6</td>
<td>4.8</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 4.1: Estimated lens blur for chequerboard images.

which crosses a scene edge [29, 89, 90]. To estimate the lens blur, calibration chequerboard images (discussed in Section 2.5.1) were used. Main steps of the procedure are:

1. Detect corners of squares (inner squares only).
2. Find centre of square using the detected corners.
3. Estimate blur using adjacent squares centre.

See Figure 4.3(a) illustrate step 1 and step 2 and Figure 4.3(b) illustrate step 3. For blur estimation (step 3) vertical gradients was computed using kernel $K$ (discussed in Section 4.3.2) and then gradient peak is selected using adjacent square centers. The peak is used to estimate blur by counting the number of edge pixels in both directions. For each adjacent squares five blur values were estimated to minimize the chances of outlier. Therefore, for a chequerboard image $7 \times 6$, 125 values were estimated for each image of the left and right cameras. Table 4.1 shows mean and standard deviation for selected calibration images; frame # 4 was at best focus (minimum blur). The maximum number of blurred pixels for both cameras is 5 pixels.
b) Motion blur estimation

Several techniques are available to estimate and compensate motion blur using captured images. They can be grouped into two categories: single image (blind) estimation techniques which depend on only one image [22, 54, 163, 164] and sequence estimation techniques which depend on more than one image [111, 150, 159]. The aim here is not to compensate for motion blur which can be computationally expansive because we have two high resolution images (left and right). Instead, the aim here is to use correct configuration for experiments (stereo configurations and specifications of cameras are discussed in Appendix A) with minimum motion blur and to know the number of pixels affected by it.

Image resolution ($w \times h$), exposure time ($\delta_{et}$), pixel size ($\rho$), focal length of lens ($f$), relative speed at which an object is moving ($\nu_o$), size of object ($s_o$), direction in which motion is occurring relative to camera ($\phi_o$) and depth of object ($Z_o$) are used to estimate the motion blur. Motion blur is calculated for static camera and moving object. The values of parameters are given in Table 4.2 where the motion blur is calculated for experiments. The object size ($s_o$) is

<table>
<thead>
<tr>
<th>Parameter</th>
<th>value/s</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>image resolution ($w \times h$)</td>
<td>1024 × 768</td>
<td>pixels</td>
</tr>
<tr>
<td>Exposer time ($\delta_{et}$)</td>
<td>3</td>
<td>ms</td>
</tr>
<tr>
<td>Pixel size ($\rho$)</td>
<td>4.56</td>
<td>µm</td>
</tr>
<tr>
<td>Focal length ($f$)</td>
<td>6, 9, 16, 25</td>
<td>mm</td>
</tr>
<tr>
<td>Object size ($s_o$)</td>
<td>1.7 × 0.8 × 0.8</td>
<td>m</td>
</tr>
<tr>
<td>Object speed ($\nu_o$)</td>
<td>1.6, 5.6</td>
<td>m/s</td>
</tr>
<tr>
<td>Object depth ($Z_o$)</td>
<td>5 ≤ $Z_o$ ≤ 70</td>
<td>m</td>
</tr>
<tr>
<td>Motion direction ($\phi_o$)</td>
<td>0, 45, 90</td>
<td>degrees</td>
</tr>
</tbody>
</table>

Table 4.2: Parameters for motion blur estimation.
4.3. **Triple Edge Points Detection**

Figure 4.4: Number of motion blurred pixels for $\varphi_o = 0^\circ$.

(a) $\nu_o = 1.6\, m/s$

(b) $\nu_o = 5.6\, m/s$
Figure 4.5: Number of motion blurred pixels for $\varphi_0 = 45^\circ$. 

(a) $v_0 = 1.6m/s$

(b) $v_0 = 5.6m/s$
4.3. **Triple Edge Points Detection**

Figure 4.6: Number of motion blurred pixels for $\varphi_o = 90^\circ$.

(a) $\nu_o = 1.6m/s$

(b) $\nu_o = 5.6m/s$
1.7 × 0.8 × 0.8m moving with constant speed of 1.6m/s and 5.6m/s in three directions at angles 0°, 45° and 90° with the principal axes. For example, box (1.7 × 0.8 × 0.8m) moving on conveyor belt with speed of 1.6m/s and 5.6m/s in three directions (0°, 45° and 90°). First the distance covered by moving the object in exposure time (δ_et) is calculated for each ν_o and ϕ_o; then the resultant effected number of motion blur pixels is calculated using similar triangles (using similar geometry to that given in Figure 2.6 for single camera).

The estimated motion blur pixels are shown as follows: Figure 4.4 for ϕ_o = 0°, Figure 4.5 for ϕ_o = 45° and Figure 4.6 for ϕ_o = 90° for both ν_o = 1.6m/s and ν_o = 5.6m/s. As expected from the geometry (Figure 2.6) highest number of pixels are affected by ϕ_o = 90° (motion orthogonal to the principal axis). From the estimated motion blur (Figure 4.4, Figure 4.5 and Figure 4.6) we can select the correct configuration; if we are using se = 4, it can cover for up to 8 pixels motion blur by searching in both directions.

c) Stereo matching algorithm smoothing

Smoothing components of cost functions often result in disparity map edges which are shifted relative to the corresponding left and right image edges. Scan line profiles from randomly selected disparity maps which were not affected by noise (visually) were compared with corresponding profile of ILε and IRε to find the shift by comparing the edges peak as shown in Figure 4.7 for two selected profiles. Twenty tests were performed and the shift value between peaks was ≤ 6 pixels in all tests.

Triple Edge points detection

Let (x, y) be the pixel coordinates in D and DS. For rectified images, y is same for ILε and IRε. Thus for each pixel px,y only corresponding xL in and xR, the pixel locations for ILε and IRε are required. The calculation of xL and
4.3. *Triple Edge Points Detection*

Find shift between corresponding edges: selected normalized scan line profiles: $I^L, I^R, D, I^{Le}$ and $I^{Re}$ (frame # 60 CITR2 sequence). The peak on each edge as shown by black dot—see (a) and (b). Corresponding peaks were compared to find shift.

$x_R$ depends on the algorithm used, e.g. SDPS generates Cyclopæan disparity maps, which are either the same width as the original image [48] or double width (SDPS hardware [107, 108]): the appropriate expression from Table 4.3
4. Point-of-Interest Detection

is used to calculate $x_L$ and $x_R$.

The next step looks for edges in $I^{Le}$ and $I^{Re}$ by searching through an $s^e$ pixel window on either side of the predicted location. If matches (edges in the
4.4 Object Based Ground Truth Generation

Generating ground truth datasets for outdoor dynamic scenes is a non-trivial task [41]. Structured light techniques [130] are not useful due to uncontrolled conditions; the laser range finder based technique used by Geiger et al. used for KITTI ground truth datasets have problems of low resolution, sparseness, synchronization of stereo cameras and laser range finder. The high price of a laser range finder is also one of the point of concern which detract its use [41, 100].

Here the object based ground truth generation technique using existing image pairs with a global stereo matching algorithm (graph-cut [82]) is used. The steps to generated ground truth are:

1. Generate disparity map using graph cut [82].
2. Manually mark object border on left or right image.
3. Extract object border.

### Table 4.3: Calculating $x_L$ and $x_R$ from $x$.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$x_L$</th>
<th>$x_R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D$ double size Cyclopæan [107]</td>
<td>$(x + d)/2$</td>
<td>$(x - d)/2$</td>
</tr>
<tr>
<td>$D$ same size Cyclopæan [48]</td>
<td>$x + d/2$</td>
<td>$x - d/2$</td>
</tr>
<tr>
<td>$D$ corresponding to $I^L$</td>
<td>$x$</td>
<td>$x - d$</td>
</tr>
<tr>
<td>$D$ corresponding to $I^R$</td>
<td>$x + d$</td>
<td>$x$</td>
</tr>
</tbody>
</table>

disparity map and left and right images are consistent) are found in left and right image, the point is marked as **Triple Edge**, otherwise it is rejected. The flowchart of this simple and effective algorithm is shown in Figure 4.8.
4. Add the extracted border to generated disparity.

5. Extract object disparity for the added border.

Example of the technique is shown in Figure 4.9. Two Middlebury ground truth datasets, Cone and Dolls, were used with selected objects as shown in Figure 4.10 for validation. The percentage of calculated bad pixels (Section
2.8 Equation 2.26), for Cones is 0.19% and for Dolls is 2.33% with bad pixel threshold $\tau = 2$ using graph-cut stereo matching [82].

