Indoor Positioning Using Stereo Cameras

By

Kun Zhuang

A thesis submitted to the Faculty of Graduate and Postdoctoral

Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Mechanical Engineering

Carleton University

Ottawa, Ontario

Abstract

In this thesis, a framework based on open-source hardware is proposed and built to study the practicability of its use in indoor robotics research. Two Raspberry Pi camera modules are used as a range sensor for indoor robot triangulation in a static scene, which infers 3D information from 2D space using stereo vision. The local Block Matching method and Semi-Global matching method are implemented in dense disparity map estimations. 3D reconstructions are performed and binary 2D maps are generated. Previous work has paid attention to iterative methods to minimize the global energy function in solving matching problems. This thesis presents two vectorized methods that are suitable in stereo vision triangulation and acceptable for use in robotic applications. Results show that the depth estimation of both methods can be accurate to centimeters in between half a meter to four meters of range. Real-time implementation of these approaches has not been investigated.

Acknowledgements

To start with, it is a great pleasure to express my sincerest thanks and gratitude to my supervisor Dr. Jie Liu for providing this research opportunity to explore robotic applications in our daily life. In addition, without his guidance, valuable suggestions, constant encouragement, this research work could not have been accomplished.

I also wish to thank my parents MD. Jiayan Zhou and MD. Yihuang Zhuang for supporting me financially, spiritually and morally throughout my undergraduate and graduate studies at Carleton University.

Finally, I am very grateful to Dr. Liu's research group members and students for offering ideas and advice for my research work.

To my extraordinary parents,

for their endless love, support and encouragement.

Table of Contents

List of Tables

List of Figures

Nomenclatures

AD: Absolute differences

AR: Augmented Reality

ASW: Adaptive support weights

BM: Block matching

BT: Birchfield-Tomasi

CAD: Computer aided design

CCD: Charge-coupled-device

CIFS: Common internet file system

CMOS: Complementary-metal-oxide-semiconductor

CNN: Convolutional neural network

CT: Census transform

DCT: Discrete cosine transform

DoF: Degree of freedom

FOV: Field of view

FPGA: Field programmable gate array

FR: Full resolution

FW: Fixed window

GC: Graph cut

GNSS: Global navigation satellite system

GPIO: General purpose input/output

GPS: Global positioning system

GUI: Graphical user interface

HCR: Home care robot

HR: half resolution

HVS: Human visual system

ICC: Instantaneous center of curvature

IMU: Inertial measurement unit

IR: Infrared

JPEG: Joint photographic experts group

KITTI: Karlsruhe Institute of Technology

LBP: Loopy belief propagation

LiPo: Lithium polymer

MAP: Maximum a posteriori

MEMS: Micro-electromechanical systems

MRF: Markov random field

MVS: Machine vision system

NCC: Normalized cross correlation

NP: non-deterministic polynomial-time

PNG: Portable network graphics

RDP: Remote desktop protocol

RF: Radio frequency

RGB: Red green and blue

RMSE: Root mean square error

RT: Rank transform

RTK: Real-time kinematic

SAD: Sum of absolute differences

SD: Square differences

SGM: Semi-global matching

SoC: System on a chip

SSD: Sum of squared differences

ToF: Time of flight

UAV: Unmanned aerial vehicles *B*

UGV: Unmanned ground vehicles

VNC: Virtual network computing

WTA: Winner-takes-all

B : Motor viscous friction constant

$C_0(x, y, d)$: Initial matching cost

$C(x, y, d)$: Aggregated matching cost

$C(p, D_p)$: Matching cost on a pixel p

$d(x, y)$: Disparity value at x, y

d_{disp} : Disparity value or disparity level

d : Baseline length

$D_p(f_p)$: Matching cost energy function

$E(d)$: Global energy function respect to disparity

f : Focal length

i : Current

$I_L, I_R, I_L(x, y), I_R(x, y)$: Intensity value in left/image at coordinate (x, y)

J : Moment of Inertial of the rotor

Kt : Motor torque constant

K_{emf} : Motor electromotive force constant

L_a : Motor inductance

L_A: Diameter of the aperture

L_w : Distance between two wheels

$L_r(p,d)$: Aggregated cost at point p on disparity d

P_1 : Small penalty in SGM

P_2 : Large penalty in SGM

R_a : Motor resistance

$S(p,d)$: Sum of cost L_r over all paths r

T : Torque of the motor shaft

$v_L(t), v_R(t)$: Left and right wheel speed

$V(t)$: Tangential velocity about ICC

V_{emf} : Back electromotive force voltage

V_{in} : Input voltage

V_{in} : input voltage in s domain

$V_{i,j}(f_i, f_j)$: Matching cost energy function with smoothness constraints

$\omega(x, y)$: Window function for aggregation

z: Image distance

Z: Object distance

$\dot{\theta}$: Angular velocity of the motor

$\ddot{\theta}$: Angular acceleration of the motor

$\Phi(x_i, Y)$: Joint compatibility function of a node x_i by given observation *Y* in MRF

$\Psi(x_i, x_j)$: Compatibility between neighboring pixels i, j in MRF

$\Omega(t)$: Angular velocity about ICC

Chapter 1. Introduction

In this chapter, the two objectives of this thesis as well as the motivations behind them, are discussed. Next, the contributions are highlighted and a summary of each chapter is provided.

1.1 Overview

Sensors play an essential role in our daily lives. As the sensor becomes smaller and affordable, there are an increasing number of emerging industries and innovative applications that are arising. Therefore, numerous Micro-electromechanical Systems (MEMS) based sensor theories and their applications are being thoroughly studied. An example of this is the increasing research in sensor applications for Unmanned Ground Vehicles (UGVs) and Unmanned Aerial Vehicles (UAVs).

In addition to the traditional MEMS, image sensor technologies are also evolving rapidly. Vision systems can be considered to be a class of sensors that imitate human eyes to infer the 3D physical world and these systems provide additional sensing information along with other sensors. In the mid-80s, probabilistic and stochastic methods were introduced in order to solve robot localization and mapping problems. The primary aim of these methods is to perform sensor fusion and reject or lower the weights of unreliable inputs or observations with the given control commands. For example, the robot's location and pose may be estimated. However, assigning meaningful messages, such as distance and bearing information from certain landmarks in images taken by a camera in robotic applications, is

still a non-trivial task although research involving the use of stereo image pairs with vast disparities or parallaxes in military, extra-planetary exploration, disaster surveys began in the 1980s.

The fundamental objective of this thesis is to find a method to extract accurate ranging information from still images using inexpensive cameras for indoor service robot applications. Additionally, the method used in stereo normal case may be further explored and applied to stereo panoramic cameras [1]. Nonetheless, the real-time processing speed and elapsed processing time are not given focus in this thesis work. The algorithm, with deterministic processing time and number of operations, is preferred.

On the other hand, open-source communities, such as Arduino™ and Raspberry Pi™, are becoming extremely popular. This provides an opportunity for the public to learn more about microcontrollers/single board computers and design fun projects easily. Interestingly, these technologies are not new to academia and certain industries, and the technologies can be found in products such as security systems, toys, household applications, *etc*. These embedded systems are using similar types of microcontroller or System On a Chip (SoC). The key innovations of these community products include modular design and easy-to-program features, which allow for fast prototyping. Therefore, the secondary objective of this thesis is to investigate the feasibility of using open-source hardware with other off-the-shelf components in order to develop an indoor mobile robot research platform such that the research work can be streamlined in an affordable and time-efficient manner.

1.2 Contributions

The contributions of this thesis work are summarized as follows:

1) To propose, design and build a ground mobile robotic framework by using open-source and off-the-shelf parts. The limitations and issues involved with using community-supported and open-source components on research tasks are also discussed. For maximum efficiency, virtualization is introduced on the onboard computer when working on multiple operating systems and collaborative research.

2) To test and evaluate four stereo correspondence matching algorithms on Middlebury stereo datasets and images that have been captured by Raspberry Pi camera modules in order to emulate the real world indoor environment in Matlab.

3) To propose the two vectorized dense stereo matching algorithms for indoor triangulation using inexpensive Raspberry Pi camera modules with moderate ambient lighting. For best processing time, window-based Block Matching is recommended. In terms of performance, the Semi-Global Matching method is advised. Unreliable points are marked and excluded, and 2D/3D depth maps from a still scene are created for robots and humans, correspondingly. The issues and challenges involved with implementation are identified and the implementation is conducted in Matlab.

1.3 Organization

This thesis is structured into six overall chapters. The first chapter provides an introduction and discusses the motivations for developing indoor domestic service robots. The key challenge arises from whether stereo cameras can provide reliable distance measurements for robotic mapping and localization. Chapter 2 introduces the conventional sensors that can be found on contemporary research mobile robot platforms and it also discusses the assessments of these sensors. In this chapter, the fundamental theorems of distance measurement by stereoscopic vision are also stated. In Chapter 3, an experimental mobile platform is proposed and built using off-the-shelf and open-source hardware. The advantages and disadvantages of such an approach are also discussed. Subsequently, Chapter 4 provides a detailed literature review on stereoscopic depth measurement from the early 80s to the modern era, with applications to animals and machines and a focus on machines. Chapter 5 evaluates four stereo correspondence matching methods for depth estimations, and these methods include the two sparse feature-matching methods, vectorized local Block Matching (BM), and vectorized dense Semi-Global Matching (SGM) method. Each method has its unique working pipeline. The sample stereo pairs from Middlebury Stereo datasets, the images that are captured by low-cost Raspberry Pi camera modules on the proposed mobile platform, are being tested and examined through these pipelines. There is an in-depth investigation of the BM and SGM methods in this chapter. In Chapter 6, it provides a list of potential future work. Finally, Chapter 7 summarizes the findings together with conclusions.

Chapter 2. SLAM, Sensors, Computer Vision and Stereoscopic Imaging in Autonomous Robot Research

In this chapter, Simultaneous Localization and Mapping (SLAM) and conventional sensors on the mobile robot are discuessed briefly. Next, the computer vision and stereoscopic imaging section provides information on normal stereo triangulation so that readers may understand the following chapters.

2.1 SLAM

In robot mobility research, the top five issues include location estimation, state estimation, localization, navigation, and motion planning [2]. A classic probabilistic approach to solve these problems is known as SLAM, which is a joint estimation method that is used to estimate robot trajectory and the location of the landmarks at once, given the control commands and observations from wheel encoders or other sensors [3][4]. SLAM is considered to be the root solution, and sensors have to provide useful inputs for SLAM algorithms. However, SLAM is not implemented in this thesis.

2.2 Active and passive sensors

Sensors are needed to make robots aware of their surrounding environments and also to perform absolute or relative positioning, heading determination, collision avoidance, navigation and path planning. Table 2-1 lists the sensors that may be commonly found in UAVs, UAGs or manned automated driving vehicles. UAVs are usually fully equipped

with heading sensors because they require 6 Degrees of Freedom (DoFs) movement in 3D space. For outdoor robotic applications, the Global Navigation Satellite System (GNSS) or the more recent Real Time Kinematic (RTK) Global Positioning System (GPS) [5] predominantly provide absolute coordinates to the vehicle (either air or ground) for navigation and mission planning. Additionally, in urban areas where tall buildings may be blocking, reflecting and weakening GPS signals, autonomous vehicles are often equipped with laser scanning rangefinders or radars that can work under harsh environments. Both are robust enough to provide reliable range readings continuously. On these research vehicles, imagery sensors or cameras can also be seen, but these imagery sensors only provide axillary or non-critical measurements to the onboard computer. The challenges of using cameras in the outdoor environment will be discussed in Chapter 4.

Unlike the outdoor autonomous vehicles that are in favor of using robust active sensors, indoor autonomous robots usually have a lower traveling speed and lower payload carrying capacities. Accordingly, bumpers with micro switches, IR rangefinders, and Hall-effect wheel encoders are necessary sensors for mainstream domestic service robots, *i.e.*, indoor robotic cleaners. Some high-end domestic service robots are supplied with active optical sensors or RF beacons in order to provide guidelines or virtual walls for robot navigation.

Table 2-1 Classification of sensors used in mobile robotics applications, adapted from [2]

General Classification	Sensor	Active(A) or Passive (P)
Tactile sensors (Detection of physical contact or closeness)	Micro switches, bumpers	P
Wheel/motor sensors (motor speed and position)	Potentiometers Optical encoders Magnetic encoders	P A A
Heading sensors (Orientation of the robot about a fixed reference frame)	Compasses Gyroscopes Accelerometers	P P P
Ground-based beacons (Localization in a fixed reference frame)	GNSS(RTK GPS/GLONASS/BeiDou) Active optical or Radio frequency(RF) beacons	A A
Active ranging (reflectivity, time-of-flight (ToF))	Ultrasonic sensors IR rangefinders Laser rangefinders/scanners/Lidar Radio detection and ranging(Radar)	A A A A
Vision-based sensors (Visual triangulation, segmentation, object recognition and classification)	CCD/CMOS cameras Optical flow sensors Structured light RGB-D cameras	P P/A A A

Generally, Time of Flight (ToF) sensors are used for relative positioning in collision avoidance and navigation. The laser scanner is commonly found in high-end robotic applications or autonomous cars, and it provides fast and accurate 2D spatial information. Similar to other optical-based sensors, the laser scanner cannot work on transparent or reflective surfaces. It requires an extremely high sampling rate in order to sample the close range return signal and this is due to the short light traveling time. Since the ultrasound range sensor and IR range sensor both have a wider sound or light beam width, they normally only provide 1D distance measurements and they are useful for obstacle avoidance for short ranges. An array can be constructed for 2D/3D range sensing, but the

issue of crosstalk may arise among sensors. Additionally, an ultrasound sensor is prone to erroneously react to loud audible noise, and an IR range sensor generally has a very short measurement range.

In indoor robotic research, vision-based sensing is a popular and challenging subject. The monocular camera may be used to perform depth estimation by varying the baseline [6] or by detecting the moving features [7]. Meaning that the monocular camera requires to register at least two states of the camera in space. Later in this chapter, the central concept in stereoscopic vision of the 1D depth estimation method is outlined. However, the 1D depth estimation method can be further extended to perform 2D/3D scene reconstruction and provides extra information than 1D ToF sensors. This will be discussed in Chapter 5.

Notably, the recent development of using RGB-D cameras in conjunction with SLAM [8][9] is another popular area of camera sensing research. The RGB-D camera, patented by PrimeSense [10], is a structured light sensor and it does not work based on the ToF principle. The IR projector emits predefined speckle reference patterns, and the IR camera then captures the laterally-shifted patterns from the projected surface. The on-board processor determines the depth of regions by comparing the reference patterns window-by-window. The dense depth map is then correlated and fused to the calibrated RGB image, and the 3D scene is reconstructed [11][12]. The RGB-D camera is a special case of stereo camera configuration in which one camera is replaced by an IR projector. In the most recent literature on sensors, the vision system was found to play a critical role in the robotic

systems, and the power of the vision system is expected to play an increasingly important role in residential robot applications.

2.3 Computer vision and stereo vision

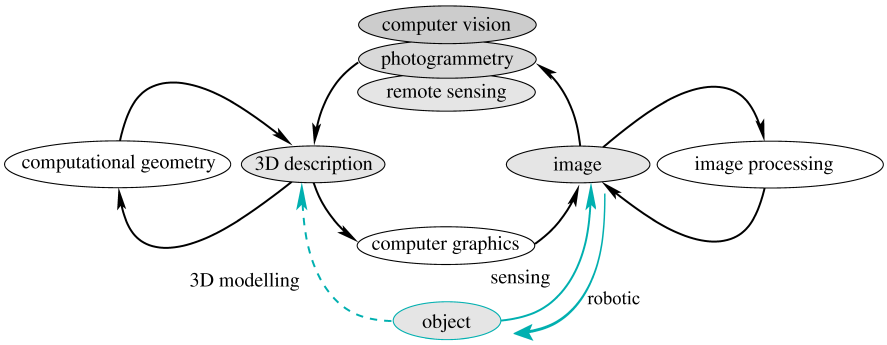


Figure 2-1 Computer vision and its applications in multi-disciplines, adapted from [13]

Computer vision is a topic derived from computer science and as shown in Figure 2-1, computer vision can be considered to be a sub-discipline within engineering. 3D modeling or Computer Aided Design (CAD) is a typical application of computer vision in the engineering domain that is used to describe an object in 3D space. In robotic applications, interpreting or inferring images back to object locations or distances by triangulating the scene or the objects, is quite challenging. To obtain depth estimations, the disparity on each pixel is first calculated to find a disparity map. Camera intrinsic and extrinsic parameters are then incorporated in order to calculate the physical distances or depth maps from object voxels to the reference image plane.

2.3.1 Digital image format, image compression, and color space

The Joint Photographic Experts Group (JPEG encoder) standard is a well-known format of lossy compression for digital images. The encoder applies Discrete Cosine Transformation (DCT) to the image and then transfers it from the time domain to the frequency domain. Meanwhile, during the quantization, a frequency coefficient (weight) is assigned to each frequency content. Since the human psycho-visual system is insensitive to high-frequency contents, these frequency coefficients are adjustable in order to achieve various compression ratios [14]. Over compression, excessive down sampling or repetitive decoding/encoding result in ringing artifacts and blurring contents and structures in the image. For example, the image may be blurred near tree branches (high frequency or high contrast contents). Although the JPEG encoder has a lossless standard, most consumer products trade off picture quality for compact file sizes. On the other hand, Portable Network Graphics (PNG) is a widely used lossless image compression method. PNG applies lossless data compression EFLATE coding, which combines an LZ77 algorithm and Huffman coding [15]. PNG preserves sharp transitions over the lossy JPEG format, in which sharp transitions often describe the edges or key features as part of the image content.

Red, Green and Blue (RGB) color encoding is used to digitize color images. Each pixel takes 24 bits (three color channels × 8 bit per channel) to store, and this creates approximately 16.7 million different colors for a pixel. To save processing time, RGB color space is converted to a single grayscale channel or intensity image so that each pixel only occupies 8 bits of space [16]. Inevitably, this conversion imposes tradeoff color spectrums for storage and time consideration.

In this thesis, the test images are captured by Raspberry Pi camera modules using Picamera [17] and are from Middlebury stereo datasets [18], and all of the images are encoded using a PNG format. For further processing, the RGB images are converted to grayscale in Matlab by a weighted sum of values of RGB channels $0.2989 \times R + 0.5870 \times G + 0.1140 \times B$ [19].

2.3.2 CCD and CMOS sensors

The Charge-Coupled-Device (CCD) and Complementary-Metal-Oxide-Semiconductor (CMOS) are both image sensors that convert light into electrons, and they then quantize analog readings into digital signals. In CCD, electronics are transported to one corner of the CCD chip across the 2D cell arrays, where the analog-to-digital converter converts the analog signal. The CCD produces higher quality and high sensitive images, but it is more expensive to manufacture. On the other hand, the CMOS reads each pixel's signal individually using phototransistors [20]. There are two types of digital shutter or triggering modes that are commonly used in these sensors for static images capturing. The CCD sensor usually has a global shutter, while the CMOS usually uses a rolling shutter. The rolling shutter scans 2D cell arrays line by line with an offset time. Thus, not all parts of the scenes are recorded concurrently, and this results in spatial distortions of the object in an image when the object in the scene is moving fast. On the contrary, the global shutter reads out all of the pixels in the whole image at once and it provides a uniform 3D to 2D projection at the same time for all pixels. In the past, the CCD with a global shutter was the only choice in machine vision applications. Nowadays, there are new CMOS sensors that are cheaper and have a higher parallel processing speed/bandwidth with less noise.

Some CMOS sensors have both a rolling shutter and global shutter and an example of this PointGray® that now offers CMOS machine vision cameras with the global shutter enabled. In this thesis, the image capture device that is used is the IMX219PQ 8M pixel color CMOS image sensor from SONY [21], and this device provides both a rolling and global shutter, but the only the rolling shutter is enabled by the Raspberry Pi camera module.

2.3.3 Basis of stereo normal, triangulation and depth estimation

A key issue to be solved in this thesis consists of how to interpret two images and get the distance measurements from certain objects to the reference image. A common method is to use a camera to capture one image as a reference image, and the camera is then shifted to another position in which the new optical axis is parallel to the previous camera optical axis. Another picture is then taken as a matching image. The distance of the axis translation has to be known, and the so-called baseline and camera orientations in two positions must be identical. Thus, a 3D scene is cast onto two horizontally-shifted perspectives. The object distance in the reference image is then estimated. The calculation requires to find the locations of the same feature voxel on the reference image and the matching image along the epipolar line. The disparity or parallax is then calculated using the difference between the 2 pixel locations on the reference image and matching image horizontally. To obtain the distance from a feature voxel (*i.e.*, a corner) in the space to the reference image plane, a physical scale, *i.e.*, baseline distance is multiplied by focal length (in pixels) and then divided by the disparity of that feature point. A higher disparity value results in a closer range.

As seen in Figure 2-2, the lens is modeled as a pinhole lens or a thin lens to simplify the calculation. Moreover, the relationship between focal length (f), image distance (z) and object distance (Z) is shown in Equation 2.1. In the pinhole lens configuration, the object distance is usually much larger than the image distance ($Z >> z$), which turns Equation 2.1 into Equation 2.2. Thus, the image distance is approximately equal to focal length. However, in this case, the image plane does not coincide with the focal plane, and the light reflected from the object voxel appears as a blur circle on the image plane. The radius of the circle R is illustrated in Equation 2.3, where L_A is the diameter of the aperture and δ is the offset of the image plane. Thus, when aperture L_A and offset δ are small enough, the radius of the blur circle approaches zero. This is the result of an increase in the depth of field (sharper image) with the trade-off of allowing less light to reach the sensor.

$$\frac{1}{f} = \frac{1}{z} - \frac{1}{Z} \text{ [22]} \tag{2.1}$$

$$\frac{1}{f} = \frac{1}{z} \rightarrow f \approx z \tag{2.2}$$

$$R = \frac{L_A \delta}{2z} \text{ [2]} \tag{2.3}$$

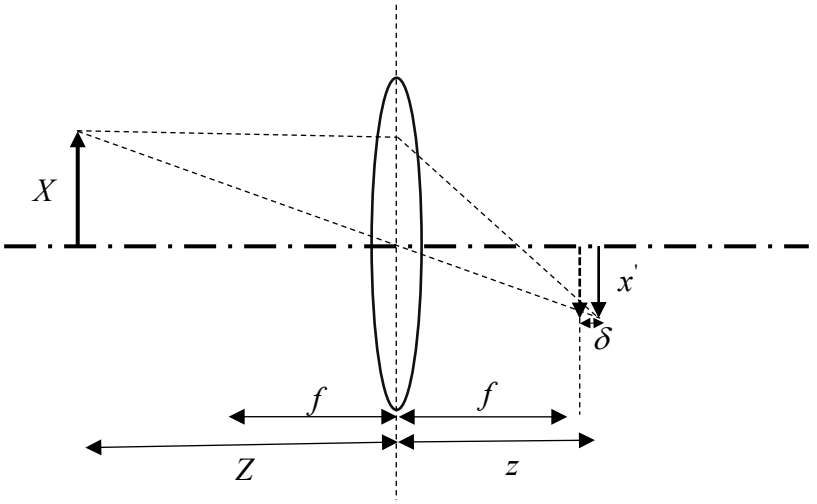


Figure 2-2 Thin lens model

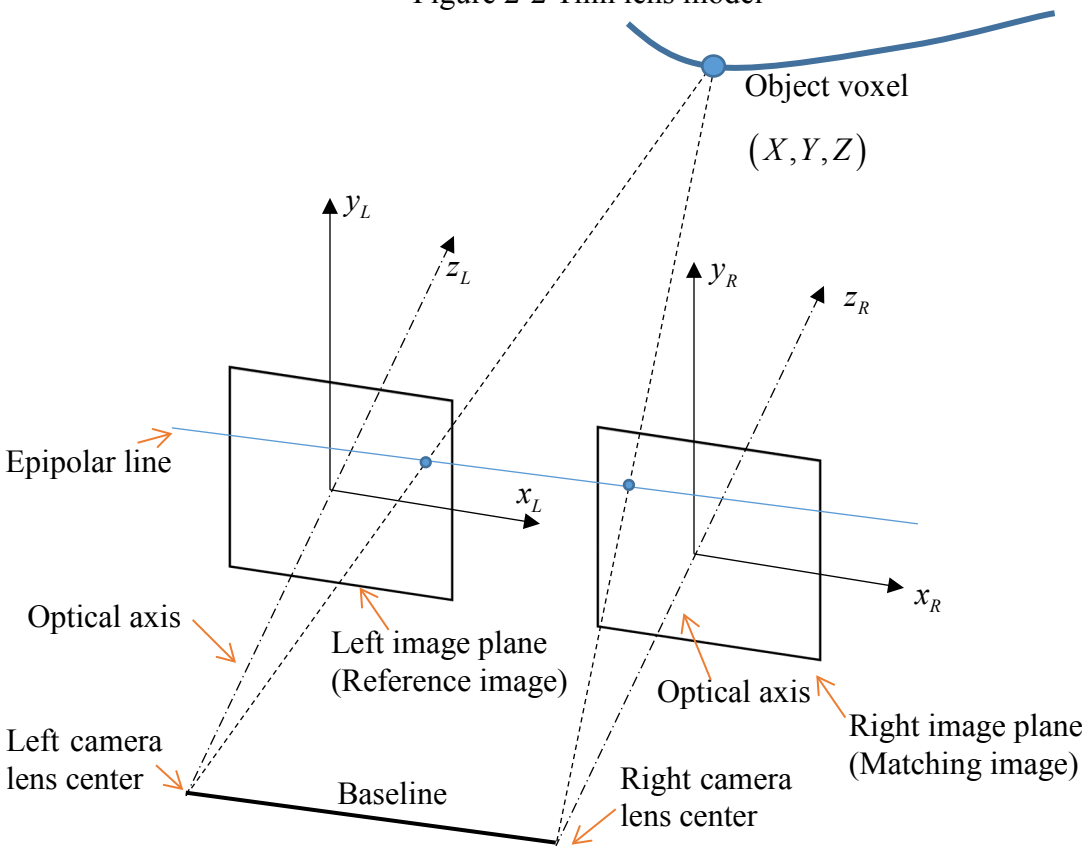


Figure 2-3 Stereo normal isometric view

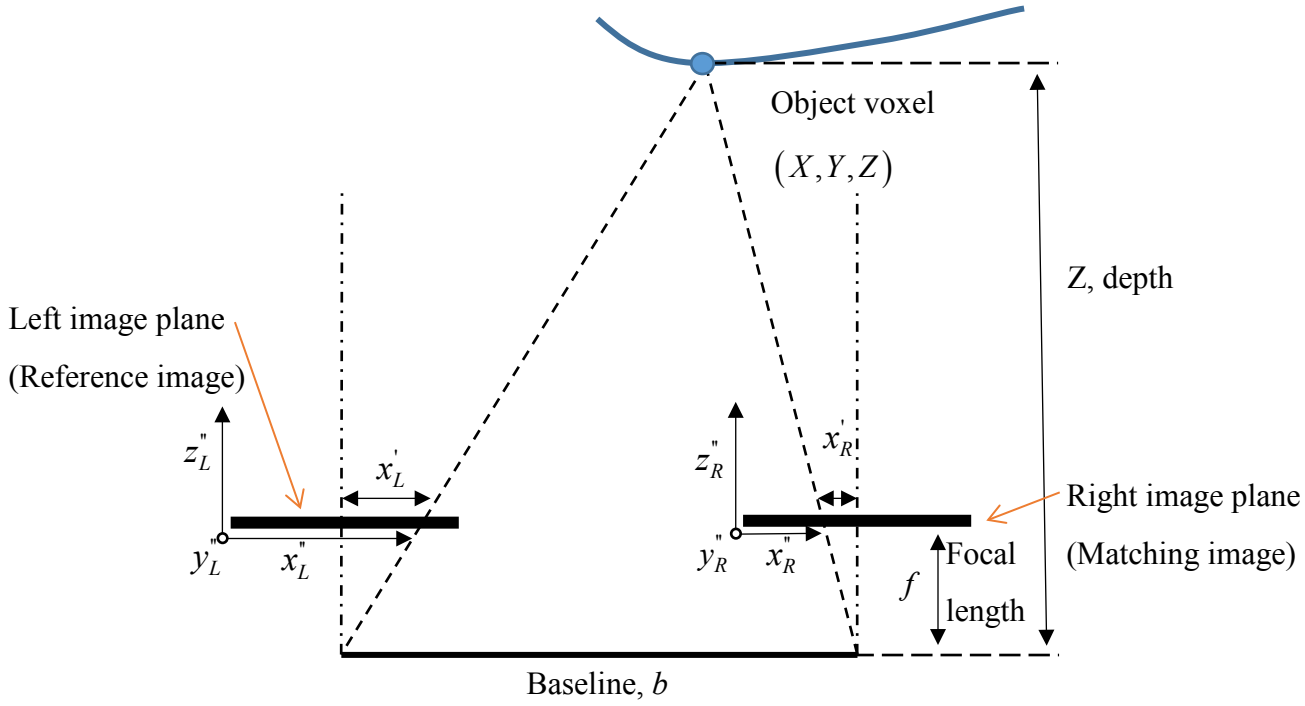


Figure 2-4 Stereo normal top view

Secondly, a normal stereo case is assumed. In this case, two cameras are apart with a baseline distance *d,* and the two cameras are assumed to have identical ideal sensors and lenses. As shown in Figure 2-3, the two cameras are also assumed to be in perfect alignment. The image plane usually appears behind the lens center but, for visualization, this image plane is moved so that it is between the object and the lens center. Applying similar triangles on Figure 2-3 and Figure 2-4, we obtain:

$$\frac{Z}{f}=\frac{X}{x_L'},\ \text{and}\ \frac{Z}{f}=\frac{X-b}{x_R'} \quad [22] \qquad (2.4)$$
$$\frac{Z}{f}=\frac{Y}{y_L'}=\frac{Y}{y_R'}$$

where the origin of object voxel coincides with the left camera lens center. Equation 2.4 becomes

$$Z = \frac{fb}{x_L^{'} - x_R^{'}}, Y = \frac{y_L^{'} b}{x_L^{'} - x_R^{'}}, X = \frac{x_L^{'} b}{x_L^{'} - x_R^{'}}$$

To simplify the calculation, new $x_L^{"}$, $y_L^{"}$ and $x_R^{"}$, $y_R^{"}$ coordinate systems are introduced in which the origins of $x_L^{"}$, $y_L^{"}$ and $x_R^{"}$, $y_R^{"}$ are located at the left top corner of the image planes (with the positive *x*-axis pointing right, and positive *y*-axis pointing downwards, and assuming the image plane has *T* by *T* pixels). Thus, the coordinates of the object in the space can be demonstrated by Equation 2.5.

$$Z = \frac{fb}{x_L^{"} - x_R^{"}}, Y = \frac{\left(\frac{T}{2} - y_L^{"}\right) b}{x_L^{"} - x_R^{"}}, X = \frac{\left(x_L^{"} - \frac{T}{2}\right) b}{x_L^{"} - x_R^{"}} \tag{2.5}$$

where *X, Y, Z,* and *b* are in centimeters, f, $x_L^{"}$, $x_R^{"}$, $y_L^{"}$, $y_R^{"}$ are in pixels, and $x_L^{"} - x_R^{"}$ is the disparity or parallax.

In Chapter 5, a more general case of stereo triangulation with lens distortion is discussed because it is rare to have two stereo images in perfect alignment.

Chapter 3. Experimental Mobile Platform Design, Setup, and Image Acquiring

The focus of this thesis is on stereoscopic vision for indoor applications. Therefore, the primary function of the experimental mobile platform is to take pictures and videos and process the graphic information either onboard or on the ground host computer. I propose an innovated approach to build an indoor mobile robot research platform to facilitate versatile and collaborative research by using open-source and off-the-shelf hardware. In this chapter, the details of the hardware selection and building of the platform are discussed.

3.1 Design requirements

This research focuses on indoor robotic applications. Off-road robotic platforms are dismissible and the following requirements must be met.

Structural and mechanical requirements:

- A heavy duty, low profile, pancake shape that has a diameter of at least 30 cm
- A large mounting area for cameras, sensors, batteries, onboard computer
- Stackable for future expansion
- The gross weight, including battery(s), electronics and onboard computer, must be less than 5 kg
- A differential drive system that is compatible to run on hard floor surfaces or carpets

- A maximum moving speed of 50 cm/s when the basic electronics only are loaded, a minimum speed of 20 cm/s under a 5 kg load, when running on hard surfaces
- Use of as many off-the-shelf parts as possible to lessen building time
- Spare parts are available to purchase
- Lithium-based battery power
- The main structure must be made from non-ferrous material

Onboard computer and electronics requirements:

- A 1Gb/s high-speed wireless communication (IEEE 802.11AC or newer standards) between the onboard computer and ground host computer
- The onboard computer must have a powerful X86 based CPU, and it must be light-weighted and powered by battery; avoid laptop if possible to save weight on redundant screen and battery
- Resources on the onboard computer must be manageable and supervised
- A stereo camera system with hardware triggering and small footprint
- Open-source parts are taken into consideration, but they must be popular or commonly used in the community for long-term support
- Wheel modules must have encoders with output capability
- Possible to integrate Inertial Measurement Units (IMUs) or digital compass
- Possible Field Programmable Gate Array (FPGA) implementation

Software considerations:

- Good for collaborative research
- Easy to backup and restore systems

3.2 Market research

After extensive Internet research, there are two options available. These are directly buying an off-the-shelf mobile platform as a whole package or constructing the platform by using off-the-shelf parts. In the first scenario, there are two possible selections as Q1 of 2015. These platforms include wheel modules, a mobile platform mainframe, and necessary hardware. The onboard X86 computer is not included, and the end-users must implement their mechanical driving controller design (sample codes are available from manufacturers). The Arlo mobile platform is made by a U.S. based company, Parallax®. The diameter of the curly top surface is 45 cm and it has a 27 kg load capacity. The cost starts from USD $1195. The second choice is the DFRobot's™ Home Care Robot (HCR) platform, which is from China. It has a diameter of 33 cm, 3 level stackable structures and a 10 kg load capacity. Its cost starts from USD $530. Alternatively, some previous research [23][24] on iRobot® Roomba® shows that building a research platform from scratch is more convenient and affordable.

3.3 Detailed system design, build and integration

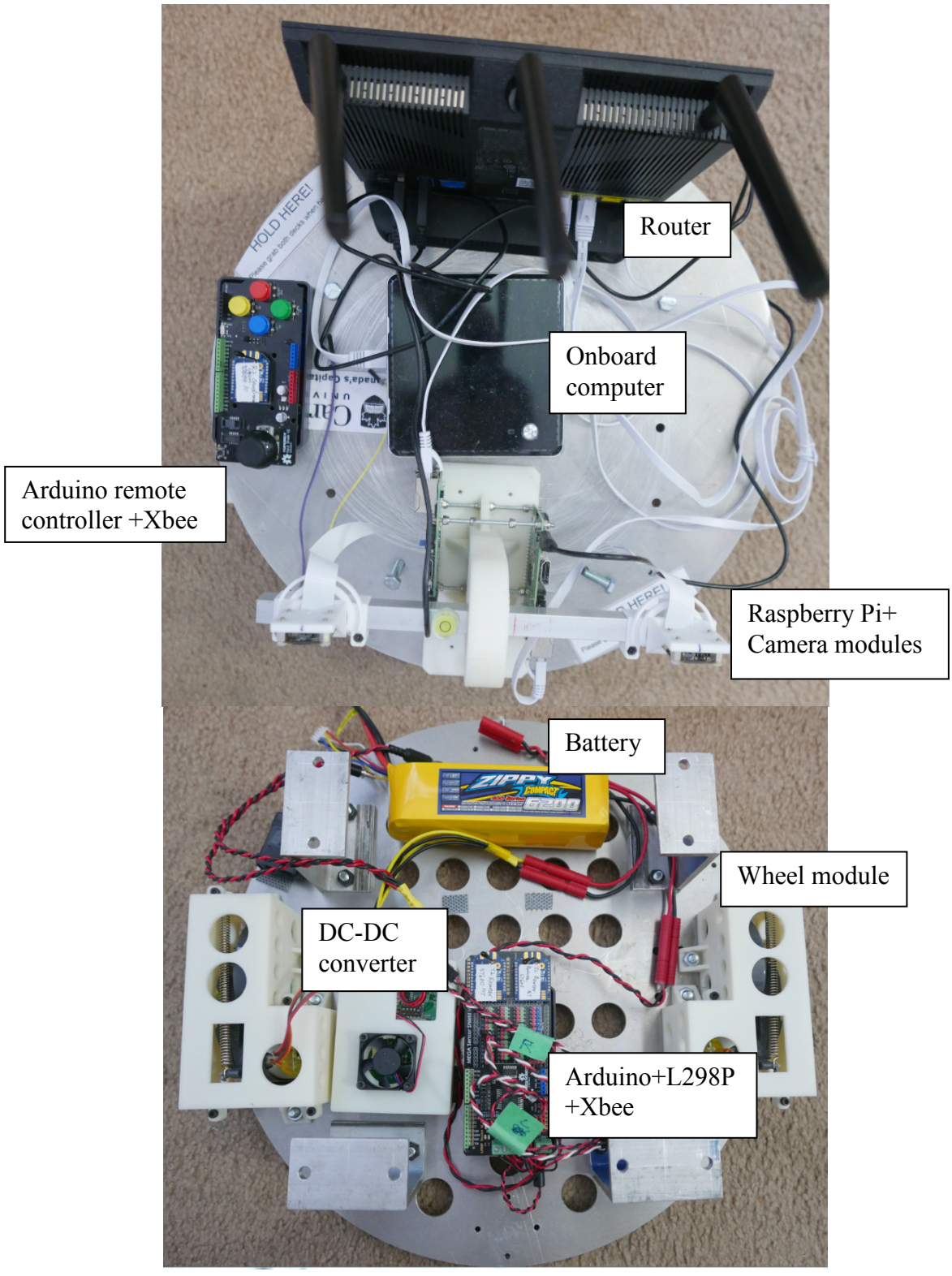


Figure 3-1 Proposed indoor service robot Level 1(top) and Level 2(bottom)