### 4.5 Experimental Results

Three sequences, CITR1, CITR2 and CITR3, were captured by SDPS GPU hardware (discussed in Section 2.10). The sequences show from one to five people walking around a flat courtyard with more than 1000 frames. One of the recently published KITTI Vision Benchmark Suite stereo sequences [41] was also used. For this sequence, $D$ was generated by using Semi-Global Matching (SGM) [58].
Summary data for all sequences is shown in Table 4.4. Figure 4.11 shows detail analysis of the CITR2 sequence. It contains two walking subjects, ST and SK. ST and SK are initially stationary at approximately the same distance, 12.5 m, from the camera. Then they start walking at the same speed, ST reaches 17.9 m, while SK reaches 19.6 m. Then they turn and return towards the camera.

Figure 4.11(a) shows, for each frame, the distribution of Triple Edge points $\text{vs}$ disparity (which is converted to depth - see the table in Figure 4.11(a)). In Figure 4.11(a) there is one peak initially, because both ST and SK are at the same distance, but in frames 214-400, we see two peaks because ST and SK are at different distances. However, in the re-projection on to the XZ-plane (Figure 4.11(b)), one peak of Triple Edge points in Figure 4.11(a) clearly separates into two (ST and SK).

For validation, object based ground truth (as discussed in Section 4.4) for every 100th frame of the CITR2 sequence was generated, by using a graph-cut algorithm (GC) [82]. The ground truth depth map is much smoother due to background removal (as shown in Figure 4.9). However, no background was removed from the real-time (SDPS generated) depth maps. Figure 4.12 com-
4.5. Experimental Results

Figure 4.11: CITR2 sequence (400 frames) contains two walking persons, ST and SK, marked in (c). (a) **Triple Edge** points depth distribution for each frame of the sequence; (b) **Triple Edge** points occupancy over the sequence and (c)-(e) Selected frames from the sequence.
pares the normalized disparity distribution for the ground truth with that for Triple Edge points. For better visualization, the distributions are scaled.

### 4.5.1 Significant Triple Edge Points

Figure 4.11(b) shows significant Triple Edge points projected onto the ground (XZ) plane over the whole sequence in which ST and SK are well separated. To determine significant Triple Edge points, a point distributions is considered (see Figure 4.11(a)) and regions of interest are isolated by identifying ranges of distances which contain significant numbers of Triple Edge points; threshold $\tau_s$ is used for this. In experiments $\tau_s = 100$ points per disparity value was used. Note that some static objects (i.e. the wall in the CITR sequences) show significant Triple Edge point counts. For a tracking application, these can be readily removed by background subtraction or identifying regions which are not of interest; this latter criterion was used to generate the significant Triple Edge points here. When several objects appear at similar distances, clusters in the XZ plane (see Figure 4.11(b)) can be used to identify and label individual objects. Clusters which contain few points can be rejected. Clustering algorithms are computationally intensive ($k$-means clustering is in general NP-hard, although $O(nkl)$ algorithms are known [65]) and thus hard to realize in real-time on full images or depth maps. However, since the Triple Edge points are a fraction (typically $\sim 1\%$, see Table 4.5) of the total points, real-time handling of large numbers of objects becomes feasible.

Results from the whole process for selected frames are shown in Figure 4.13 for CITR1, in Figure 4.14 for CITR2 and in Figure 4.16 for KITTI sequence. Figure 4.15 shows Triple Edge points for selected frames of the cluttered scene. Significant numbers of Triple Edge points are obvious for all subjects, including partially occluded ones.
4.5. Experimental Results

Figure 4.12: Normalized disparity distributions for GT and Triple Edge points (TE). The legend in (a) applies to all graphs (b-e).
Figure 4.13: Selected frame #174 from the CITR1 sequence: (a) left image; (b) disparity state map ($D^S$); (c) Triple Edge points; (d) significant Triple Edge points.

4.5.2 Triple Edge Points Count

Table 4.5 shows the average percent of Triple Edge points count: for CITR1, average Triple Edge point count is less than 1% of the total number of points ($1024 \times 768$); for CITR2, it is 1.4%; for the cluttered sequence CITR3, it is 2.6%. So when considering multiple frames, we can dramatically reduce the information to be stored by storing only the reliable Triple Edge points, e.g. for the CITR3 sequence, a raw data set ($I^L$, $I^R$, $D$) requires 3 MB compared to 120 KB for the Triple Edge points (cf. Table 4.5).
4.5. Experimental Results

Figure 4.14: Selected frame #40 from the CITR2 sequence: (a) left image; (b) disparity state map ($D^5$); (c) Triple Edge points; (d) significant Triple Edge points.

4.5.3 Performance

For the experiments a 2.4 GHz Intel Quad Core with 4 GB RAM was used. First we need to detect edges on left and right images for **Triple Edge** points detection. Edge detection time depends on the algorithm; Rao *et al.* have compared time complexities for some algorithms [121]. For the CITR sequences edge detection requires 5 ms and **Triple Edge** point detection requires 7 ms using a single core for more than one million points. Steder *et al.* [144] approach takes 1.02 s (including the time of range image creation) for 150,000-200,000 points. The faster approach discussed by Steder *et al.* [145]
Figure 4.15: Selected frames from the CITR3 sequence: (a) left image; (b) disparity state map ($D^S$); (c) Triple Edge points; (d) significant Triple Edge points.
4.6 Summary

Conventional techniques of Points-of-Interest detection from depth map are reviewed in this chapter. These techniques are not useful for current real-time stereo systems due to the amount of noise in the generated depth map. A fast Points-of-Interest detection technique is proposed which uses all available information (left image, right image and depth map) to detect **Triple Edge** points. The term **Triple Edge** points is used because the detected points are on edges in all three sources. Experiments shows that **Triple Edge** points give us a compact representation of the scene from all sources. Detecting

Figure 4.16: Selected frame #15 from the KITTI sequence: (a) left image; (b) disparity state map ($D^S$); (c) **Triple Edge** points; (d) significant **Triple Edge** points.

takes 23 ms for border detection and 27 ms for interest point extraction for a 115061 point cloud on an Intel I7 Quad Core.
### Table 4.5: Triple Edge points for each sequence.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Average M points per frame</th>
<th>Triple Edge points per frame</th>
<th>% Triple Edge points per frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>CITR1</td>
<td>44192</td>
<td>2477</td>
<td>0.3%</td>
</tr>
<tr>
<td>CITR2</td>
<td>60411</td>
<td>10631</td>
<td>1.4%</td>
</tr>
<tr>
<td>CITR3</td>
<td>70420</td>
<td>20562</td>
<td>2.6%</td>
</tr>
<tr>
<td>KITTI</td>
<td>26823</td>
<td>8451</td>
<td>1.8%</td>
</tr>
</tbody>
</table>

Triple Edge points takes of the order of 13 ms for high resolution images (1024 × 768).
Chapter 5
Contour Refinement

Depth contours outline shapes and are thus key features in object modelling, detection and recognition. However, depth contours generated from a raw disparity map are often noisy. Usually contours are improved by refining the disparity map before generating contours. However, this can distort or remove valuable fine detail in the disparity map. A novel depth contour refinement technique is proposed here which can improve the generated object contours while preserving fine details.

5.1 Introduction

Depth contours are valuable sources of information for 3-D object detection, modelling and recognition. Salmon, a fast algorithm for contour generation, was discussed in Chapter 3. However, the contours generated from ‘raw’ disparity maps inherit noise from the disparity map. For example, the contours in SDPS generated disparity maps contain streaks, which decrease their usability.

One way to refine these contours is to pre-process the disparity map before generating contours. Some of the common disparity map refinement techniques which can improve the generated contours are: median filtering [132], morphological open operators [16], disparity voting [93] and left to right consistency checks [25, 35]. However, if the disparity map contains fine detail
Figure 5.1: Contours generated from the Middlebury Dolls image: (a) original left image; (b) false colour disparity map; (c) ‘raw’ contours (for region marked by white rectangle in b); (d-f) contours after median filtering. Note that fine details are clear in the raw contours whereas median filtering distorts the contours. Short length contours are shown in black (examine the eyes and mouth contours).
these techniques can remove or distort it. For example, Figure 5.1 shows (a) the left ‘Dolls’ image from the Middlebury data set [131] with (b) false colour ground truth depth map and (c)-(f) contours for the region outlined by the white rectangle in (b); (c) contains some fine detail (mouth and eyes) but also some noise on the head and neck. Figure 5.1(d)-(f) shows the contour maps after disparity refinement by median filters with $3 \times 3$, $5 \times 5$ and $7 \times 7$ windows. The median filter smoothes the generated contour and removes noise. However, the fine details are also affected (see Figure 5.1(d)-(f)). With the $7 \times 7$ window (see Figure 5.1(f)), significant fine details have disappeared.