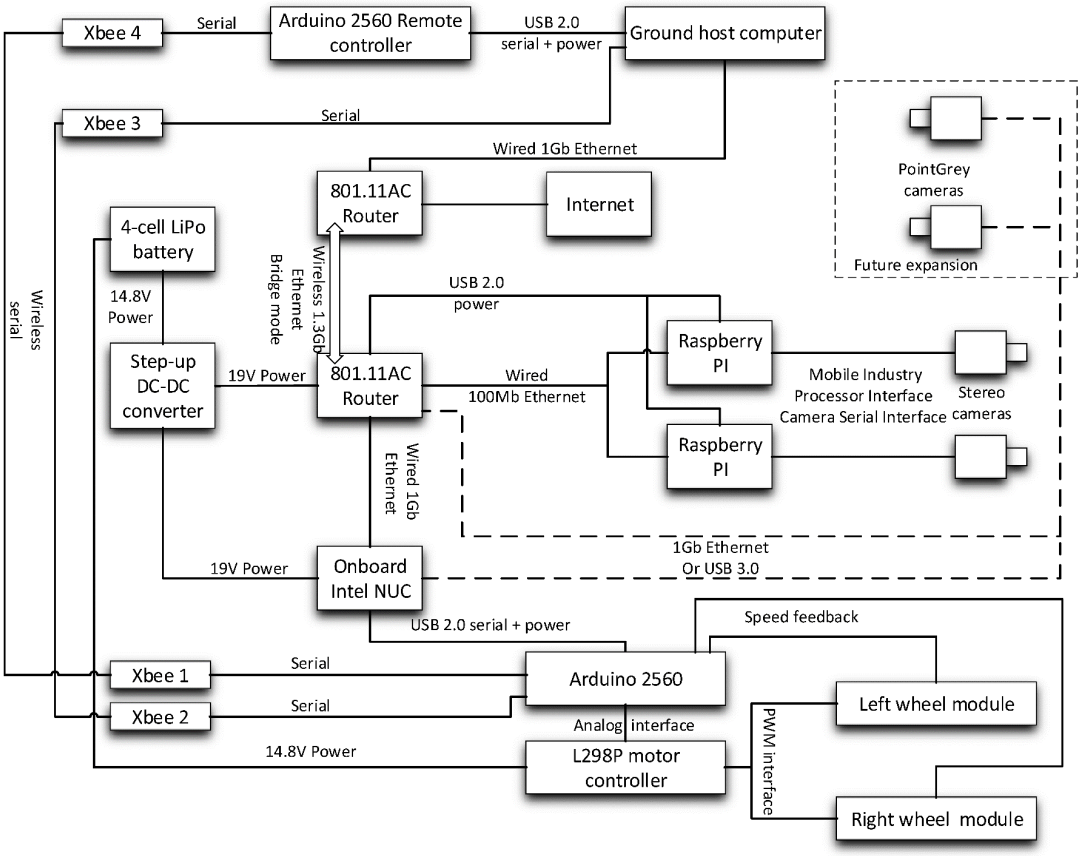


Figure 3-2 System architecture diagram

Figure 3-1 shows the implementation of the proposed system architecture in Figure 3-2. This 4.5 kg robot has a dual-level structure with a 40 cm wheelbase in a pancake appearance. In wireless communications, high-speed 802.11AC routers and low-speed ZigBee® XBee® modems handle high-speed Internet/file transfers and low-speed serial communications, respectively. The Arduino MEGA 2560 (based on Atmel® ATmega 2560), L2980 2A H-bridge dual channel motor controller shield and Raspberry Pi 3B camera module are open-source components. They are modular (with stackable shields) and have user-friendly libraries. The onboard Arduino 2560 mainly controls the

mechanical differential drive system with a L298P motor controller. Raspberry Pi and the camera modules are adapted to the image acquisition devices. The other Arduino 2560 has an attached joystick shield, and the Arduino 2560 and the joystick together functions as a remote controller to manually control the motion of the platform. High dimensional or complex data processing tasks are assigned to the onboard computer, or they are sent to the ground host computer for further processing. The entire system is powered by a 6200 mAh 14.8V 4-cell Lithium Polymer (LiPo) battery. The follow sections discuss the detailed design of this robotic framework.

3.3.1 Mechanical drive controller design and implementation

The mechanical drive system is adapted from a pair of wheel modules for the iRobot Roomba 500/600/700 series. The gear ratio is determined to be 63.6:1, and a 4-pole magnetic disk is directly attached to the motor shaft, coupled with a 5V Hall-effect encoder mounted on the back of the motor shell. Hence, for one revolution of the wheel rotation, the encoder will register approximately 254 impulses. The diameter of the rubber wheel is 70 mm. This drive system has been tested under a power input of 14.8V. The Proportional–integral (PI) controller is implemented in Arduino 2560 with hardware L298P the motor controller shield for each motor channel to control the differential drive speed. The following discussion starts with determining the kinematics of different drive system. The motor model is then established for each drive module and P, I parameters are determined. The PI controller is implemented in C++ and uploaded to Arduino. Finally, the performance of the PI controller is evaluated at 20 cm/s and 30 cm/s.

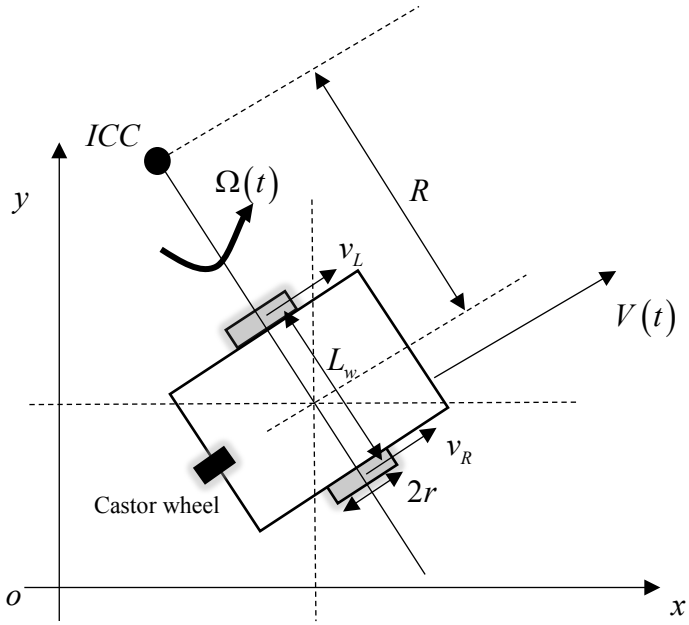


Figure 3-3 Schematic of differential drive mobile robot, adapted from [25]

In Figure 3-3, a schematic of the two-wheel differential drive system with one additional point of contact (castor wheel) system is shown traveling about the Instantaneous Center of Curvature (ICC). The individual wheel speed is modeled as Equation 3.1, and the kinematics of traveling about the ICC is shown in Equation 3.2. Hence, using Equation 3.1, the robot maneuver trajectory is controlled by the output speed of the single motor controller $v_L(t)$ and $v_R(t)$. It should be noted that the angular and triangular velocity about ICC is independent of the curvature radius (R) of the trajectory.

$$\begin{aligned} v_L(t) &= \frac{2V(t) - L_w\Omega(t)}{2} \\ v_R(t) &= \frac{2V(t) + L_w\Omega(t)}{2} \end{aligned} \qquad (3.1)$$

$$\begin{aligned} \Omega(t) &= \frac{v_L(t) - v_R(t)}{L_w} \\ V(t) &= \frac{v_L(t) + v_R(t)}{2} \end{aligned} \qquad (3.2)$$

Where: $\Omega(t)$: Angular velocity about ICC rad/s

$v_L(t), v_R(t)$: Left and right wheel speed m/s

$V(t)$: Tangential velocity about ICC m/s

L_w : Distance between two wheels m

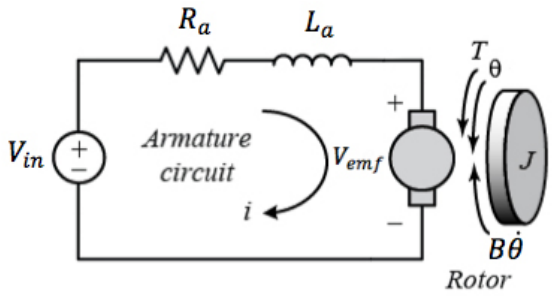


Figure 3-4 Permanent magnet motor electro-mechanical schematic [26]

Using Kirchhoff's voltage law and Newton's 2nd law of motion, the dynamics and electrical relations of the motor in Figure 3-4 are modeled as Equation 3.3. Other electrical and mechanical properties are coupled through Equation 3.4. By rearranging Equation 3.3 and Equation 3.4 and taking the Laplace transfer of the resulting function, then substituting parameters in Table 3-1, one will obtain Equation 3.5. This equation represents the relation between input voltage and output angular velocity in the continuous frequency domain.

$$\begin{aligned} V_{in} - R_a i - L_a \frac{di}{dt} - V_{emf} &= 0 \\ J\ddot{\theta} &= T - B\dot{\theta} \end{aligned} \tag{3.3}$$

$$V_{emf} = K_{emf}\dot{\theta}\ ,\ T = K_t i\ ,\ K_{emf} = K_t = K \tag{3.4}$$

Table 3-1 Motor parameters and load conditions for PI controller design, adapted from [23][24]

Variable	Description	Numerical value
R_a	Motor resistance	8.4Ω
L_a	Motor inductance	$0.0084H$
J	Moment of Inertial of the rotor	$7.568\times10^{-7}\,kg\cdot m^2$
B	Motor viscous friction constant	$2.89\cdot10^{-6}\,N\cdot m\cdot s$
Kt	Motor torque constant	$0.016N\cdot m/A$
K_{emf}	Motor electromotive force constant	$0.016V\cdot s/rad$
V_{in}	Input voltage (V)	N.A.
i	Current (A)	N.A.
$\dot{\theta}$	Angular velocity of the motor (rad/s)	N.A.
$\ddot{\theta}$	Angular acceleration of the motor (rad/s^2)	N.A.
V_{emf}	Back electromotive force voltage (V)	N.A.
T	Torque of the motor shaft $(N\cdot m)$	N.A.
Ω	Angular velocity of the motor in s domain	N.A.
V_{in}	Input voltage in s domain (V)	N.A.

$$\frac{\Omega(s)}{V_{in}(s)} = \frac{K}{L_aJs^2 + (L_aB + R_aJ)s + (R_aB + K^2)} = \frac{0.016}{6.306\times10^{-9}s^2 + 6.381\times10^{-6}s + 2.803\times10^{-4}} \tag{3.5}$$

Zero order hold at 0.05s (PI controller is running at 20Hz) is applied to Equation 3.5 to obtain the discrete transfer function in Equation 3.6. This causal system has all poles inside the unit circle, which implies that this system is stable.

$$\frac{\Omega(z)}{V_{in}(z)} = \frac{33.7z^2 + 17.66z + 5.923\cdot10^{-4}}{z^2 - 0.1z} \tag{3.6}$$

Equation 3.6 is then implemented in Matlab Simulink®, and the simulated results, at 20 cm/s and 30 cm/s target velocities, are depicted in Figure 3-5. The PI controller parameters are determined as P=30, I=70. The saturation is set to nominal battery voltage at 14.8V. Two gains after the step input and before the scope output are used to convert between linear velocity and angular velocity according to the gear ratio and wheel diameter.

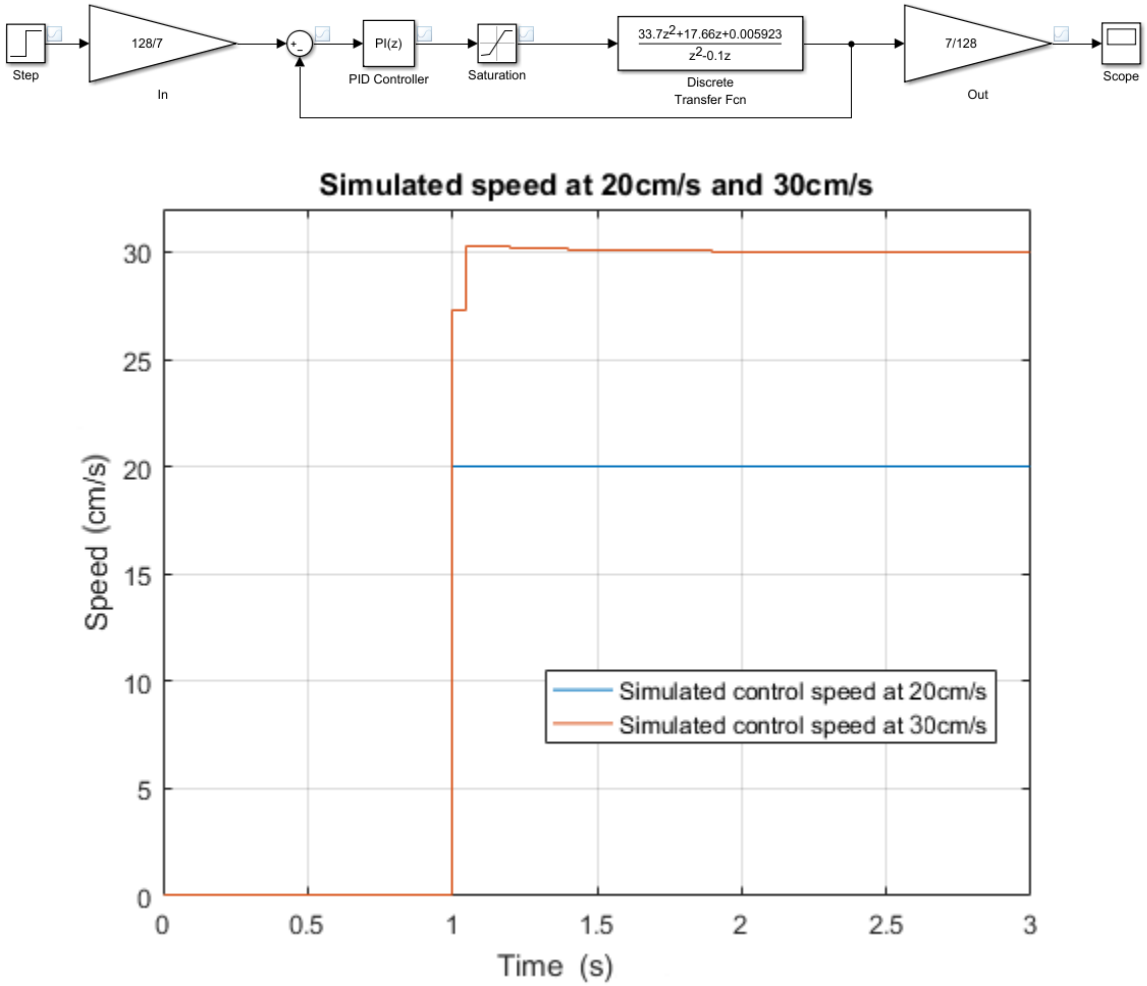


Figure 3-5 Simulation of discreet controller model and discrete PI controller (top) and simulation results at 20 cm/s and 30 cm/s (bottom)

Now, this PI controller is implemented in C++ [1] by using the Arduino PID library [27], and each motor has its own controller. The controller is capable of monitoring the wheel rotation in both directions. Hence, it has adaptive control when the desired speed is at 0 cm/s. The speed feedback is calculated by counting hardware interrupts that are generated by the Hall-effect wheel encoder. While doing the PI calculation to obtain the next voltage output, the interrupt is detached and then reattached once the PI controller finishes issuing the output voltage command. This reduces the working load on Arduino 2560, and it also eliminates possible missing counts during the calculation. The PI controller runs at 20Hz. Thus, the number of interrupts is sampled and counted within 0.05s. To smoothen the noisy readings, a moving average filter, with a bin size of 4, is used to filter the data in real-time. Figure 3-6 and Figure 3-7 show that the moving average filter is an efficient method to smoothen the speed in this mechanical drive system with the PI controller. From both of the filtered results, it can be seen that the system reaches its desired speed within 0.5s, and the speed is fluctuating around $\pm 5\%$ of the desired speed in the steady state. The possible contributions of the fluctuating data are coming from the dirty floor and the stiff metal castor wheel. The real-time data is obtained wirelessly from the Arduino to the onboard computer via a pair of XBee modules, and these real-time speed values may be broadcasted from the onboard Arduino 2560 to the ground host computer as well. When the platform is fully loaded, this mechanical drive system has been tested at a maximum speed of 48 cm/s.

[1] Available on https://github.com/kunzhuang/vacuum

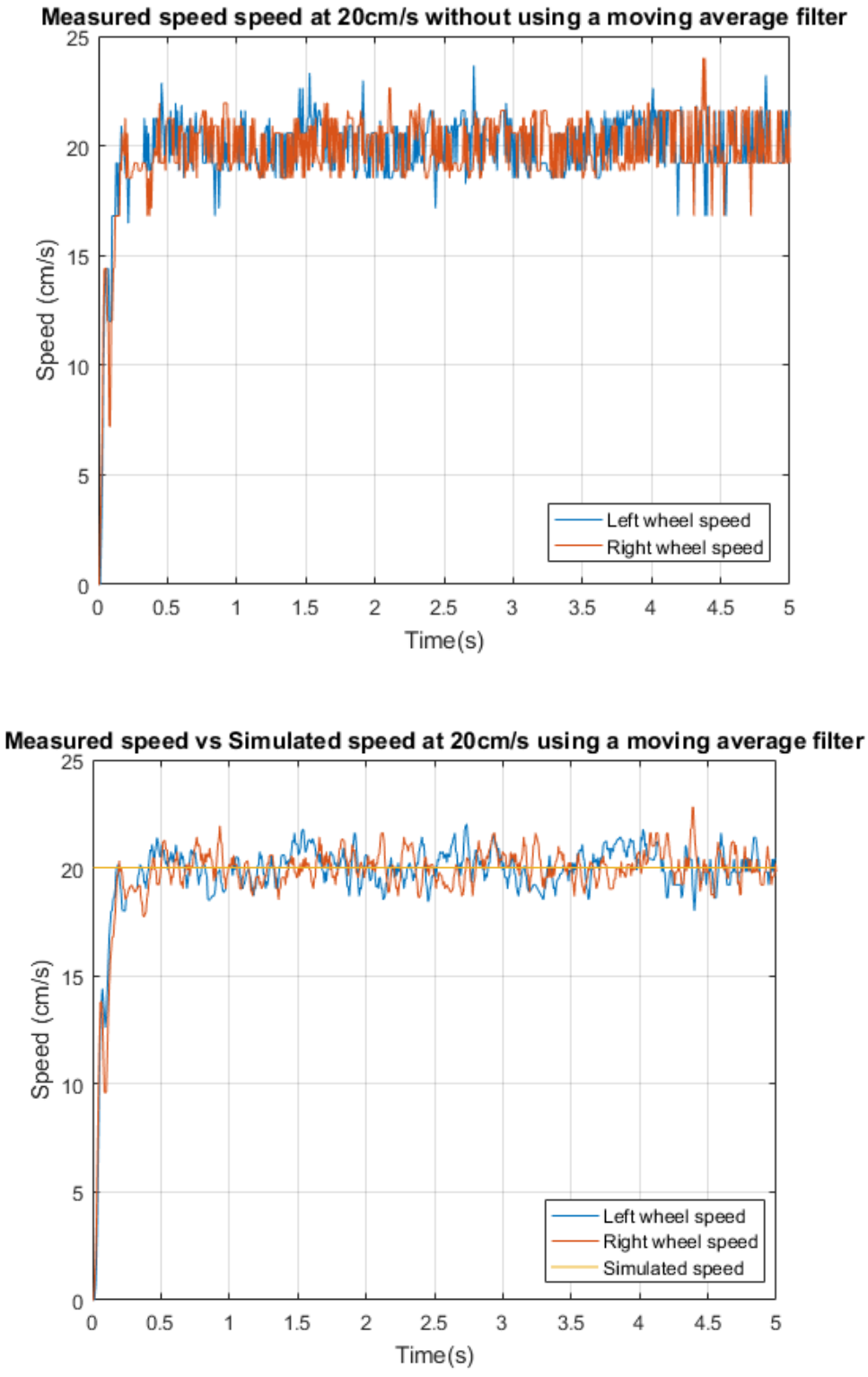


Figure 3-6 PI controller performance at target speed of 20 cm/s, without using a moving average filter (top) and using a moving average filter (bottom)

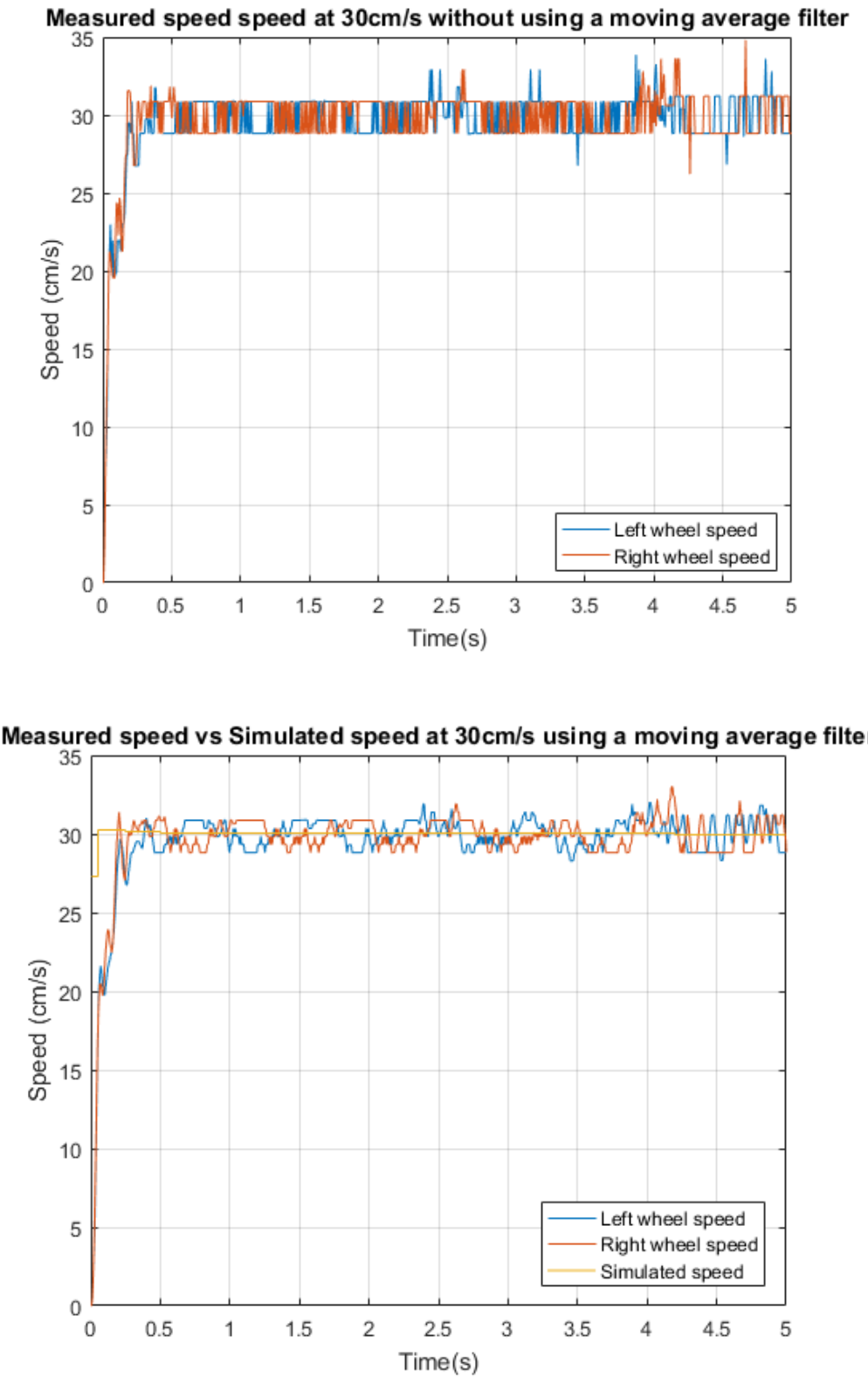


Figure 3-7 PI controller performance at target speed of 30 cm/s, without using a moving average filter (top) and using a moving average filter (bottom)

3.3.2 Mechanical structure design

The overall robot body structure is supported by two 3 mm thick and two diameter of 40 cm aluminum-alloy plates with four 68 mm high C-beams in between[2]. The chassis is designed to be durable and lightweight. The other essential part that takes the impact and lateral loads is the mechanical drive system. As shown in Figure 3-8, a pair of 3D printed motor cages is used to accommodate the wheel modules, and each motor cage is bolted to the bottom chassis directly.

Figure 3-8 A 3D printed motor cage for mechanical drive system

3.3.3 Onboard computer selection and operation systems

Recently, in robotic research, the use of an Intel® NUC™ has become popular for the onboard computer in UAVs [28], UAGs [29] and underwater vehicles [30]. This palm-sized computer has an outstanding balance design in terms of size and performance, and it

[2] Zhi Li designed and manufactured the Al alloy chassis.

runs with a 19V power supply. An Intel NUC5i7RYH with Iris™ graphics 6100 is chosen and installed on this robotic platform.

A virtual machine resembles a physical machine that runs on a real computer system, so that multiple virtual machines may be emulated concurrently. VMware® ESXi™ is an enterprise class, type-1 hypervisor developed by VMware and it is used for hosting virtual applications. It is different from the VMware Workstation™ and the Oracle® VM VirtualBox™, which must be installed in an operating system. As seen in Figure 3-9, the ESXi is similar to an operation system with essential operation system components that directly control the host's hardware and manage the guest operation systems. All of the hardware management tasks and new guest operation installations can be done via the network.

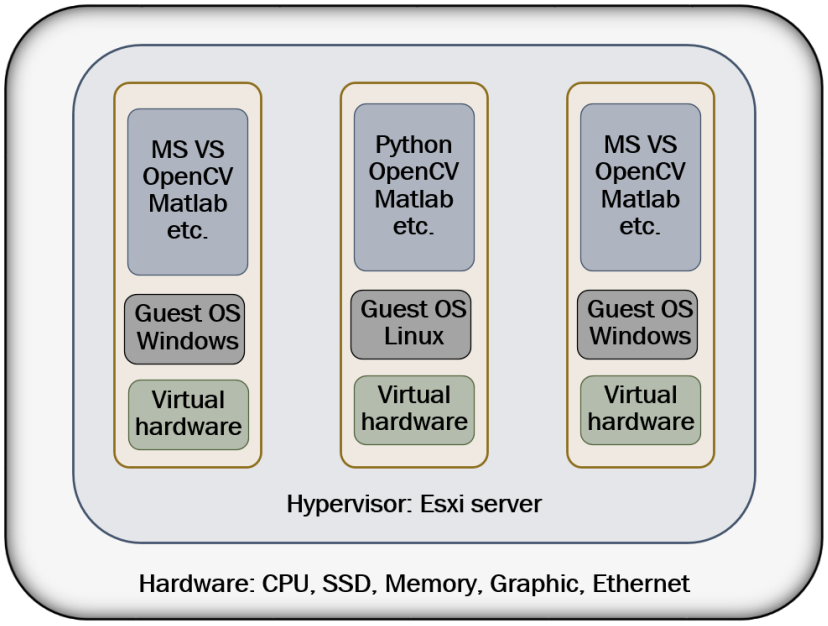


Figure 3-9 VMware ESXi software and hardware architecture

The advantage of deploying a hypervisor instead of using traditional multi-boot operation systems, is that it is easy to manage and work across different operation systems at the same time. The hypervisor allows a PC to run multiple operation systems simultaneously in a dedicated disk or memory space without having to compromise the performance too much. This means that an operation system is installed in a folder of a disk partition and this folder contains configuration files for this guest system including emulated hard drive(s) or virtual disk file(s). The memory size of each guest machine can be assigned based on the applications and tasks accordingly. Thus, each user will have his or her own working space or operation system without having to interfere or disturb other users' ongoing research. Thus, collaborative research work is possible on this platform. It is possible for a user to test a sensor fusion algorithm written in C on Microsoft® Windows® on this platform in the morning and another user to test a Python-based computer vision algorithm on Ubuntu™ on this platform in the afternoon. Each user is working under his or her dedicated environment inside the hypervisor, and the users only need to decide which operation system to boot up inside the hypervisor. The sensor configurations may or may not be changed in two tests. Since each guest system is self-contained within a folder, this folder may be retrieved to the ground host computer as archives and save disk space on the onboard computer.

Meanwhile, the other essential function is quick backup and disaster recovery. Since the majority of open-source software comes with absolutely no warranty, it is easy to misconfigure the operation systems or new applications, and this results in corrupted systems or missing key functions in an application. As shown in Figure 3-10, a snapshot

of the current operation system can be taken as a system image before installing a new application or changing key configurations on the guest system. If necessary, the snapshot can be reverted in the future. Additionally, the system can be repacked and distributed as a pre-configured system, so that other researchers can deploy the system with applications or testing algorithms on a new computer or a virtual machine.

Additionally, when working in a Graphical User Interface (GUI) environment, the guest systems can be managed by a Remote Desktop Protocol (RDP), Virtual Network Computing (VNC), VMware vSphere Client (obsoleted) or a new HTML5 based Server Appliance Management Web Console.

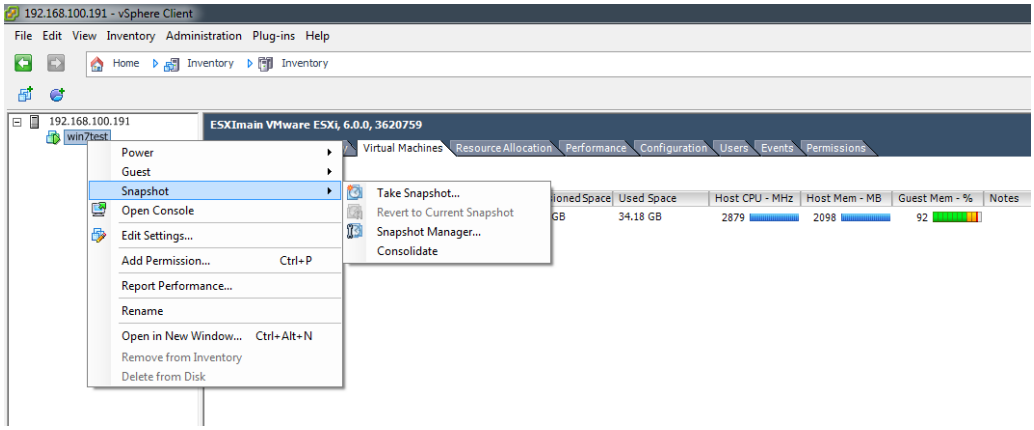


Figure 3-10 Snapshot and rapid system recovery in VMware ESXi

As mentioned, ESXi can assign any available hardware resources to the guest machines. As seen in Figure 3-9, in addition to disk or memory space management, one of the necessary hardware management operation is to pass through the hardware resources to a guest machine. Figure 3-11 indicates that a USB controller is passed to a guest machine.

Thus, this guest machine can access the USB devices, *e.g.*, Webcam or other USB interface sensors. In the guest machines, all of the emulated Ethernet adapters are in bridge mode to connect to the physical Ethernet adapter.

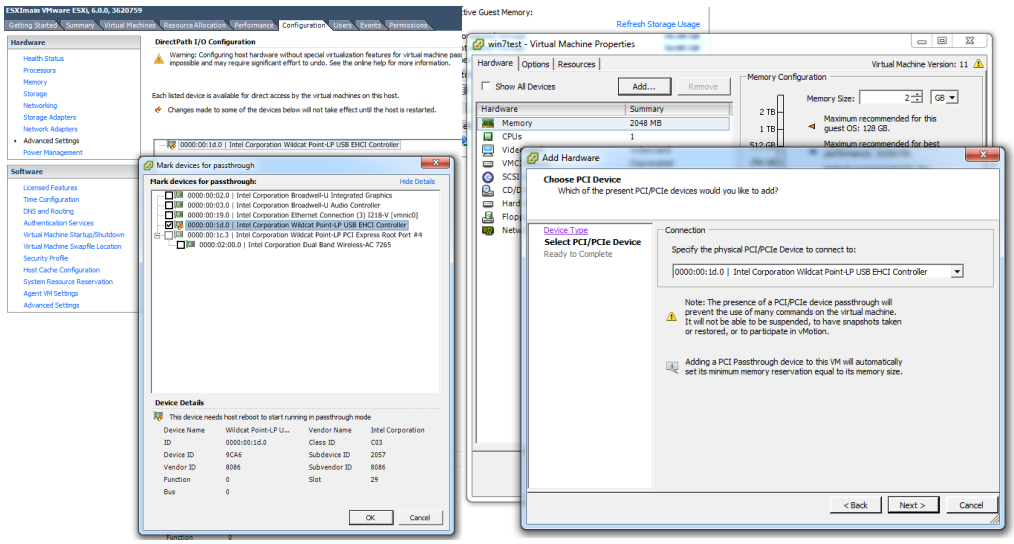


Figure 3-11 USB controller pass-through to guest operation system in ESXi

3.3.4 Stereo camera system and image acquiring

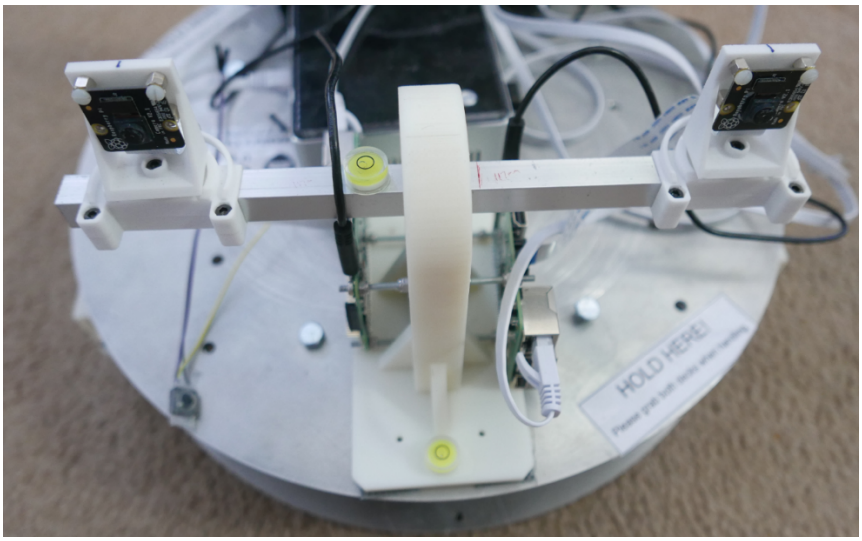


Figure 3-12 A 3D printed bracket and camera mount for Raspberry Pi single board computers and camera modules

The Raspberry Pi is a well-known single board computer in the open-source community, and several sample Pi projects are available on the Internet. The Raspberry Pi 3B features an ARM®v8 based 1.2GHz 64-bit quad-core Broadcom® BCM2837 system on a chip (SoC) and 1GB of memory. The Raspberry Pi 3B functions very differently than most of the microcontroller, and it is capable of handling more complex programs and external peripherals *e.g.,* USB devices, cameras, and monitors. Given its size and versatility, two Raspberry Pi, each with a SONY IMX219PQ camera module that can be found on some mobile devices, are used as a stereo camera rig and they are tested on the robotic platform. A 3D printed bracket supports two Raspberry Pi camera modules, and the distance from the ground to the camera is measured to be 27 cm. As shown in Figure 3-13, the distance or baseline between the two camera mounts is adjustable along the aluminum bar between 8 cm and 20 cm, and the camera mount allows the camera to rotate in the horizontal plane.

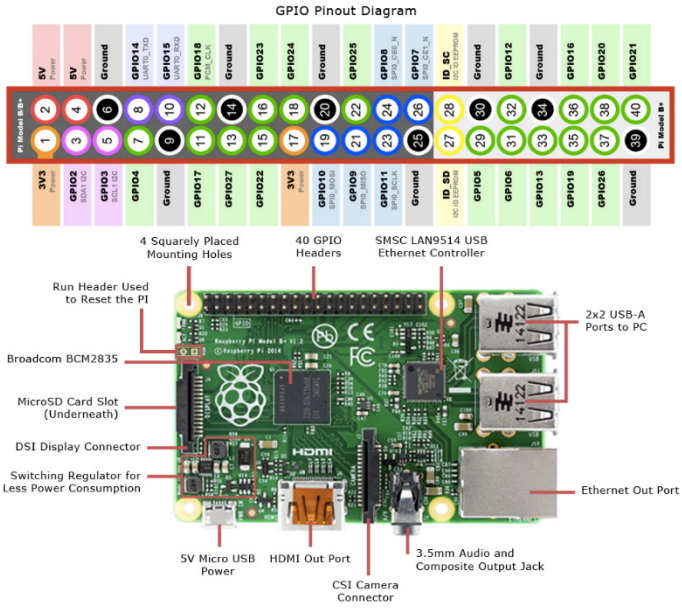


Figure 3-13 Raspberry Pi 3B GPIO Pin definitions [31]

As illustrated in Figure 3-13, the Raspberry Pi 3B has 40 general-purpose input/output (GPIO) pins in order to interact with external sensors or circuitries. A Python script is created to monitor whether the GPIO23 pin is shorted to the ground by the RPi.GPIO [32] class and if the pin 23 is shorted to the ground and the script executes capturing commands in Picamera class (supports camera control and image encoding). To trigger the two cameras simultaneously, a push-button acting, as a manual trigger, is connected between the ground and the GPIO23 pins on each of the Raspberry Pi with a simple resistor-capacitor de-bounce circuit in between. This may be triggered by the onboard Arduino 2560 digitally as well. The Raspberry Pi, with the camera module, acts as an image-capturing device only. However, the post processing of the stereo images will not be running on Raspberry Pi.

CMOS is an infrared-sensitive device. The Raspberry Pi camera module supplies two types of camera modules with and without an IR filter, respectively. The one without an IR filter is more suitable for indoor or low light conditions but there is poor color rendition. Four sets of images, at a baseline of 20 cm apart and without an IR filter, are captured. They have resolutions of 1640 x 922 and 3280 x 2464. And with an IR filter, the resolutions are 1640 x 922 and 3280 x 2464 as well. At each resolution and filter configuration, several images with checkerboard presented in the scene are also captured for stereo calibration and lens calibration. All of the images are encoded in a PNG format to preserve maximum edge information. Once a picture is captured, it is immediately transferred to a shared folder on the onboard computer via the Microsoft Common Internet File System (CIFS). As shown in the stereo normal case, both of the cameras are oriented in almost perfect

alignment and the aluminum alloy bar that supports the camera mounts is at a level during the still image capture. The focus length of the camera modules is manually adjusted to be 75 cm.

3.3.5 Communications and electronic integration

A pair of Asus® RT-AC68UTM routers are arranged in bridge mode to provide high-speed communication between the robot mobile platform and the ground host computer. All of the Ethernet devices are configured under the same subnet including the emulated Ethernet adapter on each guest operation system in the onboard computer. Thus, two Raspberry Pi, VMware ESXi and guest machines are manageable via the Ethernet, and these onboard devices are connected to the router using Ethernet cables.

Each pair of XBee (XBP24BZ7WIT) modems is configured in peer-to-peer mode (one coordinator to one end device) for low-speed serial communication (RS232) within line of sight. The host-guest mode may be configured (one coordinator to multiple end devices) for broadcasting sensor readings or issuing control commands. In the current setup, the Arduino 2560 remote controller is capable of sending messages (ASCII), such as "S0M1L15R30D2000E," to control Arduino 2560 on the platform manually. Next, the onboard Arduino 2560 will decode this command message. The message is sent from the #0 serial device to make the differential drive system work in mode 1 (reserved parameter, not used), and the left wheel speed and right wheel speed are set to 15 cm/s and 30 cm/s,

with an execution duration of 2000 milliseconds. A similar message may be issued from the onboard computer or the ground host computer wirelessly.