In high resolution stereo systems, such as the University of Auckland’s Photogrammetry Laboratory SDPS hardware, the expected depth accuracy is high ($\Delta \geq 128$) [108, 72]. However, the generated data is noisy (Figure 5.2(a) shows an example of contour generated from ‘raw’ data) as preprocessing the whole disparity map can reduce its accuracy by removing the fine detail. Therefore, there is a need for contour map refinement techniques which can improve depth contours while preserving fine detail.

The contour refinement techniques discussed here are helpful in overcoming the streaks inherent in the SDPS generated raw contours. Two algorithms are proposed: (a) one based on the visibility constraints in SDPS generated disparity map and (b) one based on Triple Edge points (discussed in Chapter 4). For evaluation, a streak point counts (introduced here), and a contour mapping measure [109] were used as error metrics. Results show that both algorithms significantly reduce the streaks in contours and do not reduce the contour mapping measure.

5.2 Contour Refinement

Streaks are well known artefacts of dynamic programming based stereo matching [132]. Figure 5.2(a) shows one contour generated from an SDPS disparity
Figure 5.2: Ground truth contour (red) with raw and refined contour (blue). from sequence CITR1 frame # 260.
map and demonstrates the seriousness of streaks in generated contours. Contour points can be moved horizontally in order to refine the contour. One simple refinement uses a sliding window (segment) around the local neighbourhood of each point. However, we must choose the operator to apply over the segment, e.g. median, mean, minimum (min), maximum (max) or any other. The streaking artefact of SDPS is evident in Figure 5.2(a). The contour was processed by moving a $1 \times 15$ segment and selecting the median, mean, minimum and maximum value within the segment, as shown in Figure 5.2(b)-(e). As expected, with the mean, streaks are added to each contour and the whole contour is distorted (see Figure 5.2(b)). The median operator is effective if the streak represents less than half of the points in the segment. However, as Figure 5.2(c) shows, the median operator can also be affected by streaks. The minimum and maximum operators (Figure 5.2(d) and (e)) show large systematic errors but some clear improvements can be observed: after applying the minimum operator (Figure 5.2(d)), for points in the MR region, the contour is better than it is after median (Figure 5.2(c)) or mean (Figure 5.2(b)) operators, but for points in the ML region then the contour is affected by streaks. On the other hand, the maximum operator shows the opposite behaviour (Figure 5.2(e)). Similar behaviour was observed when processing contours from different sequences (discussed in Section 5.6). Therefore, processing a contour by minimum operator in the MR region and by maximum operator in the ML region can reduce streaks in the object contours (ML and MR points are discussed in Section 2.7.2 and illustrated in Figure 2.14).

5.3 Proposed Algorithms

This section describes two algorithms for contour refinement that use minimum and maximum operators and Triple Edge points to overcome streaks in the SDPS generated depth contours.
Let $\Omega$ represent the set of contours for disparity map, $D$, and $\Gamma^B_j$ represent the $j^{th}$ 'raw' contour:

$\Gamma^B_j = \Omega(d, j)$

where $d$ is the disparity ($d_{\text{min}} \leq d \leq d_{\text{max}}$), $j$ is the contour index ($0 \leq j \leq n^c$) and $n^c$ is the contour count at $d$ and $\Gamma^B_j$ represents a sequence of contour points:

$\Gamma^B_j = b_1, b_2, ..., b_m$

where $b_k$ represents a single point ($1 \leq k \leq m$) in the contour and $m$ is the number points in the contour $\Gamma^B_j$. The $j^{th}$ contour selection is discussed in Section 5.6.

$b_k = (b^x_k, b^y_k)$

where $b^x_k$ is the $x$ coordinate and $b^y_k$ the $y$ coordinate of contour point $b_k$. The segment length for point $b_k$ is the number of points along the segment which the algorithm searches over. If $w_r$ is the number of points on each side of $b_k$ in a segment of the contour centred on $b_k$ within the range, then the contour segment length is $w_r \times 2 + 1$ (used as segment length in Section 5.6), the total number of contour points processed for each $b_k$. $\Gamma^A_j$ represents the $j^{th}$ processed contour:

$\Gamma^A_j = a_1, a_2, ..., a_m$

### 5.3.1 The Mn-Mx Algorithm

For each point $b_k$, the Mn-Mx\(^1\) algorithm first determines whether it is in an ML region or MR region. If $b_k$ is in the MR region, it chooses the $x$-coordinate for the refined contour point, $a_k$, by selecting the minimum $x$ of all the points in the $2w_r + 1$ point segment centred on $b_k$:

$a^x_k \leftarrow \min(b^x_k - w_r, b^x_k - (w_r - 1) ... b^x_k + w_r)$

\(^1\)The term Min-Max is used elsewhere for different algorithms, e.g. statistical inference methods [87, 138]. Therefore, I use Mn-Mx to differentiate the algorithms discussed here.
otherwise it selects the maximum. The steps are set out in Algorithm 5. Figure 5.3 shows the results for a selected contour.

**Algorithm 5 Contour refinement using Mn-Mx**

**Input:** A $j^{th}$ 'raw' contour $\Gamma_j^B$

**Output:** A process contour $\Gamma_j^A$

for all points $b_k$ in $\Gamma_j^B$ do

if $b_k$ is in MR region then

$\alpha_x^k \leftarrow \min (b_x^k - w_r, b_x^k - (w_r - 1), \ldots b_x^k + w_r)$

else

$\alpha_x^k \leftarrow \max (b_x^k - w_r, b_x^k - (w_r - 1), \ldots b_x^k + w_r)$

end if

$\alpha_y^k \leftarrow b_y^k$

Add point $a_k$ to contour $\Gamma_j^A$

end for
Algorithm 6 Contour refinement using Triple Edge Mn-Mx

Input: A \( j \)th ‘raw’ contour \( \Gamma_j^B \)

Output: A process contour \( \Gamma_j^A \)

for all point \( b_k \) in \( \Gamma_j^B \) do

if \( b_k \) is Triple Edge point then

\( a_k \leftarrow b_k \)

else

\( t^x_L \leftarrow \) find Triple Edge point \( (b^x_{k-1}, b^x_{k-2}, ..., b^x_{k-w_r}) \)

\( t^x_R \leftarrow \) find Triple Edge point \( (b^x_{k+1}, b^x_{k+2}, ..., b^x_{k+w_r}) \)

if both \( t^x_L \) and \( t^x_R \) exist then

interpolate \( a^x_k \) from \( t^x_L \) and \( t^x_R \)

else if either \( t^x_L \) or \( t^x_R \) exist then

\( a^x_k \leftarrow t^x_L \) or \( t^x_R \)

else if \( b_k \) is in MR region then

\( a^x_k \leftarrow \min (b^x_{k-w_r}, b^x_{k-(w_r-1)}, ..., b^x_{k+w_r}) \)

else

\( a^x_k \leftarrow \max (b^x_{k-w_r}, b^x_{k-(w_r-1)}, ..., b^x_{k+w_r}) \)

end if

\( a^y_k \leftarrow b^y_k \)

end if

Add point \( a_k \) to contour \( \Gamma_j^A \)

end for

5.3.2 The Triple Edge Mn-Mx Algorithm

This algorithm uses Triple Edge points with the Mn-Mx algorithm (Algorithm 5). As mentioned in Chapter 4, Triple Edge points are consistent points from all sources of SDPS hardware generated data [108, 72]. Therefore, for each point \( b_k \), the algorithm first determines whether it is a Triple Edge point
5.3. Proposed Algorithms

Figure 5.4: **Triple Edge Mn-Mx** Algorithm flowchart. For further detail see Algorithm 6. Conditions C1, C2, C3 and C4 refer to the graphs shown in Figure 5.5 and Figure 5.12.
Figure 5.5: Contour refinement with **Triple Edge Mn-Mx** (Algorithm 6). The \( x \) values for points; that will be chosen if the conditions C1, C2, C3 and C4 with the same label in the flowchart (Figure 5.4) are met. For C4 the \( \min \) value is chosen because \( b_k \) is assumed in an MR region: in an ML region, the \( \max \) value would be chosen.

or not. If the point is a **Triple Edge** one, then no operation is required. Otherwise, it searches \( w_r \) points along a segment of the contour centred on \( b_k \) to find **Triple Edge** points, \( t_l \) and \( t_r \). If both \( t_l \) and \( t_r \) exist, then it interpolates to refine \( b_k \). If only one of \( t_l \) or \( t_r \) exists then it selects it. If there are no nearby **Triple Edge** points, the **Mn-Mx** algorithm is used to refine the contour. The steps of the algorithm are given in Algorithm 6 and Figure 5.4 shows a flowchart of the algorithm. Figure 5.5 shows a contour segment showing the points replacing \( b_k \) for each condition of Algorithm 6. In the
ideal case, if all contour points are **Triple Edge** points, then this algorithm (Algorithm 6) will not change any point while the **Mn-Mx** algorithm (Algorithm 5) for each points will still select either the min or max point from \( w_r \) points on each side.