3.3.6 Power system

The entire platform is driven by a 6200 mAh 4-Cell (nominal 14.8V, maximum 16.8V, minimum 13.2V) LiPo battery, which has a 120A constant discharge rate and it weighs 490g. In the fully loaded state, the estimated working time is around 40 minutes. A low voltage alarm is attached to the battery and it will buzz when the battery voltage is lower than 14V. A DC-to-DC step-up converter from a mini-box [33] converts battery voltage to 19V to supply the power for the onboard computer and wireless router. 5V is provided by USB ports from the onboard computer or the router for Raspberry Pis and the Arduino 2560. The power of the mechanical drive system is provided directly the by battery via L298P.

3.4 Summary and issues in this proposed robot platform

The proposed indoor robotic platform was built and it met the majority of requirements that were outlined in the beginning of this chapter. For indoor research, the platform is being considered as a key infrastructure. The motor controller with a moving average filter is successfully implemented on an open-source onboard, Arduino 2560, with clean wiring and it is capable of broadcasting real-time wheel speed and taking commands from an Arduino 2560 with a joystick shield as a remote controller. A VMware ESXi hypervisor is installed on the onboard computer in order to run multiple operation systems

simultaneously and in an isolated mode, allowing for a dedicated working environment for each user, snapshots recovery, hardware resource assignments, and hardware pass-through. These essential functions are covered by the VMware ESXi free license, while advanced enterprise functions are disabled and they require a paid license. Examples of these functions include accessing ESXi Application Program Interfaces (APIs) and advanced data protection. The chassis and motor mount are built and assembled as requirements. However, the current metal ball caster wheel is easy to be seized and it shall be replaced by a rubber swivel caster that has lower drag and offers a more stable maneuver.

Two Raspberry Pis, each with one camera module, are acting as an image acquiring system for stereoscopic image research. Occasionally, the two cameras will not be triggered simultaneously from the GPIO pins. The digital and mechanical switches, with various software debounce methods, have both been tested. However, the cause remains unknown. In this thesis, the video acquisition system is not being investigated. However, synchronization issues may appear when the use of multiple sensors is involved such that the sensors and cameras are not able to be triggered at the same time. Additionally, in a dynamic environment, two images are required to be acquired simultaneously. Nevertheless, even in the still image mode, two images are not generated altogether after the triggering command is issued and this may be due to a slight variation in the processing time and low system input or output bandwidth. This problem may be solved by working in low-resolution videos and pictures but further investigation on this issue is required in future work. It is concluded that the Raspberry Pis with camera modules is not recommended for real-time image acquisition and image processing. In the future, this

issue shall be further investigated or the camera rig shall be replaced. Unfortunately, the stereo research camera, BumbleBee®, from PointGray only offers a legacy IEEE FireWire 1394b interface. Using other suppliers or using two machine vision cameras to form a stereo rig may be considered. After all, for this thesis work, the stereovision rig is enough to provide compelling raw image data, assuming that the robot is in a fixed position and the images are taken from a static scene.

Chapter 4. Literature Review on Stereo Matching and Depth Estimation

In the Machine Vision System (MVS), the goal is to imitate and surpass the Human Visual System (HVS) [16]. Stereo vision is a well-known ranging method that consists of examining the location of the same object from two different viewpoints of the same scene along the epipolar line. This emulates the basic mechanism of the human eye and the human perception system. The basic physics and geometry relating to visual disparity and depth estimation have been discussed in Chapter 2. In this chapter, the biological vision system is reviewed, and an overview of automatic stereo matching methologies will be presented, assuming that the image is rectified before stereo matching.

4.1 Principle of binocular vision in humans and animals

The eye and the brain are two essential components of the HVS. In the human eye, light passes a lens that contains the iris, diaphragm and the pupil and the light is then projected onto the sensitive retina. The surface of the human retina is coated with cones and rods in which cones have S, M and L types representing blue, green and red sensors and rods have sensitivity to levels of illumination [16] [34]. The signal is then converted to electric signals and it is sent to the visual cortex for information extraction [16]. Equivalently, in MVS, the CCD/CMOS sensor emulates human eyes as a result of observing the product of an illumination spectrum and reflection spectrum [22]. Additionally, the artificial vision system/algorithm is required to process vision information similar to the human brain. Next, the brain fuses the pictures together into a coherent perception of 3D space, and the limbs

act accordingly. Research shows that binocular vision significantly improves the accuracy of limb movements and the kinematics of human prehension [35]. The study [36] shows that in the human brain, most of the neurons in the visual cortex are sensitive to binocular disparity. However, to generate stereoscopic depth perception, this multi-level process involves both dorsal and ventral streams (visual cortex as a common source) to compute and solve region-based matching and multiple matching problems, respectively. The ability to associate kinetic with pictorial depth sensory information is developed during the early infancy between the age of 5 and 7 months [37]. Similar results can be found in kittens [38]. Even in monocular vision, humans are capable of estimating appropriate depth by instinctively employing "in-born" cues when viewing objects according to size, brightness, color and details of the objects [39][40]. When an observer is sitting on a fast-traveling vehicle, the nearby mailbox and the light post appear to move faster than the distant house. This is an example of parallax and this phenomenon can be further exploited.

4.2 Early era on stereo matching research

In computational vision research, stereo vision is one of the prevailing domains. Each year, new approaches are proposed and this makes it difficult for researchers to stay up-to-date [41]. Before the spread of the digital sensor era, stereo vision research was already focusing on applications relating to interpolating aerial images for cartography, obstacle avoidance and automated guidance for autonomous vehicle control and understanding and modeling human stereo vision [42]. In the 1980s, theories concerning image acquisition, camera modeling, feature acquisition, and evaluation criteria were well established [42]. Both of the feature-based and local area/window based matching methods were explored. Due to

limited computational resources, feature-based methods were more popular [43]. The edge-based matching method requires model-based interpolation to generate a dense disparity map [42]. Nowadays, the majority of matching methods and techniques are further exploited for real-time application and high-resolution depth map generation.

4.3 Feature matching in sparse stereo matching

Traditionally, stereo correspondence algorithms are arranged into two groups, those that produce sparse outputs and those that provide dense results [41]. Feature-based matching relies on matching the correspondence of points, segments or edges between two images, and this consequently results in sparse outputs. In 1982, Baker proposed the Canny edge [44] and intensity-based stereo depth estimation method [45] in order to reduce the computational costs of finding correspondences for automatic aerial surveillance. To avoid the local ambiguous edges, Medioni *et al*. proposed a segment-based stereo matching [46] based on the Nevatia-Babu method. Then, Venkateswar *et al.* [47] proposed a hierarchical feature matching framework in which matching starts hierarchically from the surface to the lines and vertices to the edges. More recent work involves 3D wireframe-based robot localization and uses pairwise geometric histograms that were proposed by Tian [48].

When illumination changes, corner features are less sensitive than edge features. Once a corner has been detected, its relative positions will be nearly consistent with the ground truth. However, corners are less distinctive than edges and more ambiguous matches must be distinguished and eliminated [49]. Thus, corner features are often used in ego-motion

tracking or visual odometry to perform the motion estimation of a camera system to a rigid scenes [43] or camera calibration. In 2004, distinctive Scale-Invariant Features (SIFT) [50] were introduced for robot localization and mapping, and the SIFT descriptors have been used as landmarks for SLAM [51] [52]. Sharma *et al*. [53] improved the SIFT algorithm with the use of the Self-Organizing Map (SOM) to reduce the computational time in stereo image matching for an autonomous vehicle. The computational time is reduced dramatically and because only certain feature points are correlated, only a scattered disparity map is generated.

4.4 Dense stereo matching methods

With the dramatic rise of computational performance lately, research on the dense disparity map has regained its popularity. Therefore, recent efforts towards producing dense disparity maps are being reported more frequently compared to sparse results. Apparently, this issue is more challenging than finding the sparse correspondences, and often, inferring the depth information for texture-less regions involves more guessing work [43].

4.4.1 Dense stereo matching working pipeline and assumptions

Generally, solving stereo matching problems means to find the dense disparity given a rectified image. Scharstein and Szeliski [54] identified and generalized the working pipeline, from the past decade of stereo matching works, into four steps:

1) Matching cost computation
2) Cost aggregation
3) Disparity computation and optimization
4) Disparity refinement

The given images are rectified, which constrains the search space into 1D. Also, for computation stereo matching, some other assumptions have to be made. Firstly, surfaces in the image are Lambertian, for which the surface appearance (the amount of reflected light) does not differ within the two viewpoints. The other critical assumption is that the real world contains piece-wise smooth surfaces, such that the single surface does not jump around [54]. Lastly, the image pair must have radiometric consistency.

For generating a dense disparity map, there are two mainstream techniques, which are known as the local matching method and the global matching method. Gupta and Cho [55] stated that the local algorithms are statistical methods based on finding a correlation, while the global algorithms are subjected to explicit smoothness assumptions that are solved by various optimization techniques. In the local method, the correlation process is to find the matching pixels in the reference image and the matching image by identifying the minimum aggregation costs within a supported window or neighboring pixels. The smoothness assumptions are made implicitly by the aggregating support. On the other hand, in the global method, smoothness assumptions are defined explicitly by the energy function and finding the disparity of each pixel includes applying global optimizing techniques in order to minimize the energy function of the whole image. Although most of the high-quality depth map results, based on Middlebury stereo datasets, are generated by the global

method, studies in [55] and [41] also claimed that the global method is impractical in the real-time system due to its iterative nature. The tradeoff between accuracy and speed is clear.

4.4.2 Dissimilarity measurement, matching cost calculation and Disparity Space Image (DSI)

The first step of calculating the dense disparity map is to determine pixel dissimilarities along the epipolar line. Then, the stereo matching cost calculation is defined as a method to find the parallax values in each pixel between the reference image and matching image [56]. The point matched in the matching image should be distinctly different from its neighboring pixels with no ambiguity. The most widely used matching cost function is area based or support region based to provide rich data, such as Normalized Cross Correlation (NCC) [57], the Sum of Squared Differences (SSD) [57], and the Sum of Absolute Differences (SAD) [58]. Table 4-1 shows the common matching cost computation methods in literature. The window function is defined as Equation 4.1 and the window size is of $2m+1$ by $2m+1$ pixels with x, y located at the center of the window. I_L and I_R are intensity values (grayscale values) in the left and right image respectively.

$$\omega(x,y)=\left\{u,v \mid x-\frac{m}{2}\le u\le x+\frac{m}{2}, y-\frac{m}{2}\le v\le y+\frac{m}{2}\right\} \tag{4.1}$$

The AD, SD, SAD and SSD are easy to implement in the real-time system, and they require minimum computing resources. However, they are still prone to being to noise sensitivity (SD) [59], producing coarse disparity maps (AD) [60], and poor performance near the edge

or texture-less regions (SAD) [61]. The NCC is robust to intensity offsets and changes in contrast, but it will blur the discontinuity regions, and it requires more computational resources [62]. Rank transform (RT) is a local non-parametric transform [63], which computes the number of neighborhood pixels $I(u,v)$ that have a lower intensity than the central pixel $I(x,y)$. The matching cost is then calculated based on the absolute difference of the two ranks. The results are based on the relative ordering of pixel intensities instead of the intensities themselves [56]. Moreover, a variation of RT [63] named Census Transform (CT), encodes the spatial distribution of ranks into a bit string. Matching is then performed by using the Hamming distance [56]. The results in [64] show that the RT and CT are inherently robust to radiometric distortions and occlusions, and they are both suitable for the outdoor environment. This is a result of SAD that only requires simple computation models, as discussed in the survey paper, SAD is a preferred matching cost calculation method in real-time implementation [41].

Table 4-1 Common matching cost computation methods in block-matching[56][65]

Match Metric	Definition
Absolute differences (AD)	$AD(x,y,d)=\left\|I_L(x,y)-I_R(x-d,y)\right\|$ (4.2)
Square differences (SD)	$SD(x,y,d)=\left\|I_L(x,y)-I_R(x-d,y)\right\|^2$ (4.3)
Sum of absolute differences (SAD)	$SAD(x,y,d)=\sum_{(u,v)\in\omega}\left\|I_L(u,v)-I_R(u-d,v)\right\|$ (4.4)
Sum of squared differences (SSD)	$SSD(x,y,d)=\sum_{(u,v)\in\omega}\left\|I_L(u,v)-I_R(u-d,v)\right\|^2$ (4.5)
Normalized cross correlation (NCC)	$NCC(x,y,d)=\frac{\sum_{(u,v)\in\omega} I_L(u,v)\cdot I_R(u-d,v)}{\sqrt{\sum_{(u,v)\in\omega} I_L^2(u,v)\cdot\sum_{(u,v)\in\omega} I_R^2(u-d,v)}}$ (4.6)
Rank transform (RT)	$RT(x,y,d)=\sum_{(u,v)\in\omega}\left\|Rank_L(u,v)-Rank_R(u-d,v)\right\|$ Where: $Rank(u,v)=\sum_{(i,j)(u,v)} L(i,k)$ $L(i,j)=\begin{cases}1: & I(i,j)<I(u,v)\\ 0: & \text{Otherwise}\end{cases}$ (4.7)
Census transform (CT)	$CT(x,y,d)=\sum_{(u,v)\in\omega}\text{Hamming}\left(Census_L(u,v)-Census_R(u-d,v)\right)$ Where: $Census(u,v)=\text{Bitstring}_{(i,j)\in\omega}\left(I(i,j)<I(u,v)\right)$ (4.8)

One disadvantage of the support region method is assuming that all of the pixels in the support region have similar disparity (depth) values. This is not always true for the pixels that are close to the depth discontinuities or edges. Thus, improperly choosing the window size will result in a feeble matching cost and it can lead to poor depth estimation. Some

robust matching functions, such as Birchfield-Tomasi (BT) pixelwise matching function [66] [67] and Mutual Information (MI) [68], can also be found in literature. Birchfield and Tomasi proposed a matching cost mechanism, which is insensitive to the image sampling. With this mechanism, each pixel in the reference image is compared to the linearly interpolated intensity functions around the pixel that are in the matching image, instead of comparing the pixel values shifted by integral amounts. MI is a measure for image alignment based on the amount of uncertainty in a probability density function or entropy. Matching cost is calculated by matching unoccluded pixels at a maximum of the sum of two image entropies and their joint entropy.

As displayed in Figure 4-1, the concept of Disparity Space Image/Volume (DSI) is introduced for an easier illustration in Chapter 5. The matching costs (dissimilarities) of a reference image are stored in the 3D space (x, y, d), such that, each element on DSI (x, y, d) projects the pixel (x, y) in the reference image (left) and to ($x, y+d$) in the matching image (right) [69]. DSI represents the confidence or like-hood of a particular match [43]. In the top image of Figure 4-1, the black boxes indicate that these regions in the reference image are highly like to match the matching image at certain disparity levels d. After aggregation on each slide along $D(d)$, the purple boxes represent the minimum cost (dissimilarity) along the disparity space $D(d)$. The line connecting the green boxes is the simple disparity map in 1D.

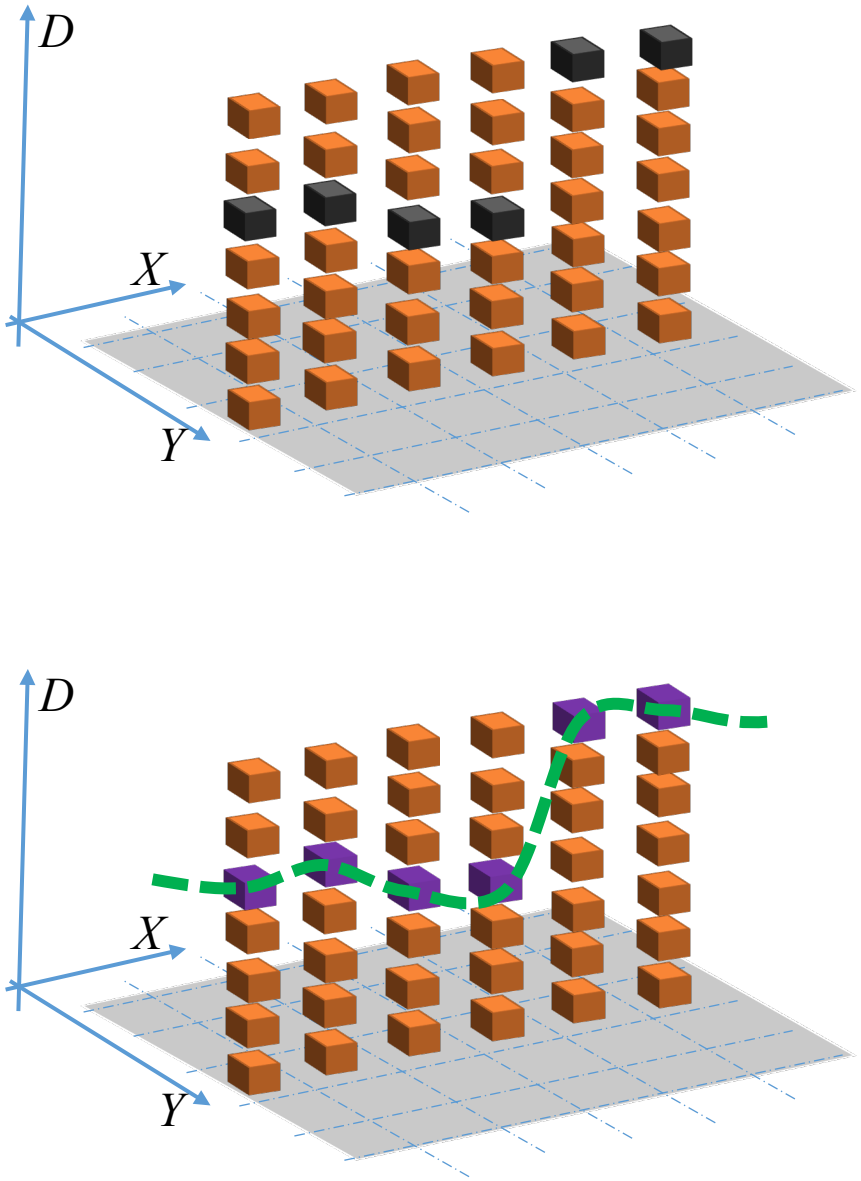


Figure 4-1 Disparity Space Image (DSI) (top), after aggregation, the minimum cost state is expressed in 1D as green boxes (bottom), the 1D disparity is along the dashed line.

4.4.3 Local window-based matching method

Traditionally, in local window-based matching, for a given pixel, a key step in calculating the disparity is subject to finding the sum of all intensity values within a finite support region in which the pixel is located at the center of the interested region [43]. Therefore the equations in Table 4-1 already consist of the both matching cost and cost aggregation steps. For example, in the SAD method, the matching cost is to find the difference of intensity values at the given disparity level, and the DSI is generated. Then, summing the differences over the window within each slide (along with *D-axis*) of DSI is the aggregation process. Even if the DSI is noisy and ambiguous, it still provides useful information. Disparity selection is then performed afterwards. The simpler algorithm makes use of Winner-Takes-All (WTA) strategy by locating the constant value of disparity level *d* that is corresponding to the minimum matching cost. As seen in the bottom of Figure 4-1, finding the disparity level $d(x, y)$ for a pixel (*x,y*) is done by locating aggregated minimum dissimilarity or cost along *D-axis* as Equation 4.9.

$$d(x, y) = \arg \min_{d \in D} \text{AggregatedMatchingCost}(x, y, d) \tag{4.9}$$

As discussed earlier, in the Fixed-size Window aggregation (FW) or Constant window Aggregation (CW) technique, the disparity is assumed to be constant within a finite support window, and this results in biased results towards the frontal-parallel surface [70] and blur across depth discontinuities. Moreover, it may exhibit erroneous correspondences in texture-less regions and occluded regions. Furthermore, it means that the smaller window yields good precision with the finer details, but it is sensitive to noise. However, when using a larger window size, it is more robust to noise with the tradeoff of less precision.

Hence, defining the window size for one or multiple scenes is a non-trivial problem [43], and this will affect the outcome of matching. When using DSI at the aggregation stage, various constraints can be added to improve the distinctiveness of the matching costs, such as using different convolution kernels (window sizes) in 2D $\omega(x, y)$ or 3 D $\omega(x, y, d)$ [43] as Equation 4.10 illustrated, and $C_0(x, y, d)$ is representing the DSI.

$$C(x, y, d) = \omega(x, y, d) * C_0(x, y, d) \tag{4.10}$$

Muhlmann *et al*. [71] proposed a method that is based on the SAD method in RGB color images and it achieved reasonable quality and speed. Kanade *et al.* [58] presented an Adaptive Window (AW) size method. Here, the appropriate window size is adaptively selected by evaluating the local variation of intensity and the disparity by using a statistical model, which was developed to represent the uncertainty of the disparity of points over a window. Then this method was further improved by Veksler [72], Lu *et al*. [73], and they achieved high-quality results near depth discontinuities and inside the homogeneous regions. Yoon *et al*. [74] described an Adaptive Support Weights (ASW) method by adding weights to pixels that match the colors of the center of the kernel, assuming that the same colors are likely to be at the same disparity. Gu *et al*. [75] further explored the AWS method with a customized RT, and the object boundaries were well preserved. Tombari *et al*. [76] introduced a segmentation-based aggregation method with similar outcomes in terms of accuracy. Moreover, Tombari *et al*.[77] then assessed the most popular support-based strategies, and the results show that the FW method is the fastest one with moderate accuracy. Segment support yielded the highest accuracy and it took quite a long time to obtain the results.

4.4.4 Global matching method

The essential difference between the global and local method is that in the global method, the disparity calculation is accomplished by performing the optimization or iteration steps after the matching cost computation, and often, the aggregation step is skipped. The objective is to find the disparity *d* that minimizes the global energy function in Equation 4.11 that can be expressed by summing the matching cost (DSI) and smoothness cost as Equation 4.12 and Equation 4.13, respectively. A weighting factor for the smooth term is defined as β. The smoothness term evaluates if the neighboring pixels have similar disparity.

$$E(d) = E_{match}(d) + \beta E_{smooth}(d) \quad [43] \tag{4.11}$$

$$E_{match}(d) = \sum_{(x,y)\in \text{image space}} \left\| I_L(x,y) - I_R(x-d,y) \right\| \tag{4.12}$$

$$E_{smooth}(d) = \sum_{\text{neighbor pixles } i,j} \left| d_{disp,i} - d_{disp,j} \right| \tag{4.13}$$

This question can be formulated as a Markov Random Fields (MRFs) problem with a Maximum a Posteriori (MAP) estimator [78] as shown in Equation 4.14, where the Ψ function represents the compatibility between neighboring pixels i, j . Y expresses observations or given stereo image set and x_N is a set of pixels in the disparity map with a disparity value. Additionally, Φ is a joint compatibility function of a node x_i when given observation *Y*. Taking the logarithm of both of the sides on Equation 4.14, and Equation 4.15 is obtained. Thus, finding the MAP becomes minimizing Equation 4.15. However, MRF-MAP is a NP-hard problem generally [79]. Graph Cuts (GC) and Loopy Belief

Propagation (LBP) are both labeling algorithm and they are trying to iteratively approximate the optimal solution to the problem.

$$P\left(x_1, x_2, ...x_N \mid Y\right) = \prod_p \Phi\left(x_i, Y\right) \prod_{(i,j)} \Psi\left(x_i, x_j\right) \text{ [80]} \tag{4.14}$$

$$E\left(x_1, x_2, ...x_N \mid Y\right) = \sum_p D\left(x_i, Y\right) + \sum_{(i,j)} V\left(x_i, x_j\right) \text{[81]} \tag{4.15}$$

LBP is an iterative process that computes messages at each node (pixel) and sends them to connected neighbors. Eventually, the message values at each node will converge, and the message values are used to estimate the state of node (costs) [82]. Further details about the message formation can be found [82] [83]. GC can also be used to find the global minimum in Equation 4.16. Finding stereo correspondences can be modeled as a pixel labeling problem where *p* represents a set of pixels. The objective is to find a labeling *f*, which minimizes the energy function.

$$E\left(f\right) = \sum_{p \in \text{Ref}} D_p\left(f_p\right) \; + \sum_{i,j \in neighbours} V_{i,j}\left(f_i, f_j\right) \text{ [79]} \tag{4.16}$$

where $\sum_{p \in P} D_p\left(f_p\right)$ is an energy function and it measures how relevant a label is assigned to the pixel *p* given the observations *Y*. In fact, this corresponds to the match cost or DSI. The energy term $V_{i,j}\left(f_i, f_j\right)$ encodes the prior or smoothness constraint. Two influential pieces of literature, using GC in stereo matching, are published by Roy *et al*. [84] and Boykov *et al.* [81]. Tappen *et al*. made a comparison between GC and LBP [83] and they concluded that the two methods are comparable.

Zitnick *et al*. [85] and Bleyer *et al*. [86] proposed a method to label each region with a disparity in the segmented image. More recently, Zhang [87] proposed a mesh stereo method to produce a better visual effect compared to the traditional point-cloud-based methods in 3D scene reconstruction. The input images are partitioned into 2D triangles with shared vertices, and a two-layer MRF is then applied on both 2D triangle mesh and input images while optimizing region-based stereo matching in the input images. Those methods produce high-quality depth prospection after 3D reconstruction, and they are designed for image-based rendering.

Dynamic program (DP) or dynamic optimization is another technique that is used in the global method. This method works by finding the minimum cost path through the DSI between the reference image and matching image scanlines [43]. DP is a fair tradeoff between the quality of results and computational complexity. This approach was later improved by Hirschmuller [88] who proposed the Semi-Global Matching (SGM) method, which is an efficient approximation for optimizing an MRF [89] [43]. As shown in Figure 4-2, in the SGM method, a cumulative function is constructed, and it scans eight or sixteen cardinal directions.

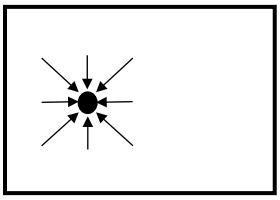


Figure 4-2 Cost aggregation from 8 directions in SGM

Some other novel methods try to address global stereo issues, such as data-driven matching cost by using Convolutional Neural Networks (CNN) [90] [91] and iteratively updating disparity estimation by cooperative algorithms [69]

4.4.5 Depth refinement

The disparity estimates are computed in the discretized space (integer pixel values). Depending on the applications and the accuracy requirements, the disparity map may need to be refined in order to reduce noise and smoothen edges. For robotic applications, the initial discrete results can be sufficient, but the quantized maps may lead to unappealing synthesized results for image-based rendering [43]. A common method to improve disparity map resolution is to fit a quadratic curve to the matching costs at discrete levels [92]. In addition, occluded areas may be detected by a left-right-left consistency check and filled with disparities similar to the background [93]. Lastly, for local and regional refinement, a medium filter and Gaussian filter may be used to remove small and isolated mismatches. Moreover, the medium filter is a better choice since it preserves edges and it is more suitable for real-time processing [94].

4.5 Distance determination

Once the perfect conjugate pixel pairs have been found in two images, the parallax or disparity can then be calculated along the epipolar line. The depth or distance determination from a voxel in the space to the reference image plane can be easily determined by simple triangulation, which is outlined in Section 2.3.3: Basic stereo normal, triangulation and depth estimation.

4.6 Stereo benchmark datasets

To evaluate the performance qualitatively and equally (speed and error rate) of the stereo matching algorithms, there are two well-known datasets for the field of stereo matching research. The Middlebury stereo dataset contains high-quality image that were captured in the controlled indoor environment, and the ground truth was measured by carefully constructed structured light [95] [96]. This dataset also provides three measurement metrics, which include all pixels (all), non-occluded pixels (nonocc), and pixels near discontinuities (disc). On the other hand, the Karlsruhe Institute of Technology (KITTI) vision benchmark suite [97] provides grayscale/color stereo images, and video streams with ground truth measurements and an absolute location from the laser scanner and GPS while all of the data was recorded from a moving manned vehicle in the outdoor environment. On both institutions' websites, newly proposed matching algorithm and benchmark results can be found. However, it is unrealistic to simulate each scene and light condition in the real world, and this is an inherent limitation in all of these datasets. In the next chapter, selected stereo sets from the Middlebury dataset with ground truth will be used to examine various stereo matching algorithms.

4.7 Challenges of using the stereoscopic camera as a range estimation device in the outdoor, indoor environment and general applications

An advantage of stereoscopic triangulation is that it is capable of capturing two images and reconstructing the 3D scenes at once. It is superior to the point and measure type 1D sensors, *i.e.*, ultrasonic sensors and IR sensors. There are some challenges associated with all of the cameras in the real world environment, especially while working outdoors. The first important issue is light condition. Since the camera sensor is a passive sensor, it requires sufficient illumination and photons to activate the sensor to capture images. From the object to image sensing stage, the photometric variations and radiometry variations constantly exist, such as sudden illumination variations and shield sunshine by clouds. Due to the sensor and lens quality, sensor noise, defocusing, chromatic aberrations and perspective distortions also contribute to erroneous measurements. In the artificial or natural environments, special senses and structures, such as reflections and transparency, specular surface glare, occlusions, and shadows, will also result in invalid measurements. Some local feature matching algorithms may be confound by the surfaces that appear to be texture-less and ambiguous, and contains repetitive structures or strong discontinuities. These are the key reasons for why stereo vision is still considered to be an unsolved problem in computer vision [85].

Also, because minimizing the global energy function is a NP-hard problem, the computational cost of stereo images is more expensive as there are an increasing number of pixels that are contained in the image, and this requires more calculations to find the possible correspondences, especially in real-time computation. A simple solution is to use

hardware acceleration to process the image data chunk by chunk, since any images can be partitioned as pixels, lines, and regions. Moreover, the matching algorithms are often required to process the data in parallel in order to speed up the calculation. The algorithm is always designed and evaluated in software for CPUs first, and it is then later implemented in specifically designed hardware [41] [98]. The modern approach to toggle this problem is to use parallel computing in the Graphics Processing Unit (GPU) or GPU accelerator boards, like the Nvidia® Tesla and its associated specialized parallel programming platform Compute Unified Device Architecture (CUDA). The power consumption and size of such a system are both large, and it requires a large platform to carry such a manned vehicle. The other hardware approach is using FPGA due to its inherent parallel nature. National Instruments®, Xilinx®, and Altera® already have industrial machine vision products available. Survey papers [98] [41] [99] show that most stereo matching algorithms will significantly outperform in GPUs and FPGAs than CPUs due to recent advances in hardware technology in GPU and FPGA. In the embedded stereo vision system, FPGA with additional external memory [100] is a better choice due to its low power consumption [101] and compact footprint. Some literature on GPU and FPGA implementations can be found [98][102]. It should be noted that the processing speed and efficiency of each algorithm implementation are both difficult to compare in the literature. This is because some algorithms are not fully optimized (including different parameters used) and different hardware architectures are used. Similar findings are reported in [41] [98].

4.8 Camera calibration and epipolar geometry

In the scotopic vision, precise camera calibration is required. In this thesis, the main purpose of camera calibration is to obtain rectified images since epipolar geometry is the fundamental constraint in stereo vision. Also, the stereo camera rig may not have perfect alignment.

4.8.1 Single camera calibration

Camera calibration involves the calculation of extrinsic and intrinsic parameters for perspective transformation from world coordinates to pixel coordinates. The extrinsic parameter specifies the camera pose in the word, which includes rotation and translation parameters. On the other hand, intrinsic parameters map the 3D scene in camera coordinates to the final image in pixel coordinates, and these parameters consist of focal length and optical center information [22]. The calibration has to account for lens distortions (intrinsic group). Commonly used models are radial distortions and tangential distortions. These two sets of parameters may be estimated by maximum like-hood estimation with given checkerboard images (control points) [103]. In most of the applications, the camera requires to be calibrated only once. However, in some applications such as estimating poses of a helmet mounted camera in Augmented Reality (AR) [104] [105], online calibration is required.

4.8.2 Stereo camera calibration and epipolar geometry

Figure 4-3 illustrates the basic epipolar geometry, such that a point X in space is projected to left image plane x_L and any point on the ray from left image optical center will also be projected to x_L. Then, this line viewed from the right image plane will generate the epipolar constraint in the right image along $x_R o_R$. Similarly, any point in the right image also has its corresponding epipolar line in the left image [106]. A matrix must be used to express the relationship between two matching points. The fundamental matrix describes such a relation for uncalibrated cameras, and it can be estimated by using a coplanarity constraint on $Xo_L, Xo_R, o_L o_R$ and eight point algorithm (from control points, or checkerboard pattern) [106]. Moreover, the essential matrix is used in calibrated camera images. The coplanarity constraint can be simplified, and only the translation and rotation between two camera coordinates must be considered [13]. The rectified image will have all of the epipolar lines ($x_L e_L, x_R e_R$) in parallel.

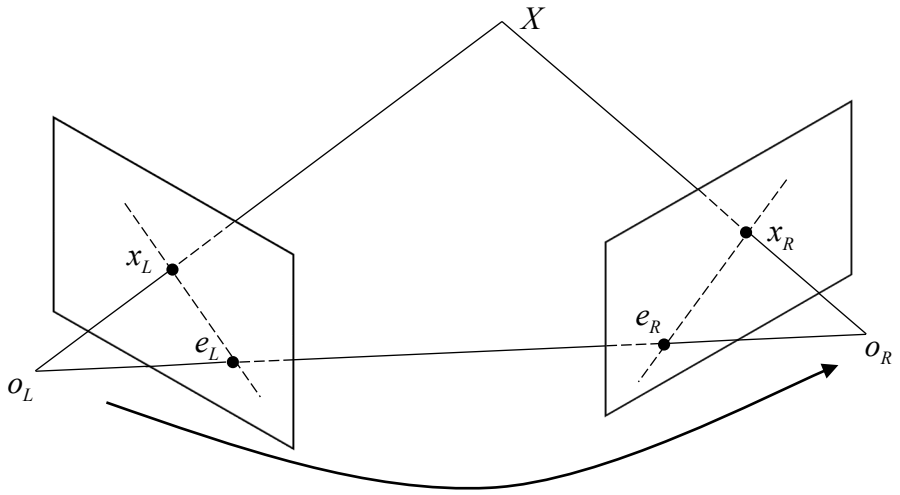


Figure 4-3 Stereo and epipolar geometry

Chapter 5. Stereo Matching for Depth Estimation in the Indoor Environment

As discussed in the previous chapters, stereo vision is the operation of recreating ranging or depth information using a pair of images of the same scene. The objective of this technique is to recreate 3D scenes in image rendering, or for the robot to perceive its surrounding environments. In this chapter, two sparse matching methods and two dense matching methods are investigated[3], and human-readable coarse 3D scene renderings and binary maps for the robot are also presented. The target measurement range is defined as 0.5 m to 4 m for domestic robots.

5.1 Introduction

Figure 5-1 shows a workflow from image acquisition to 3D scenes reconstruction for a depth map. Lens distortion removal and image rectification are done using the Matlab Computer Vision Toolbox™. This chapter focuses on stereo matching and 2D/3D maps reconstructions. Two dense methods, the SAD window-based Block Matching (BM) method and the Semi-Global Matching (SGM) method, will be examined in the indoor robot application. On the other hand, the feasibility of the two sparse depth map generation methods, based on corner features and SIFT features, will also be assessed.

[3] Code is available on https://github.com/kunzhuang/vacuum

Contrary to commonly used Digital Single-Lens Reflex Cameras (DSLRs) or research grade camera rigs in literature, the stereo camera rig that is employed in this thesis work only consists of two Raspberry Pi camera modules (SONY 8MP sensors [21] with pin-hole lens) including IR (with an IR filter) and NonIR (without an IR filter) sets. These sensors, with pinhole lens, are prone to radiometric errors and individual variations. However, they are inexpensive and commonly seen in tablets or portable electronics and this means that the image quality from such sensors would be closer to the actual image quality delivered to customers in photography or security surveillance applications. Consequently, it's interesting to evaluate the quality of 3D information that is inferred from these images that capatured from this sensor. The Middlebury stereo datasets will be used in the standard test method as benchmarks since it contains ground truth, while the images captured by Raspberry Pi camera modules are used to examine how well the algorithm works in real world scenarios.