## 5.4 Evaluation Metrics

The evaluation metrics commonly used for disparity maps are bad pixel count and RMS error (discussed in Section 2.8), but they are not useful for contour evaluating improvement. Other error measures used in segmentation, e.g. region based error measures [38], boundary based error measures [105] and mixed error measures [162, 109], are not sensitive to streaks, wiggles and some large shape features [109]. Therefore, the following error metrics are used to evaluate contour improvement:

- Streak Points Count
- Contour Mapping Measure.

### 5.4.1 Streak Points Count

Streaks are the main artefact of dynamic programming stereo matching [132] and literature review found no suitable error metric that can quantify streaks. A new error metric, namely streak points count (SPC), is introduced here. SPC is the raw measure of the number of contour points which lie outside a ground truth bounding box. The percent SPC is calculated as a fraction of total contour points. Ground truth bounding box generation which we need for SPC is discussed in Section 5.5.
5. Contour Refinement

(a) Example contours, $\Gamma^A$ and $\Gamma^G$

Figure 5.6: Mapping between two contours: $\Gamma^A = a_1, a_2, a_3, a_4, a_5$ and $\Gamma^G = g_1, g_2, g_3$ and their correspondence map $\hat{M} = a_1 \leftrightarrow g_1, a_2 \leftrightarrow g_2, a_3 \leftrightarrow g_2, a_4 \leftrightarrow g_3, a_5 \leftrightarrow g_3$. The trace is $\hat{T} = z(1,1), (2,2), (3,2), (4,3), (5,3)$ and the distances are $\gamma(\hat{M}) = \text{dist}(a_1, g_1) + \text{dist}(a_2, g_2) + \text{dist}(a_3, g_2) + \text{dist}(a_4, g_3) + \text{dist}(a_5, g_3)$. 

(a) The mapping graph
5.4.2 Contour Mapping Measure

The contour mapping measure as an error metric was introduced by Movahedi and Elder [109]. The procedure is similar to elastic contours matching methods [43, 34, 134, 133] which minimize the matching cost to align two contours.

Let $\Gamma^G$ be a ground truth contour and $\Gamma^A$ a processed contour. $\tilde{M}$ is the mapping sequence, $\tilde{M} = m_1, m_2, \ldots, m_h$ between $\Gamma^G$ and $\Gamma^A$. The cost of sequence $\tilde{M}$, is:

$$\gamma(\tilde{M}) = \sum_{i=1}^{h} \gamma(m_i) \quad (5.1)$$

where $\gamma(m_i)$ is the Euclidean distance between points in $\tilde{M}$ and the mapping distance is [109]:

$$\delta(\Gamma^G, \Gamma^A) = \min \gamma(\tilde{M}) \quad (5.2)$$

Dynamic programming was used to find the minimum cost. A trace $\tilde{T}$ contains a set of ordered pairs, the mappings from $\Gamma^G$ to $\Gamma^A$; see Figure 5.6 for an example of a mapping sequence and trace. The contour mapping measure $CMM$, is the normalized distance between $\Gamma^G$ and $\Gamma^A$:

$$CMM(\Gamma^G, \Gamma^A) = \frac{1}{|\tilde{T}|} \delta(\Gamma^G, \Gamma^A) \quad (5.3)$$

where $|\tilde{T}|$ is the length of $\tilde{T}$. It is assumed that the first and last point on $\Gamma^G$ and $\Gamma^A$ were matched. If the two points do not match, then they minimize the cost around the cycle to calculate $CMM$ [109]:

$$CMM(\Gamma^G, \Gamma^A) = \frac{1}{|\tilde{T}|} \delta([\Gamma^G], [\Gamma^A]) \quad (5.4)$$

5.5 Ground Truth

Two types of ground truth were generated manually to calculate SPC and the contour mapping measure: ground truth bounding boxes and ground truth
5. Contour Refinement

Figure 5.7: Ground truth for same frame of 'Board' sequence: (a) Ground truth bounding boxes - one bounding box is added (blue rectangle) for the second only points (top left red and bottom right blue circle) are shown (the gray lines shows bound of each points); (b) Ground truth contours.

The ground truth bounding boxes were generated for each frame using the left image. For each bounding box two points (top left and bottom right) were selected and stored as ground truth bounding box. Figure 5.7(a) shows a selected frame on which one bounding box is added and for the second bounding box only selected points are shown. Ground truth bounding boxes were generated for the whole sequence.

Ground truth contours were generated by marking the required edges on the left image (right can also be used). The contours $\Gamma^G$ were then extracted and the contour mapping measure calculated (Equation 5.3). Figure 5.7(b) shows an example of generated ground truth contours.
Figure 5.8: One frame selected from each sequence: (a) ‘Walk’; (b) ‘Board’; (c) ‘S-Ball’. The top row shows left frame, second row shows false colour disparity map (purple represents closer and blue represents farther away), third row shows ground truth bounding boxes on left frame and and the bottom row shows ground truth contours on left frame.
5.6 Experimental Results

Three sequences - 'Walk', 'Board' and 'S-Ball' - captured by the SDPS FPGA hardware (see Section 2.10) were used. Ground truth bounding boxes (discussed in Section 5.5) were generated for all frames of each sequence and ground truth contours (discussed in Section 5.5) were generated for selected frames of each sequence. Figure 5.8 shows left frame, false colour disparity map, ground truth bounding boxes and ground truth contours (marked on left frame) for each sequence.

The 'Walk' sequence has 150 frames with one subject walking, while the 'Board' sequence has 28 frames with two subjects, ST and SB (marked in the left frame Figure 5.8). Both subjects were static throughout the sequence. SB holds a board which makes it easy to generate ground truth bounding boxes due to its regular shape but the textureless board generates more streaks and tests the streak removal procedures! The 'S-Ball' sequence contains 52 frames selected from the 'Ball' sequence in which a ball was thrown by the subject.
Figure 5.10: Average percent of SPC over the whole ‘Board’ sequence: (a) subject ST; (b) subject SB.
5. Contour Refinement

5.6.1 Streak Points Count

The average percent SPC (discussed in Section 5.4) was calculated for contours over the whole sequence after processing them by mean, median, max, min, Mn-Mx (Algorithm 5) and Triple Edge Mn-Mx (Algorithm 6) for contour segment length with 3 - 25 points \((2 \times w_r + 1)\) for \(w_r\) from 1 to 12 (discussed in Section 5.3). Figure 5.9 shows the average percent of SPC for the ‘Walk’ sequence and Figure 5.10 shows SPC for both subjects ST and SB in the ‘Board’ sequence. The ‘Board’ sequence was static, so the standard deviation was also calculated: 3% for mean and 4% for all other operators. Figure
5.6. Experimental Results

<table>
<thead>
<tr>
<th>Segment length</th>
<th>Mean</th>
<th>Median</th>
<th>Max</th>
<th>Min</th>
<th>Mn-Mx</th>
<th>Triple Edge Mn-Mx</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>12.1</td>
<td>11.3</td>
<td>13.1</td>
<td>12.6</td>
<td>9.7</td>
<td>9.7</td>
</tr>
<tr>
<td>5</td>
<td>12.0</td>
<td>11.1</td>
<td>14.3</td>
<td>13.3</td>
<td>8.6</td>
<td>8.5</td>
</tr>
<tr>
<td>7</td>
<td>11.9</td>
<td>11.0</td>
<td>15.3</td>
<td>13.9</td>
<td>7.9</td>
<td>7.9</td>
</tr>
<tr>
<td>9</td>
<td>11.8</td>
<td>10.8</td>
<td>16.1</td>
<td>14.4</td>
<td>7.5</td>
<td>7.5</td>
</tr>
<tr>
<td>11</td>
<td>11.8</td>
<td>10.6</td>
<td>16.9</td>
<td>14.9</td>
<td>7.2</td>
<td>7.2</td>
</tr>
<tr>
<td>13</td>
<td>11.7</td>
<td>10.6</td>
<td>17.5</td>
<td>15.3</td>
<td>7.0</td>
<td>7.0</td>
</tr>
<tr>
<td>15</td>
<td>11.7</td>
<td>10.5</td>
<td>18.2</td>
<td>15.6</td>
<td>6.8</td>
<td>6.8</td>
</tr>
<tr>
<td>17</td>
<td>11.7</td>
<td>10.4</td>
<td>18.8</td>
<td>15.8</td>
<td>6.7</td>
<td>6.7</td>
</tr>
<tr>
<td>19</td>
<td>11.6</td>
<td>10.4</td>
<td>19.3</td>
<td>16.1</td>
<td>6.6</td>
<td>6.6</td>
</tr>
<tr>
<td>21</td>
<td>11.6</td>
<td>10.4</td>
<td>19.9</td>
<td>16.4</td>
<td>6.6</td>
<td>6.6</td>
</tr>
<tr>
<td>23</td>
<td>11.6</td>
<td>10.4</td>
<td>20.4</td>
<td>16.6</td>
<td>6.5</td>
<td>6.5</td>
</tr>
<tr>
<td>25</td>
<td>11.5</td>
<td>10.4</td>
<td>20.8</td>
<td>16.9</td>
<td>6.5</td>
<td>6.4</td>
</tr>
</tbody>
</table>

Table 5.1: Contour mapping measure for frame # 57 from the ‘Walk’ sequence.

5.11 shows the average percent SPC for the ‘S-Ball’ sequence. The results show that both Mn-Mx and Triple Edge Mn-Mx algorithms consistently reduced streaks if segment length was increased and were better than all other operators.

5.6.2 Contour Mapping Measure

The contour mapping measure (CMM) was calculated for selected frames from each sequence for contour segment length with 3 - 25 points \((2 \times w_r + 1)\) for \(w_r\) from 1 to 12. Smaller CMM means that contours match well and if two contours match exactly then their CMM = 1. Table 5.1 shows CMM for frame # 57 from the ‘Walk’ sequence; Table 5.2 shows CMM for frame # 12 from the ‘Board’ sequence for both subjects (ST and SB) and Table 5.3 shows CMM
Table 5.2: CMM for frame #12 from the ‘Board’ sequence.
5.6. Experimental Results

<table>
<thead>
<tr>
<th>Segment length</th>
<th>Mean</th>
<th>Median</th>
<th>Max</th>
<th>Min</th>
<th>Mn-Mx</th>
<th>Triple Edge Mn-Mx</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4.8</td>
<td>5.0</td>
<td>5.8</td>
<td>4.9</td>
<td>4.7</td>
<td>4.9</td>
</tr>
<tr>
<td>5</td>
<td>4.7</td>
<td>4.9</td>
<td>6.4</td>
<td>5.0</td>
<td>4.5</td>
<td>4.9</td>
</tr>
<tr>
<td>7</td>
<td>4.8</td>
<td>4.9</td>
<td>6.9</td>
<td>5.2</td>
<td>4.5</td>
<td>5.1</td>
</tr>
<tr>
<td>9</td>
<td>4.9</td>
<td>5.0</td>
<td>7.3</td>
<td>5.5</td>
<td>4.5</td>
<td>4.8</td>
</tr>
<tr>
<td>11</td>
<td>5.0</td>
<td>5.1</td>
<td>7.8</td>
<td>5.8</td>
<td>4.5</td>
<td>4.9</td>
</tr>
<tr>
<td>13</td>
<td>5.1</td>
<td>5.2</td>
<td>8.2</td>
<td>6.2</td>
<td>4.6</td>
<td>5.0</td>
</tr>
<tr>
<td>15</td>
<td>5.3</td>
<td>5.3</td>
<td>8.6</td>
<td>6.6</td>
<td>4.7</td>
<td>4.9</td>
</tr>
<tr>
<td>17</td>
<td>5.4</td>
<td>5.5</td>
<td>9.0</td>
<td>7.0</td>
<td>4.8</td>
<td>4.9</td>
</tr>
<tr>
<td>19</td>
<td>5.7</td>
<td>5.6</td>
<td>9.6</td>
<td>7.4</td>
<td>5.0</td>
<td>4.9</td>
</tr>
<tr>
<td>21</td>
<td>5.9</td>
<td>5.8</td>
<td>10.0</td>
<td>7.8</td>
<td>5.1</td>
<td>5.0</td>
</tr>
<tr>
<td>23</td>
<td>6.1</td>
<td>6.0</td>
<td>10.4</td>
<td>8.2</td>
<td>5.3</td>
<td>5.1</td>
</tr>
<tr>
<td>25</td>
<td>6.3</td>
<td>6.1</td>
<td>10.7</td>
<td>8.6</td>
<td>5.5</td>
<td>5.2</td>
</tr>
</tbody>
</table>

Table 5.3: Contour mapping measure for frame # 45 from the ‘S-Ball’ sequence.

for frame # 45 from the ‘S-Ball’ sequence. Compared to all other operators the contour, processed by Mn-Mx and Triple Edge Mn-Mx algorithms have lower CMM.

5.6.3 Analysis

The average percent of SPC (Figure 5.9, Figure 5.10 and Figure 5.11) show that Mn-Mx (Algorithm 5) performs better than Triple Edge Mn-Mx (Algorithm 6) for all sequences. The CMM results are also similar for the ‘Board’ sequence (Table 5.2). However, although for the ‘Walk’ and ‘S-Ball’ sequences (Table 5.1 and Table 5.3) there are more streaks, the CMM results of Triple Edge Mn-Mx are better than those obtained with the Mn-Mx algorithm, specially for segment length > 17.

If we increase the segment length then we can search further for either Triple
Edge or Mn-Mx point and the streaks (percent of SPC) are reduced (Figure 5.9, Figure 5.10 and Figure 5.11). However, CMM can increase as shown by subject ST from the ‘Board’ sequence (Table 5.2) and ‘S-Ball’ sequence (Table 5.3).

The complexity of the Mn-Mx algorithm is $\mathcal{O}(mw)$ and the complexity of Triple Edge Mn-Mx is $\mathcal{O}((m - te)w)$ where $m$ is the number of contour points, $w$ is segment length and $te$ is Triple Edge points count. If large numbers of contour points are Triple Edge points, then the constant factor (omitted from big-Oh) is smaller and the algorithm Triple Edge Mn-Mx becomes significantly faster. Figure 5.12 shows the number of points which satisfied each condition in the Triple Edge Mn-Mx algorithm for the ‘Board’ sequence. More than 20% of the points are Triple Edge points.

### 5.7 Summary

The conventional way to improve object contours derived from disparity maps is to pre-process the disparity map before contour detection is carried out. The pre-processing step can improve the contours but can also remove the fine details in the disparity map. Here, the contours are extracted from ‘raw’ SDPS generated disparity map (without affecting possible fine details). The extracted contours are then improved by applying the two proposed algorithms Mn-Mx and Triple Edge Mn-Mx. The Mn-Mx intelligently uses minimum and maximum operators for ML and MR regions to improve contours, while Triple Edge Mn-Mx also uses Triple Edge points.

Two evaluation metrics were used: streak points count (SPC) and contour mapping measure (CMM). SPC is introduced in this chapter as it is easy to build ground truths for it and can quantify streaks. Three sequences were used for experiments which were captured by SDPS FPGA hardware. Experiments showed that the proposed algorithms (Mn-Mx and Triple Edge
Figure 5.12: Fraction of points satisfying each condition in Algorithm 6 at different segment length (3-25). Average points over the whole ‘Board’ sequence after background removal. C1, C2, C3 and C4 represent conditions in Algorithm 6 indicated in Figure 5.4.

\( \text{Mn-Mx} \) can refine the contours extracted from the ‘raw’ SDPS generated disparity maps.
Chapter 6  

Conclusion and Future Work

The aim of this research was to design algorithms that could improve the usability of real-time Symmetric Dynamic Programming Stereo (SDPS) hardware. The main focus was on algorithms for fast depth contour extraction, robust identification of Points-of-Interest in the presence of depth noise and depth contour refinement algorithm. The key challenges were to process the large amount of data generated by SDPS hardware and to handle the noise in the generated depth maps efficiently and quickly so as to minimize possible bottlenecks for high level vision tasks.