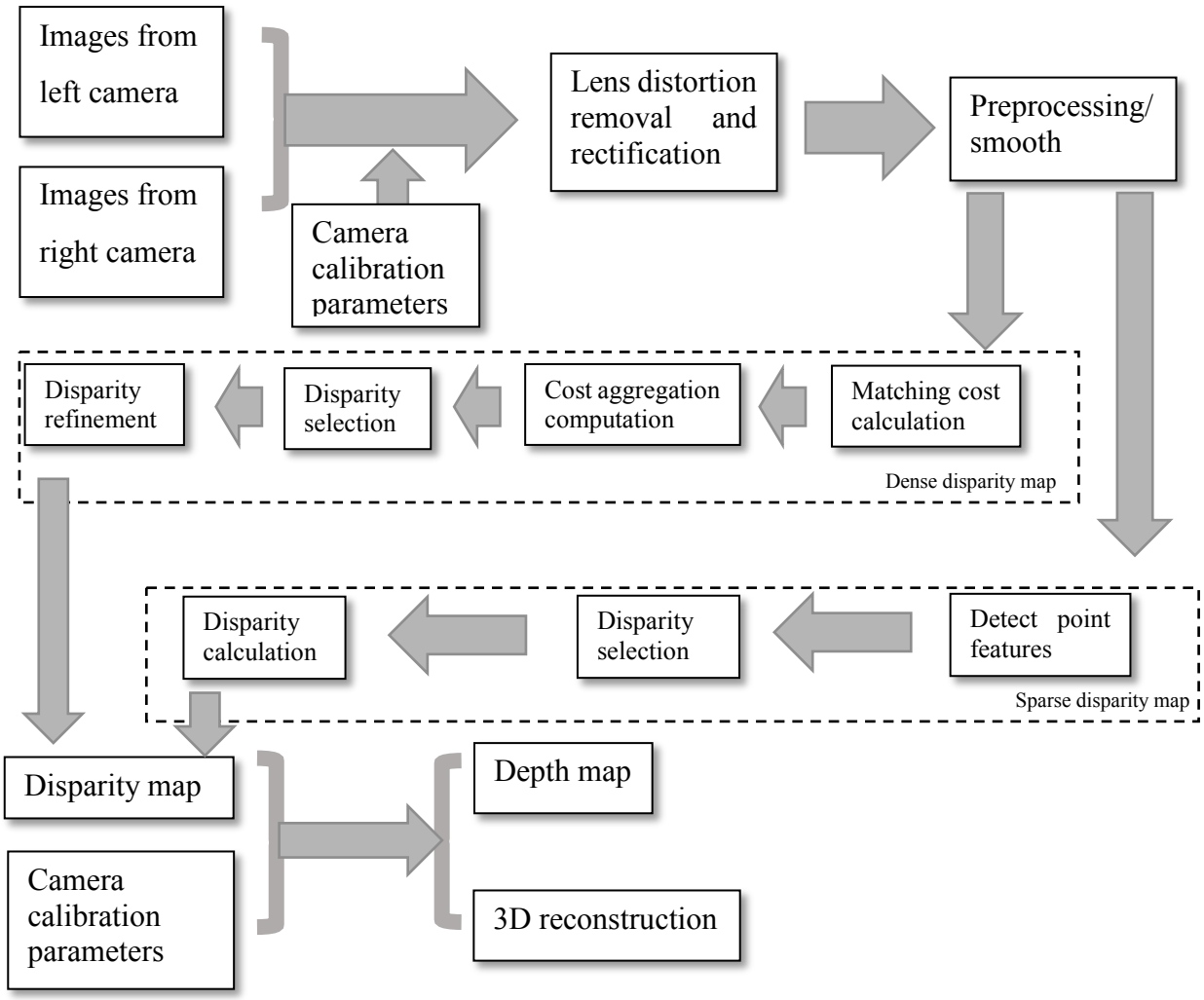


Figure 5-1 Working pipelines from image acquisition to depth map generation

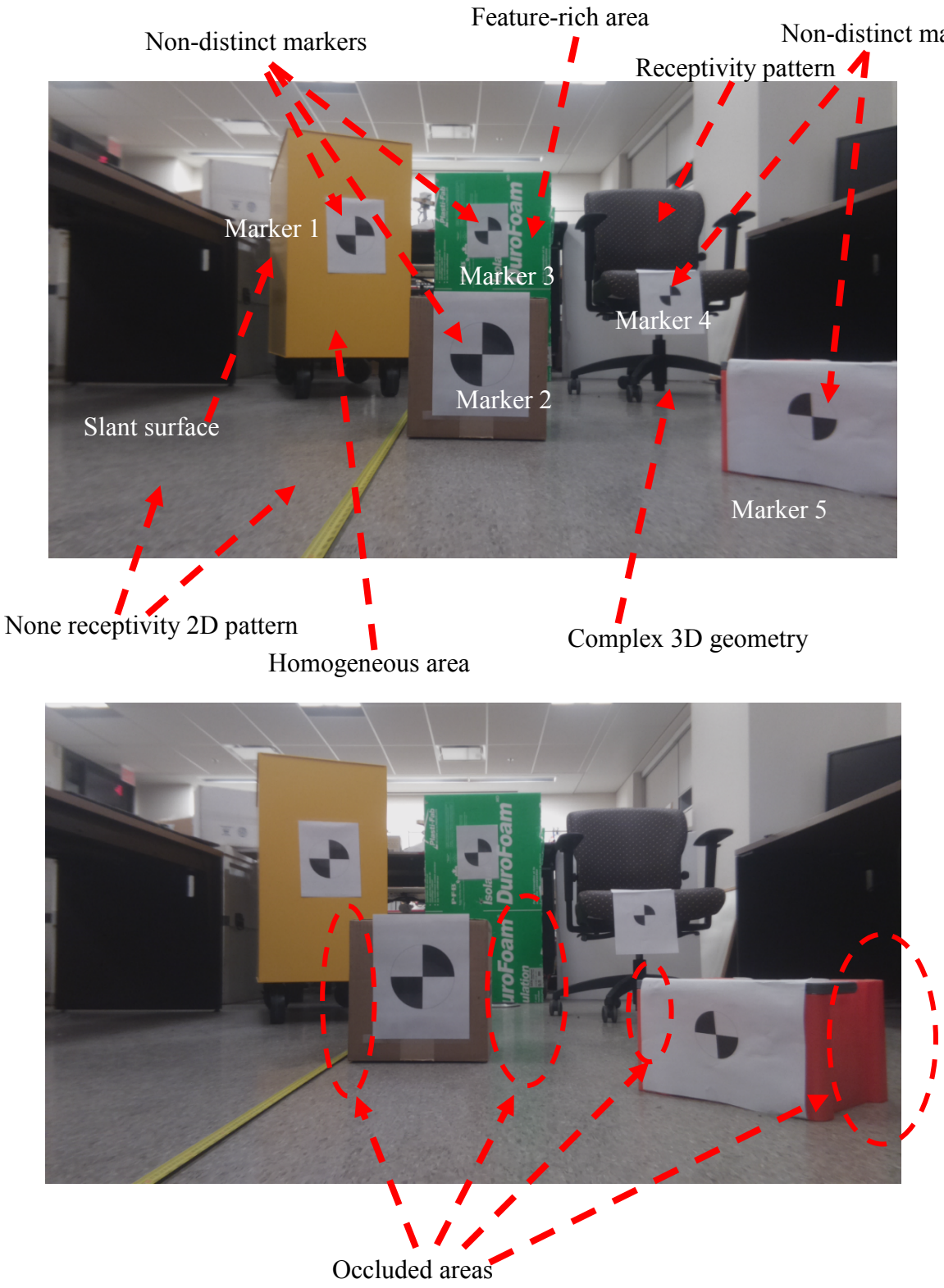


Figure 5-2 The constructed scenes in the laboratory, image captured by left camera as a reference image (top), image captured by right camera as a matching image (bottom)

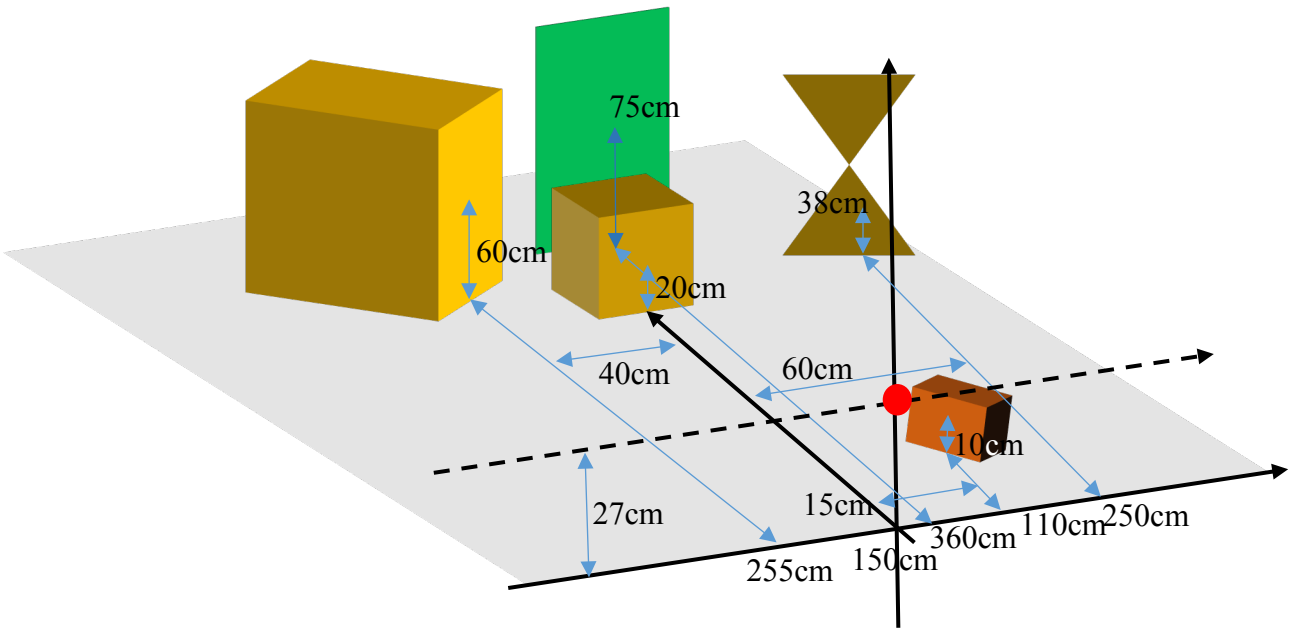


Figure 5-3 Relative distance measurements between markers in lab images

Other than the Middlebury stereo image datasets that are used as standard test sets, a scene is created in the laboratory to simulate the domestic environment. As shown in Figure 5-2, this human-made scenario contains challenging conditions, such as homogenous (color) surfaces and a large slant surface on the trolley, five non-distinct markers, respective pattern surfaces on the chair, complex 3D geometry objects, scattered patterns on the ground and four large, occluded areas. In the image, only one distinct feature-rich area on the green insulation foam is presented. The 3D laser scanner is not used in this work. Thus,

only the relative distance of markers is available, and the relative positions are explained as seen in Figure 5-3. The red dot in the figure represents the location of the left camera or the reference camera in the space. The origin of world coordinate coincides with the center of the left camera.

5.2 Image calibration and rectification using Matlab

Stereo camera sets are rare in the ideal capture setting, such that, misalignments dramatically affect the matching results and the quality of the disparity map. Additionally, epipolar geometry is the fundamental constraint in the stereo matching problem and rectification aligns epipolar lines with a scanline for reliably 1D search. A pair of Raspberry Pi camera modules, that are apart by a baseline of 20 cm, are used to capture images, and the lens distortion calibration and image rectification are conducted automatically by the Computer Vision Toolbox in Matlab [107]. This is shown in Figure 5-4. During the lab image capturing and camera calibration, the two cameras are orientated at almost stereo normal condition. The focus of each camera module is manually adjusted to 75 cm. Intrinsic parameters including two radial distortion coefficients for each camera module and essential matrix are computed from a calibration checkerboard that has known dimensions. All of the mean reprojection errors are less than one pixel.

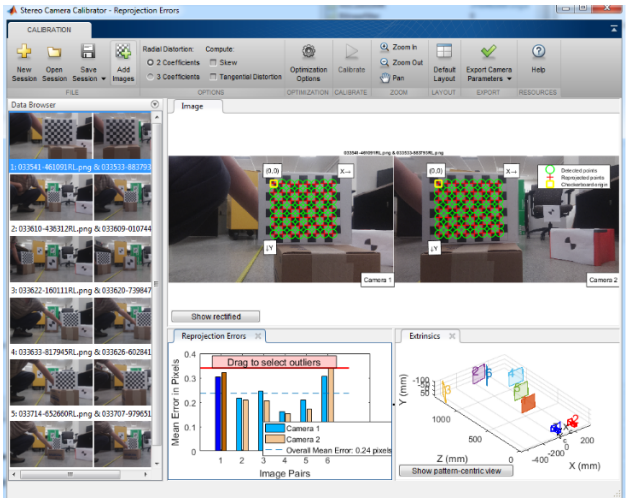


Figure 5-4 Stereo camera calibration in Matlab

Table 5-1 Key camera parameters estimated from Matlab Vision Toolbox

NonIR (no IR filter)	Cam 1	H focal	V focal	HC Pix.	VC Pix.	RD coeff.
	FR	2491.5	2483.7	1665.4	1239.7	[0.2146, -0.3698]
	HR	1331.7	1331.5	839.3	465.3	[0.2361, -0.4458]
	Ratio	1.87	1.87	1.98	2.66	
	Cam 2					
	FR	2466.9	2468.1	1700.4	1243.8	[0.1955, -0.1922]
	HR	1330.7	1331.7	862.3	463.8	[0.2115, -0.3130]
	Ratio	1.85	1.85	1.97	2.68	
IR (IR filtered)	Cam 3	H focal	V focal	HC Pix	VC Pix	RD coeff.
	FR	2619.0	2613.4	1641.4	1236.0	[0.1834, -0.3149]
	HR	1278.6	1277.4	801.6	482.3	[0.1057, 0.0345]
	Ratio	2.05	2.05	2.05	2.56	
	Cam 4					
	FR	2624.4	2620.4	1620.4	1233.0	[0.1704, -0.2955]
	HR	1274.2	1271.7	811.6	472.5	[0.1252, 0.0645]
	Ratio	2.06	2.06	2.00	2.61	

Notations:
IR: Raspberry camera module without IR filter;
NonIR: Raspberry Pi camera module with IR filter;
FR: Resolution at 3280x2464, 4:3 full FOV;
HR: Resolution at 1640x922, 16:9 full FOV with two by two binning and cropped;
H focal: Horizontal focal length in pixels;
V focal: Vertical focal length in pixels;
HC Pix: Horizontal center pixel location (horizontal principle pixel);
VC Pix: Vertical center pixel location (vertical principle pixel);
RD coeff.: Radial distortion coefficient.

As shown in Table 5-1, two sets of cameras are used to capture images in this work, the Raspberry Pi camera modules without an IR filter (Cam1 and Cam 2) and camera modules with an IR filter (Cam 3 and Cam 4). In each camera set, two sets of images are captured that are Full Resolution (FR, 3280x2464, 4:3 full field of view (FOV) 62.2 x 48.8 degrees [108] and Half Resolution (HR, 1640x922, 16:9 full FOV with 2 by 2 binning and cropped). The maximum FOV was used to preserve the maximum view angle. The NonIR cameras usually work in indoor environments or at night where illumination is limited. The cameras with an IR filter filter out IR spectrums and provide images that are more colorful.

For depth estimations, the most important parameter is the focal length in pixels. Comparing the focal length for FR and HR in Cam 3 and Cam 4, the ratio is 2.06, that is consistent with the theoretical value of 2. Nevertheless, the focal length ratio of FR and HR is 1.8 in Cam 1 and Cam 2, which is about 7.5% lower than the ideal value. On the other hand, the theoretical focal length [21] [108] given by the manufacturer in pixel is:

$$\frac{focal\ length}{CMOS\ unit\ cell\ size} = \frac{3mm}{1.12\mu m} = 2678.6$$

With binning of 2 by 2 pixels:

$$\frac{focal\ length}{CMOS\ unit\ cell\ size} = \frac{3mm}{2 \cdot 1.12\mu m} = 1339.3$$

The theoretical focal lengths in pixels in two cases approximately matches the NonIR camera in the HR case and IR filtered camera in the FR case. The parameters listed in Table 5-1 provide accurate depth estimations in the following calculations. However, the reason

why the estimated focal length does not match the ideal focal length, and the reason why the focal length ratio of FR and HR is off by 7.5% in cameras without IR filter are both still unknown.

Due to computing resource limitations, only HR image sets are used in the depth estimation tests and validation. The reason for this will be explained later. Hence, for images without an IR filter, the resolution after image rectification is 1506×818, and the rectified image resolution is 1615×863 for images with an IR filter. Additionally, in stereo matching, the disparity levels have to be identified by using Equation 2.5, *i.e.*, for the desired measurement range of 0.5 m to 4 m, the corresponding disparity level is from 512 to 64, which is the only disparity range used in the lab images. The disparity range for Middlebury stereo datasets is normalized to 256 to 1.

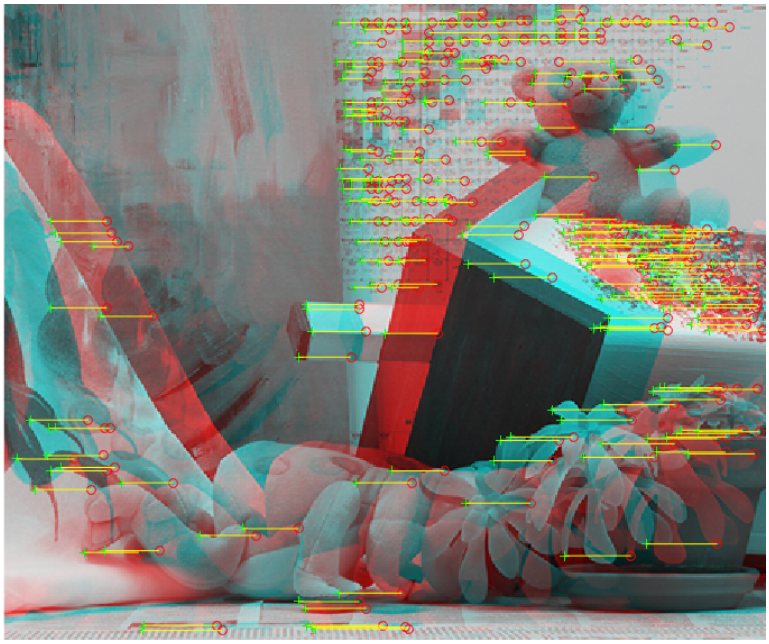
5.3 Feature-based stereo matching method

As discussed earlier, corner features may be used to construe sparse depth maps. It is still worthwhile to investigate the feasibility of building sparse depth maps in the real world scenario. The Harris corner [109] method is a robust corner descriptor. The working principle is to look at intensity values within a small window. If there is a junction of contours in the window, shifting the windows in any direction should yield a large change in appearance. Mathematically, it is done by finding two eigenvalues of structure tensor of a pixel M computed from the derivatives about a point. If the threshold ($R = \det M - k(trace\ M)^2$, k –empirical constant between 0.04 to 0.06) is larger than 0,

this point is a corner [43]. This threshold is set to 0.01 to find the corners in the two test images. Once the corner features are located, Normalized Cross Correlation (NCC) is employed to search for the correspondences along the scanline with the window size 5 by 5 in the Middlebury stereo image Teddy and a 25 by 25 window on the lab image. The matching results can be seen in Figure 5-5, where the green markers represents point features and the red markers are the matching features.

SIFT is a feature detector and descriptor for the invariant and distinctive features. It finds and chooses key points from local extrema, and it then creates gradient directions by an orientation histogram [50]. The evaluation is conducted by applying the SIFT detector, descriptor and matcher from VLFeat [110] on Middlebury image sets and lab images. Figure 5-6 illustrates the matching results and Figure 5-7 shows the 3D sparse reconstruction from the matches.

The point features are insensitive to texture-less areas, which can be observed in Table 5-2, only certain key features are matched, which results in a limited density of points for which there may be some key features in the scene. Thus, though the computational time is reduced by only obtaining correlated point features, this technique is not preferred. The sparse stereo methods provide insufficient map information. In this work, there will be not be any further investigations on these methods. However, these methods can be used in visual odometry, feature tracking (no image rectification required) and feature-based image stitching. These areas are not covered in the scope of this thesis work.