The key contributions are a novel algorithm for depth contour generation, Points-of-Interest (Triple Edge) detection algorithm and two contour refinement algorithms. To reduce computational complexity both contour generation and Points-of-Interest detection algorithms reduce the search space in one pass and then use the reduced search space for subsequent processing. The Points-of-Interest detection technique is robust to depth noise because it uses the left and right images. The contour refinement algorithms were developed to refine the extracted depth contour without processing the whole depth map which preserves the fine details.

For depth contour extraction, the main problem with the border following technique is the processing time and its inability to handle known noise in the depth maps. It operates on binary images in order to generate contours. The novel contour generation technique, Salmon, proposed and discussed in Chapter 3 is able to generate a number of critical contour maps in real-time.
Experimental results show that this algorithm is as accurate as the border following algorithm and more than six times faster. The Salmon algorithm overcomes common SDPS depth noise (single line streaks) while generating contour. The experimental results show that the generated contours by Salmon have 30% fewer contour points than the contours generated by border following algorithm.

Conventional 3-D Points-of-Interest detection techniques use only the generated depth map. These techniques have difficulty with depth maps generated by a real-time stereo vision system due to the amount of noise. A novel, fast and reliable Points-of-Interest (Triple Edge) detection technique was proposed and discussed in Chapter 4. It uses all available information - left image, right image, disparity map and occlusion map (if available). Some tracking sequences were used to show how the Triple Edge points enable us to rapidly locate objects in a scene for subsequent processing exercises. Marking Triple Edge points takes time of the order of 13 ms for 1024 × 768 high resolution images. Triple Edge points were used here to overcome noise and artefact problems in depth maps generated in real-time in applications such as object detection, pose and recognition and tracking.

Depth contours generated from a ‘raw’ disparity map are often noisy which decreases their practical use. For example, the contours generated from an SDPS disparity map contain known streaks. The conventional way is to refine the disparity map before generating contours using either median filter [132], morphological open operators [16], disparity voting [93] or left to right consistency check [25, 35]. However, these techniques can remove the fine details especially in high resolution stereo disparity maps where depth accuracy is high (Δ ≥ 128).

Two algorithms Mn-Mx and Triple Edge Mn-Mx are proposed and discussed in Chapter 5. They reduce the streaks in an object contour by processing the contour points and by using rules derived form the SDPS depth
map. Streak point count (SPC) and contour mapping measure (CMM) were used as evaluation metrics. The experimental results show that both $Mn-Mx$ and *Triple Edge Mn-Mx* reduced streaks with better contour mapping measure than when processing a contour by mean, median, minimum and maximum operators.

### 6.1 Directions for Future Research

There are two hardware implementation of SDPS - one on FPGA and other using GPU (discussed in Chapter 2). The current FPGA implementation of SDPS is very efficient and uses only a fraction of the resources available in modern FPGAs [67] so that additional processing on the FPGA surface is possible. The preprocessing step in contour generation which need two other scan lines beside current scan line can be transferred to FPGA. This freeing further CPU cycles for high level vision tasks. Additional FPGA processing will add a small latency (typically less than 1 ms) to the time between image capture at the camera and receipt of scan lines in the host processor.

The contour generation algorithm is tested on multiple cores processor (Quad core). The time to generate each contour is proportional to its length but the algorithm generates each contour independently and is able to take advantage of the GPU commonly available now. The GPU implementation will further improve the performance and will provide extra time for high level vision tasks.

The generated contours can be used to compress the depth map by saving only the contours information [64]. The precise object tracking technique discussed by Butt and Morris [16] and the gesture recognition technique discussed by Maldeni *et al.* [96] which are using depth contours can be improved by the contour generated algorithm discussed here.
The **Triple Edge** point detection (discussed in Chapter 4) requires simple calculations and uses only information that is local to a scan-line and can therefore easily be transferred to FPGA or GPU hardware adding only a few cycles - microseconds or less - to the latency. The **Triple Edge** point detection and contour generation algorithms can be merged together which will reduce further processing time.

Multiple pass algorithms are now available to improve disparity maps [83]. Currently SDPS hardware uses fixed occlusion cost which is one of the main factors for streaks in the resulting disparity maps [68]. **Triple Edge** points can be used to dynamically change occlusion cost in the second pass which will most likely improve the depth maps.

The **Mn-Mx** and **Triple Edge Mn-Mx** algorithms (discussed in Chapter 5) use only local information for contour processing global smoothness constraints can be used to generate smoother contours. GPU implementation of **Mn-Mx** and **Triple Edge Mn-Mx** will improve performance and will provide extra time for matching contours to object profiles for object modelling, detection, recognition and tracking.
Appendix A

Appendix A: Stereo Configuration

A.1 Cameras used with SDPS Hardware

SDPS FPGA and GPU hardware were used for experiments [107, 108, 72] with three pairs of high resolution cameras:

- Sentech CL83A [137] cameras were used with FPGA hardware
- Basler acA1000-30gc [6] cameras were used with GPU hardware
- Basler acA2000-50gc [7] cameras were used with GPU hardware.

Table A.1 shows the basic specification of each camera type. All of the mentioned cameras (Table A.1) support C-mount lenses; a pair of either 6mm, 9mm, 16mm or 25mm lenses were used.

<table>
<thead>
<tr>
<th>Features</th>
<th>Sentech CL83A</th>
<th>Basler acA1000-30gc</th>
<th>Basler acA2000-50gc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>Cameralink</td>
<td>Gigabit Ethernet</td>
<td>Gigabit Ethernet</td>
</tr>
<tr>
<td>Resolution</td>
<td>1024 × 768</td>
<td>1032 × 778</td>
<td>2046 × 1086</td>
</tr>
<tr>
<td>Pixel size</td>
<td>4.65µm × 4.65µm</td>
<td>4.65µm × 4.65µm</td>
<td>5.5µm × 5.5µm</td>
</tr>
<tr>
<td>Frame rate</td>
<td>30 fps</td>
<td>31 fps</td>
<td>50 fps</td>
</tr>
<tr>
<td>Pixel depth</td>
<td>10 bits</td>
<td>12 bits</td>
<td>12 bits</td>
</tr>
</tbody>
</table>

Table A.1: Basic specification of cameras.
A. Appendix A: Stereo Configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>value/s</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image width ((w))</td>
<td>1024</td>
<td>pixels</td>
</tr>
<tr>
<td>Image height ((h))</td>
<td>768</td>
<td>pixels</td>
</tr>
<tr>
<td>Baseline ((b))</td>
<td>50 – 1000</td>
<td>mm</td>
</tr>
<tr>
<td>Maximum disparity ((d_{\text{max}}))</td>
<td>128</td>
<td>pixels</td>
</tr>
<tr>
<td>Minimum disparity ((d_{\text{min}}))</td>
<td>40</td>
<td>pixels</td>
</tr>
<tr>
<td>Pixel size ((\rho))</td>
<td>4.56</td>
<td>(\mu m)</td>
</tr>
<tr>
<td></td>
<td>5.5</td>
<td>(\mu m)</td>
</tr>
<tr>
<td>Focal length ((f))</td>
<td>6</td>
<td>(mm)</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>(mm)</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>(mm)</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>(mm)</td>
</tr>
</tbody>
</table>

Table A.2: Parameters for stereo configuration.

A.2 Stereo Configuration used for Experiments

The design of an optimum stereo configuration is important for experiments because different cameras (presented in Table A.1) and lenses (e.g. 6mm, 9mm, 16mm or 25mm) can be used. In order to calculate the depth of an object using stereo the object must be in a common field of view of both cameras. The distance between cameras (baseline) and the angle between cameras can also affect the stereo accuracy [66].

For quick experimental set-up the extent of common field of view (extent), depth \(Z\) and depth resolution \(dz\) were calculated for the parameters given in Table A.2 for canonical stereo configuration. Figure A.1 shows the calculated values for 6mm, Figure A.2 shows the values for 9mm, Figure A.3 shows for 16mm and Figure A.4 shows the calculated value for 25mm lenses for cameras Sentech CL83A and Basler acA1000-30gc (pixel size \(\rho\) 4.65\(\mu m\)) for
both $d_{\text{max}}$ and $d_{\text{min}}$. For Basler acA2000-50gc (pixel size $\rho = 5.5\mu m$) Figure A.5 shows calculated values for 6mm, Figure A.6 for 9mm, Figure A.7 for 16mm and Figure A.8 for 25mm lenses. In each figure the extent and depth $Z$ are on the primary axis (in meters) and $dz$ is given on the secondary axis (in millimetres).
Figure A.1: The calculated extent, $Z$, and $dz$ for $f = 6\text{mm}$ and $\rho = 4.65\mu\text{m}$ (a) for $d_{\text{max}}$ (b) for $d_{\text{min}}$. 
A.2. Stereo Configuration used for Experiments

Figure A.2: The calculated extent, $Z$, and $dz$ for $f = 9\text{mm}$ and $\rho = 4.65\mu\text{m}$ (a) for $d_{\text{max}}$ (b) for $d_{\text{min}}$. 
Figure A.3: The calculated extent, $Z$, and $dz$ for $f = 16\, \text{mm}$ and $\rho = 4.65\, \mu\text{m}$ (a) for $d_{\text{max}}$ (b) for $d_{\text{min}}$. 
A.2. Stereo Configuration used for Experiments

Figure A.4: The calculated extent, $Z$, and $dz$ for $f = 25mm$ and $\rho = 4.65\mu m$ (a) for $d_{max}$ (b) for $d_{min}$. 

(a) 

(b)
Figure A.5: The calculated extent, $Z$, and $dz$ for $f = 6\, \text{mm}$ and $\rho = 5.5\, \mu\text{m}$ (a) for $d_{\text{max}}$ (b) for $d_{\text{min}}$. 
A.2. Stereo Configuration used for Experiments

Figure A.6: The calculated extent, $Z$, and $dz$ for $f = 9\, \text{mm}$ and $\rho = 5.5\, \mu\text{m}$
(a) for $d_{\text{max}}$; (b) for $d_{\text{min}}$. 
Figure A.7: The calculated extent, $Z$, and $dz$ for $f = 16mm$ and $\rho = 5.5\mu m$ (a) for $d_{max}$ (b) for $d_{min}$. 
A.2. Stereo Configuration used for Experiments

Figure A.8: The calculated extent, \( Z \), and \( dz \) for \( f = 25 \text{mm} \) and \( \rho = 5.5 \mu\text{m} \) (a) for \( d_{\text{max}} \) (b) for \( d_{\text{min}} \).
List of Figures

1.1 An example to indicate a possible high probability that a monocular system will interpret the scene differently from each image. ......................................................... 4

1.2 Face detection by monocular and stereo vision based system. The aim is to identify actual faces. ................................. 6

1.3 Thesis structure. .............................................................. 11

2.1 Pinhole camera model. .................................................... 14

2.2 Projection using pinhole camera model in which a virtual image plane has been placed in front of the optical centre. This simplifies the mathematics by removing the inversion of the image on the real image plane. ......................................................... 15

2.3 Sample image. The origin is at the top left corner and the intensity values are given for the selected region. .................. 17

2.4 Stereo camera setup: the textured region shows the common field of view. ......................................................... 18

2.5 Projections of an object with two pinhole cameras. ........... 19
2.6 Depth $Z$ can be calculated using similar triangles $QO_lO_r$ and $Qx_lx_r$. ......................................................... 20

2.7 Depth ($Z$) is inversely proportional to disparity ($d$) - two corresponding points at different depths (one near and one far) are marked on left (top) and right (bottom) images which shows that $Z$ is inversely proportional to $d$. .......................... 21

2.8 Epipolar geometry - showing epipolar plane, epipolar line and epipolar points for point, $Q$. ........................................... 22

2.9 The epipolar constraint: points in one image are projected onto the epipolar line in the other image. ................................. 23

2.10 Contour map of distortions from our laboratory’s CameraLink (STC-CL83A) camera. The central region has essentially zero distortion. ................................................................. 24

2.11 Camera calibration procedure flowchart. ................................. 26

2.12 Stereo image pair with ground truth disparity map [131]. .... 29

2.13 Left image of the scene and its reconstruction using SDPS from side view. Reconstruction does not consider sub-pixel disparities. ................................................................. 32

2.14 Camera configuration, showing a part of Cyclopæan image seen by a virtual Cyclopæan camera (centre) and a scene object profile. Visibility states (ML, B or MR) are marked on profile from Morris et al.[108]. ................................................................. 35

2.15 Left images for real-time stereo matching [131, 71]. ............ 42

2.16 RMS error for SDPS, CM, SGM, BP and BP-CS (from Kalarot et al. [73]). ................................................................. 43

2.17 Percent bad pixels for SDPS, CM, SGM, BP and BP-CS. ...... 44

2.18 FPGA implementation block diagram (from Morris et al. [108]).
2.19 Distortion removal and rectification lookup table (LUT) generation. ......................................................... 46

2.20 Three frames from a video sequence. Note that, in the Cyclopæan view ‘images’ have twice as many pixels per scan line. In the false colour depth map purple represents closer and blue represents farther away. In the occlusion map white represent binocular points. ......................................................... 48

2.21 GPU implementation of SDPS, forward pass. Inputs are rectified scan lines and the outputs are predecessor array and final cost array for back tracking phase (taken from [74]). ............... 50

3.1 Intensity contours vs depth contours for the Middleburry data set Baby [131]: (a) Left image; (b) Depth map (ground truth) where lighter colour represent near to camera; (c) Detected contours of left image; (d) Detected depth contours. ................. 54

3.2 Order in which the neighbours are visited for each of the three Salmon states for a Salmon ‘descending’ ($\pi \leq \phi \leq 2\pi$) an ML edge. The order is reversed for a Salmon ‘climbing’ ($0 \leq \phi \leq \pi$) an MR edge. ................................................................. 60

3.3 Example Salmon ‘run’: boxes represent pixels in the disparity and occlusion maps; they are labeled with the disparity and the visibility state after C states have been assigned. The background pattern for each pixel shows the Salmon state as it visits a pixel - see the legend. The Salmon starts with the highlighted (7, ML) pixel at the top, ‘descends’ through ML pixels and climbs back (not shown) through MR pixels to reach its starting point again. ................................................................. 62

3.4 Disparity map for scene object with horizontal edges and locations where multiple contours share pixels: the marked regions (a-h) have been expanded in Figure 3.5 to show details. ............ 63
3.5 Salmon traversal for expanded regions (a-h) marked in Figure 3.4. ......................................................... 64

3.6 Selected frames from each sequence: rectified left images and false colour disparity maps (purple represents closer (larger \(d\)) and blue represents farther away (smaller \(d\)). (a) Ball frame # 122; (b) Gujral frame # 97; (c) Bob frame # 43; (d) Dolls. .................. 67

3.7 Salmon contours for frame # 43 from Bob sequence. ............... 68

3.8 Percent reduction in contour points using Salmon for each frame of the Gujral and Bob sequences. The mean reduction is 30% for Gujral and 33% for Bob. ................................. 73

4.1 This frame (from CITR2 Sequence discussed in Section 4.5) contains two subjects with bounding boxes. A false colour disparity map, a disparity state map and Triple Edge points are shown only for regions inside the bounding boxes for ground truth subject edges. ..................................................... 78

4.2 Triple Edge (TE) points detection. ........................................ 80

4.3 Lens blur estimation using calibration image: (a) centre of square detection; (b) blur estimation using detected centres. .................. 82

4.4 Number of motion blurred pixels for \(\varphi_o = 0^\circ\) ...................... 85

4.5 Number of motion blurred pixels for \(\varphi_o = 45^\circ\) ...................... 86

4.6 Number of motion blurred pixels for \(\varphi_o = 90^\circ\) ...................... 87

4.7 Find shift between corresponding edges: selected normalized scan line profiles: \(I^L\), \(I^R\), \(D\), \(I^{Le}\) and \(I^{Re}\) (frame # 60 CITR2 sequence). The peak on each edge as shown by black dot- see (a) and (b). Corresponding peaks were compared to find shift. ................................. 89

4.8 Flowchart of Triple Edge points detection algorithm. ............. 90
4.9 Object based ground truth generation. For cropped subject: (1) graph cut disparity map; (2-3) manually marked object border; (4) subject disparity enclosed by extracted border; (5) subject ground truth disparity. The shadow in (1) and (4) shows the occluded region.

4.10 Middlebury ground truth images with selected objects [131].

4.11 CITR2 sequence (400 frames) contains two walking persons, ST and SK, marked in (c). (a) **Triple Edge** points depth distribution for each frame of the sequence; (b) **Triple Edge** points occupancy over the sequence and (c)-(e) Selected frames from the sequence.

4.12 Normalized disparity distributions for GT and **Triple Edge** points (TE). The legend in (a) applies to all graphs (b-e).

4.13 Selected frame #174 from the CITR1 sequence: (a) left image; (b) disparity state map \(D^S\); (c) **Triple Edge** points; (d) significant **Triple Edge** points.

4.14 Selected frame #40 from the CITR2 sequence: (a) left image; (b) disparity state map \(D^S\); (c) **Triple Edge** points; (d) significant **Triple Edge** points.

4.15 Selected frames from the CITR3 sequence: (a) left image; (b) disparity state map \(D^S\); (c) **Triple Edge** points; (d) significant **Triple Edge** points.

4.16 Selected frame #15 from the KITTI sequence: (a) left image; (b) disparity state map \(D^S\); (c) **Triple Edge** points; (d) significant **Triple Edge** points.
5.1 Contours generated from the Middleburry Dolls image: (a) original left image; (b) false colour disparity map; (c) ‘raw’ contours (for region marked by white rectangle in b); (d-f) contours after median filtering. Note that fine details are clear in the raw contours whereas median filtering distorts the contours. Short length contours are shown in black (examine the eyes and mouth contours). .......................... 104

5.2 Ground truth contour (red) with raw and refined contour (blue). from sequence CITR1 frame # 260. .......................... 106

5.3 Mn-Mx filtered contour (blue) with ground truth contour (red). 109

5.4 Triple Edge Mn-Mx Algorithm flowchart. For further detail see Algorithm 6. Conditions C1, C2, C3 and C4 refer to the graphs shown in Figure 5.5 and Figure 5.12. .......................... 111

5.5 Contour refinement with Triple Edge Mn-Mx (Algorithm 6). The $x$ values for points; that will be chosen if the conditions C1, C2, C3 and C4 with the same label in the flowchart (Figure 5.4) are met. For C4 the min value is chosen because $b_k$ is assumed in an MR region : in an ML region, the max value would be chosen. .......................... 112

5.6 Mapping between two contours: $\Gamma^A = a_1, a_2, a_3, a_4, a_5$ and $\Gamma^G = g_1, g_2, g_3$ and their correspondence map $\tilde{M} = a_1 \leftrightarrow g_1, a_2 \leftrightarrow g_2, a_3 \leftrightarrow g_2, a_4 \leftrightarrow g_3, a_5 \leftrightarrow g_3$. The trace is $\tilde{T} = z(1,1), (2,2), (3,2), (4,3), (5,3)$ and the distances are $\gamma(\tilde{M}) = \text{dist}(a_1, g_1) + \text{dist}(a_2, g_2) + \text{dist}(a_3, g_2) + \text{dist}(a_4, g_3) + \text{dist}(a_5, g_3)$. 114
5.7  Ground truth for same frame of ‘Board’ sequence: (a) Ground
truth bounding boxes - one bounding box is added (blue rect-
angle) for the second only points (top left red and bottom right
blue circle) are shown (the gray lines shows bound of each
points); (b) Ground truth contours. .......................... 116

5.8  One frame selected from each sequence: (a) ’Walk’; (b) ’Board’;
(c) ’S-Ball’. The top row shows left frame, second row shows
false colour disparity map (purple represents closer and blue
represents farther away), third row shows ground truth bound-
ing boxes on left frame and and the bottom row shows ground
truth contours on left frame. ................................. 117

5.9  Average percent of SPC over the whole ’Walk’ sequence. . . . 118

5.10 Average percent of SPC over the whole ’Board’ sequence: (a)
subject ST; (b) subject SB. ................................. 119

5.11 Average percent of SPC over the whole ’S-Ball’ sequence. . . . 120

5.12 Fraction of points satisfying each condition in Algorithm 6
at different segment length (3-25). Average points over the
whole ‘Board’ sequence after background removal. C1, C2,
C3 and C4 represent conditions in Algorithm 6 indicated in
Figure 5.4. .......................................................... 125

A.1  The calculated extent, $Z$, and $dz$ for $f = 6\text{mm}$ and $\rho = 4.65\mu\text{m}$
(a) for $d_{\text{max}}$ (b) for $d_{\text{min}}$. .............................. 134

A.2  The calculated extent, $Z$, and $dz$ for $f = 9\text{mm}$ and $\rho = 4.65\mu\text{m}$
(a) for $d_{\text{max}}$ (b) for $d_{\text{min}}$. .............................. 135

A.3  The calculated extent, $Z$, and $dz$ for $f = 16\text{mm}$ and $\rho = 4.65\mu\text{m}$
(a) for $d_{\text{max}}$ (b) for $d_{\text{min}}$. .............................. 136

A.4  The calculated extent, $Z$, and $dz$ for $f = 25\text{mm}$ and $\rho = 4.65\mu\text{m}$
(a) for $d_{\text{max}}$ (b) for $d_{\text{min}}$. .............................. 137
A.5 The calculated extent, $Z$, and $dz$ for $f = 6\text{mm}$ and $\rho = 5.5\mu\text{m}$
   (a) for $d_{\text{max}}$ (b) for $d_{\text{min}}$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 138
A.6 The calculated extent, $Z$, and $dz$ for $f = 9\text{mm}$ and $\rho = 5.5\mu\text{m}$
   (a) for $d_{\text{max}}$; (b) for $d_{\text{min}}$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 139
A.7 The calculated extent, $Z$, and $dz$ for $f = 16\text{mm}$ and $\rho = 5.5\mu\text{m}$
   (a) for $d_{\text{max}}$ (b) for $d_{\text{min}}$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 140
A.8 The calculated extent, $Z$, and $dz$ for $f = 25\text{mm}$ and $\rho = 5.5\mu\text{m}$
   (a) for $d_{\text{max}}$ (b) for $d_{\text{min}}$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 141
List of Tables

2.1 KITTI stereo matching algorithms ranking [40]. Only matching accuracy is used for ranking. Time is given in seconds s for 1242 $\times$ 375 images. High rank algorithms are too slow (300s and 8s top two). ................................................. 39

2.2 Real-time stereo matching algorithms. ................................................. 40

2.3 Frames per second (fps) vs disparity range for 512$\times$512 pixel images [73]. The top row (in bold) is disparity range. ........... 41

3.1 Salmon state definitions. .................................................. 59

3.2 Experimental data overview: $\Delta_c$ is the range of disparities for critical contours which are used to outline and model the objects of interest in the scene. ................................................. 65

3.3 Number of points in each state. .................................................. 66

3.4 Salmon contour validation for all frames in each sequence using ground truth $D^{pa}$. True positive and false positive points were calculated using Algorithm 3 no false negative points were detected using Algorithm 4. ................................................. 70
3.5 Contour validation using ground truth $D^\text{gm}$ for each frame of the \textbf{Bob} and \textbf{Gujral} sequences. No false negative points were detected using Algorithm 4. ........................................ 71
3.6 Salmon execution time: all times are given in milliseconds on a 2.4 GHz quad core processor with 4 GB of memory. One core was used for task assignment and synchronization. ........... 74
3.7 Comparison of the Salmon algorithm with border following algorithm. All times in milliseconds. ......................... 75
4.1 Estimated lens blur for checkerboard images. ................. 83
4.2 Parameters for motion blur estimation. ......................... 84
4.3 Calculating $x_L$ and $x_R$ from $x$. ............................. 91
4.4 Sequence statistics. ............................................. 94
4.5 \textbf{Triple Edge} points for each sequence. ................. 102
5.1 Contour mapping measure for frame # 57 from the ‘Walk’ sequence. .......................................................... 121
5.2 CMM for frame # 12 from the ‘Board’ sequence. ............ 122
5.3 Contour mapping measure for frame # 45 from the ‘S-Ball’ sequence. .......................................................... 123
A.1 Basic specification of cameras. ............................... 131
A.2 Parameters for stereo configuration. .......................... 132


ci
a, Spain, January 22-24, 2010*, pages 391–397. INSTIC


[100] Stephan Meister, Shahram Izadi, Pushmeet Kohli, Martin Hammerle, Carsten Rother, and Daniel Kondermann. When can we use kinectfusion for ground truth acquisition? In Workshop on Color-Depth Camera Fusion in Robotics, IROS, 2012, 2012.