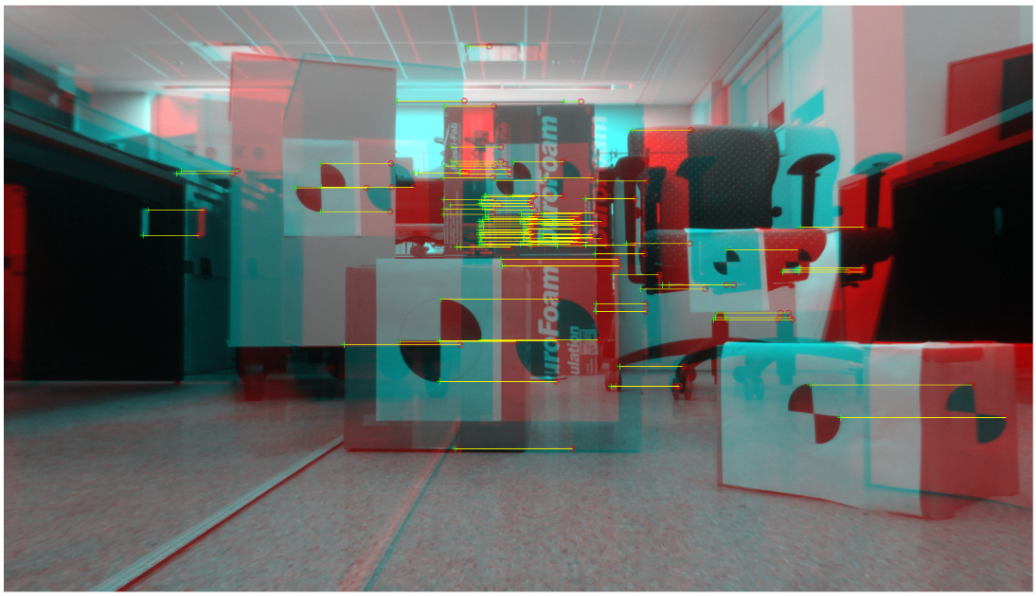


Figure 5-5 Harris corner matching in the Teddy (top) and lab image (bottom)

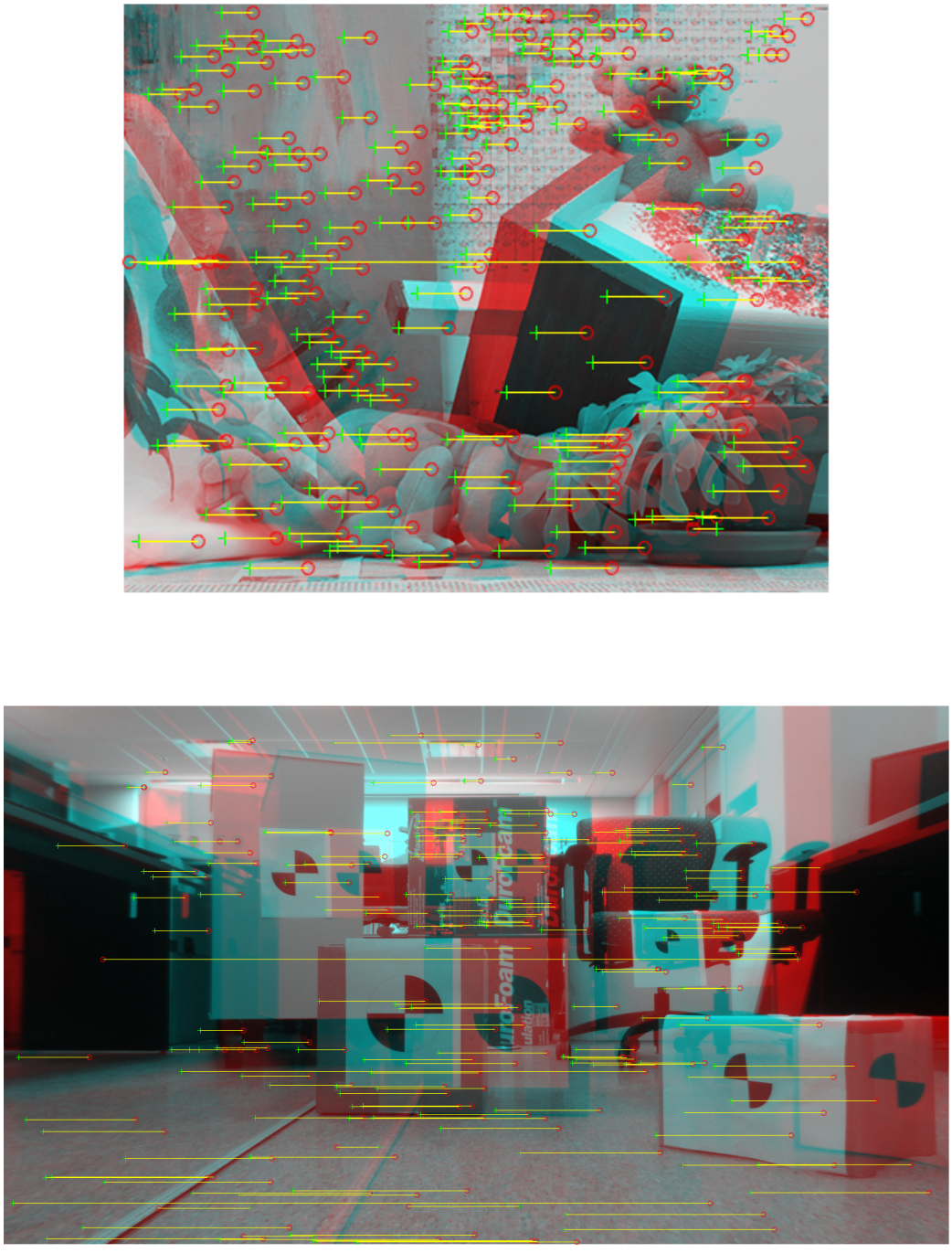


Figure 5-6 SIFT matching in the Teddy (top) and lab image (bottom)

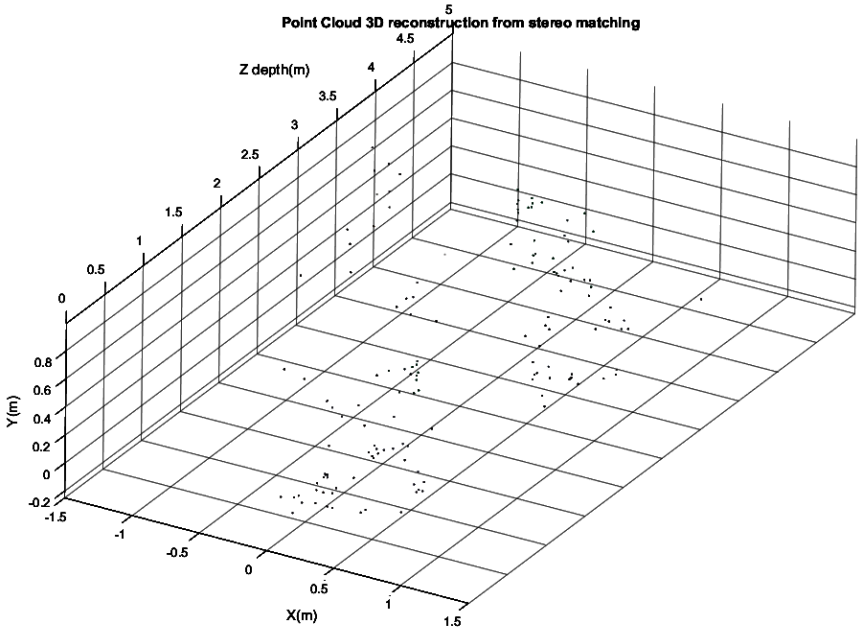


Figure 5-7 3D sparse reconstruction of the lab image from SIFT

Table 5-2 Number of matching points by Harris corner feature and SIFT

	Harris (pixels)	% of image pixels	SIFT (pixels)	% of image pixels
Teddy	279	0.1683%	247	0.1490%
Lab	123	0.0094%	223	0.0170%

5.4 Dense stereo matching method

The following sections investigate dense disparity map constructions and 3D triangulations. The local window-based Block Matching (BM) method and Semi-Global Matching (SGM) method are both tested and evaluated on Middlebury stereo datasets and the lab images.

5.4.1 Left-right consistency check (L/R check) mask and evaluation metrics

In addition to the workflow outlined in Figure 5-1, one crucial step is to eliminate or mark the unreliable matching pixels in the refinement stage. These mismatch points may be the result of occlusions from the images or radiometric effects during image capturing. Here a binary mask is proposed. The mask checks left-right-left consistency and enforces ordering constraint. Each image shall have its own disparity map calculated, *i.e.*, use the left image as a reference image and the right image as a matching image to generate a disparity about the left image and vice versa. As Figure 5-8 illustrates, once the disparity maps for the left and right images are obtained, the strategy is to check the disparity value d_R at location of $x_L - d_L$ (matches x_L on the left image) in the right image.

$$\left(x_L - d_L\right) + d_R \le x_L + threshold \tag{5.1}$$

If Equation 5.1 is true, then this pixel x_L is reliable with the disparity d_L in left image. In robotic applications, a high resolutions depth map is not required. Thus, a binary mask is created to label the unreliable pixels. This mask will create vacancies in the depth map, but it increases the possibility of finding the correct correspondences in the disparity map and reducing the noise in the depth map.

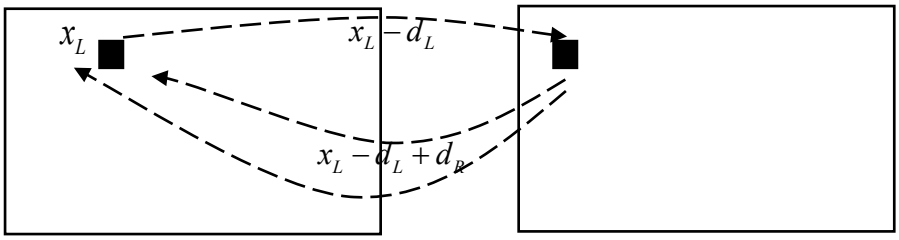


Figure 5-8 Processing steps for dense disparity matching with consistency check

Besides using visual inspection to evaluate the correctness and accuracy of the depth maps and 3D reconstructions, in the standard benchmark test, Middlebury stereo datasets suggest that the matching methods should be evaluated on "all", "non-occlusion" and "non-discontinuities" regions.

In this chapter, we consider the "all" condition, which compares the estimated disparity map directly to the ground truth. Moreover, the second scenario is to apply the L/R check mask to both of the calculated disparity maps and ground truth, and then compare the results. Three evaluation metrics that are applied to each scenario for comparison of the estimated results to ground truth are proposed here:

- Percentage of the valid pixels left after applying L/R mask in reference image
- Root mean square error (RMSE) as Equation 5.2 that offers a precise single value, which compares the calculated disparity map versus the ground truth and it is used to rank the quality of results of different stereo vision algorithms.

$$RMSE = \sqrt{\frac{\sum_i \sum_j \left(d(i,j) - \hat{d}(i,j) \right)^2}{n}} \tag{5.2}$$

- The percentage of pixels that have been found to be the correct match within a threshold range

However, in the lab image datasets, the ground truth is not given. Thus, only the distances from the markers to the reference camera are evaluated.

5.4.2 Local window-based Block Matching (BM) method

The window-based Block Matching (BM) method tries to estimate disparity at a point in the reference image by comparing a small support region about that point with a series of small regions extracted from the matching image. Sum of Absolute intensity Difference (SAD) is applied for evaluation of the local window-based BM matching method due to its calculation simplicity.

As shown in Figure 5-9 (a), in the conventional window-based BM method, all of the pixel values are first summed up within a support window on both images. Then, the total value is assigned to the pixel at the center of the window, which characters the support region about the center pixel using a numerical value. A scan is performed, such that, for a center pixel in the reference image, the location of the most matching center pixel value in the matching image along the same line (epipolar line) is found from left to right. This is shown in Figure 5-9 (b). A threshold value is defined to determine whether a pair of pixels is a

good match. In the matching image, once the location of a pixel that has a similar value to the pixel on the reference image is found, the disparity value about the center pixel in the reference image is evaluated by subtracting the two locations of the center pixel in the reference and matching image from each other along the *x-axis*. This scanning procedure has to carry through for each pixel in the reference image. This makes it extremely inefficient in resource-limited systems.

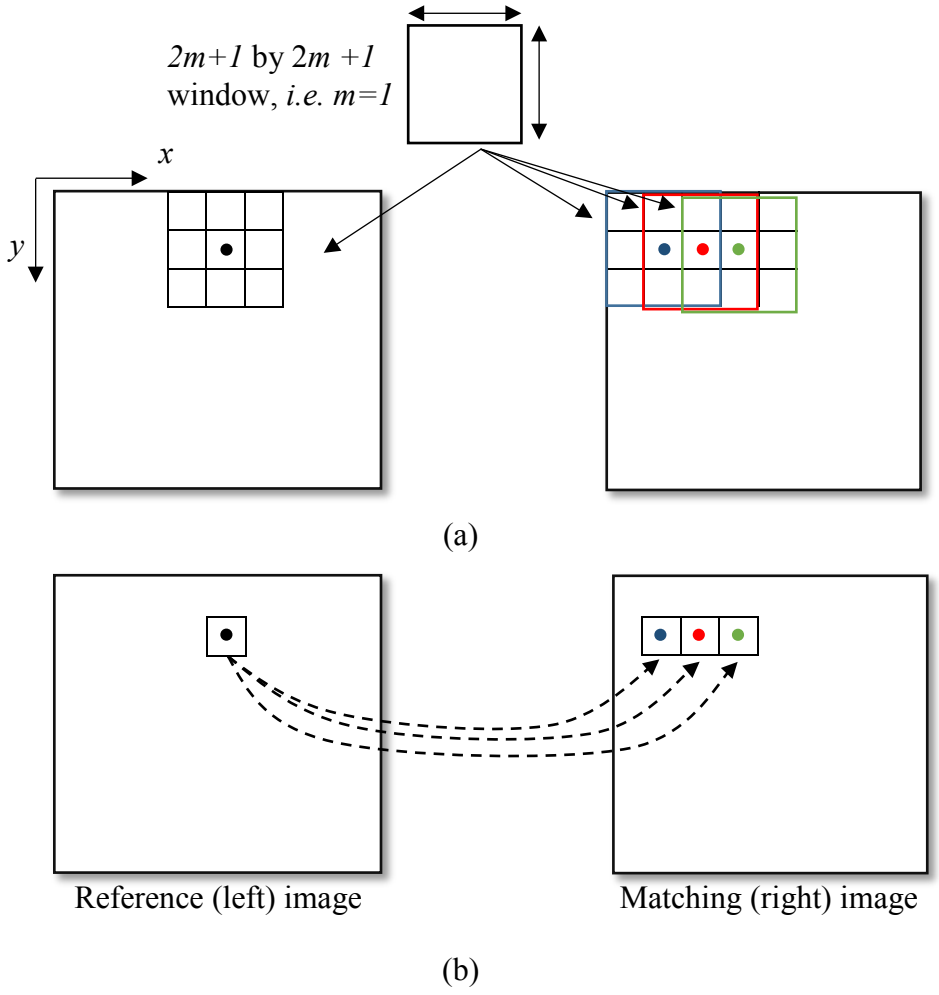


Figure 5-9 Conventional local window-based block matching (BM) method

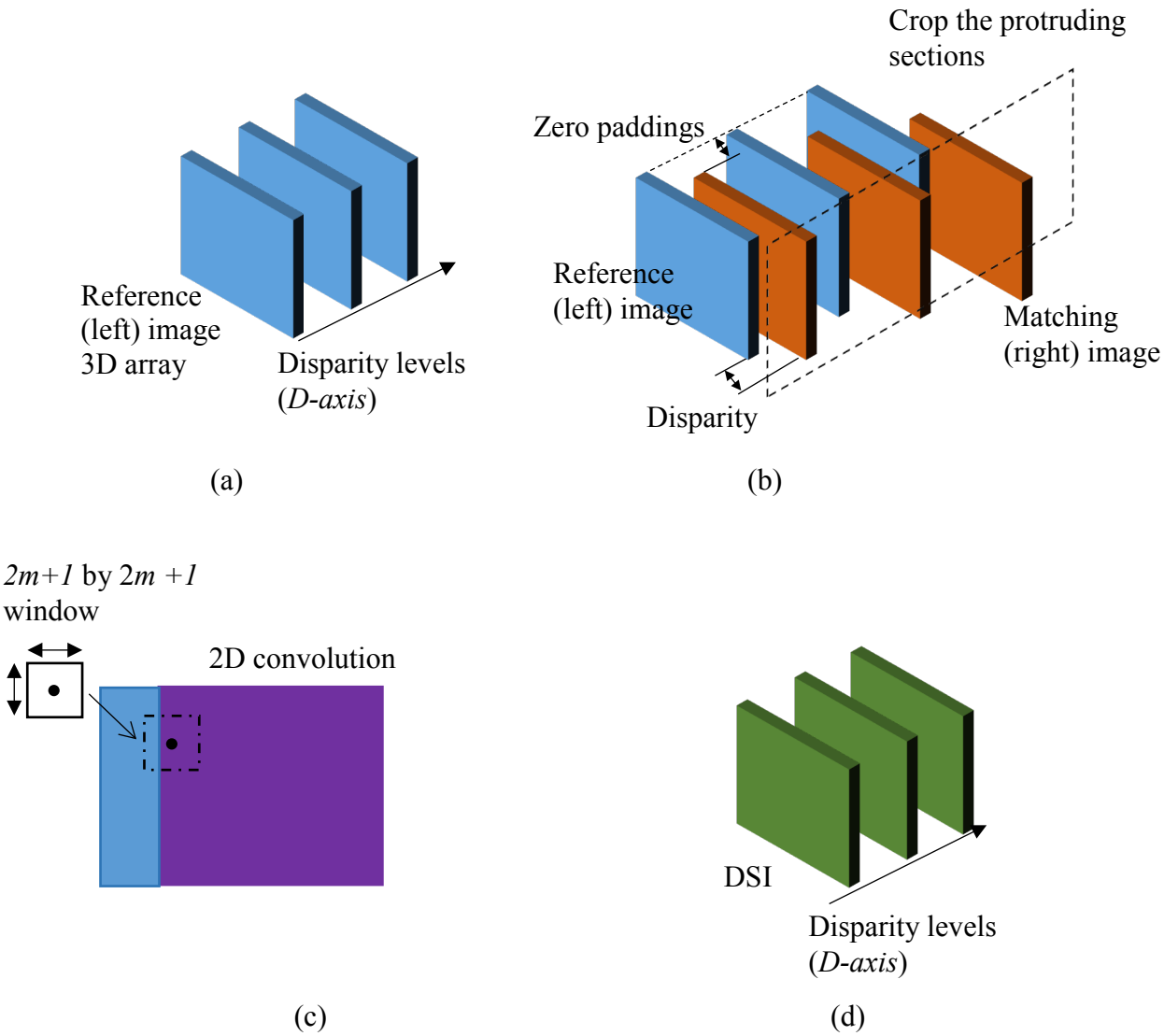
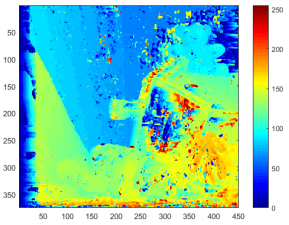


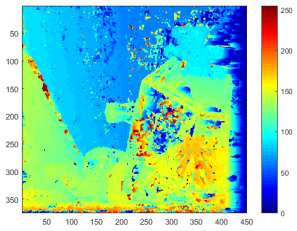
Figure 5-10 Vectorized local window-based Block Matching (BM) method

To facilitate the vectorization optimization in Matlab and to avoid using "for" loops, a new method is proposed to produce a DSI as depicted in Figure 5-10. The procedures can be described as following:

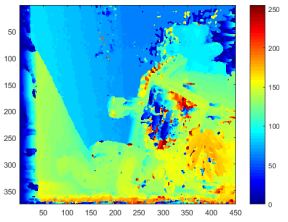
1) Given a rectified reference image (left image), duplicate this reference image in 3D space, such that the number of duplicated layers equals the disparity level along the *D-axis* as seen in Figure 5-10 (a).
2) Given a rectified matching image (right image), create a 3D duplicated space by padding zero to its right with a width that equals its dispaity level. This is illustrated in Figure 5-10 (b) by the orange slices.
3) Then find the difference between each blue slice and each right-padded orange slice. Crop the results to the size of the reference image to form a new 3D matrix.
4) Aggregation is performed by convoluting a $2m+1$ by $2m+1$ window that fills with ones about its center pixel in each layer as Figure 5-10 (c) shown.
5) As a result, a DSI is created as seen in Figure 5-10 (d). Then, the WTA strategy is applied on the aggregated DSI space along the *D-axis*. The smallest aggregated cost is selected as the disparity value for the specified pixel.
6) Redo these procedures by using the right image as a reference image and left image as a matching image to produce a new disparity map about the right image.
7) By comparing the two disparity maps, a binary mask that marks reliable and unreliable pixels is generated for L/R consistency check.
8) Apply this L/R check mask and the 2D medium filter (3 by 3) to the reference image disparity map. A more accurate disparity map is then produced.



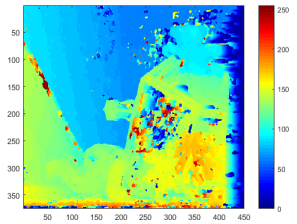
Window size 3, left image as reference



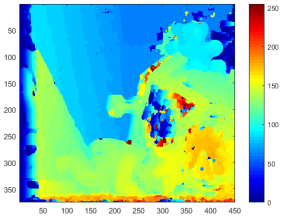
Window size 3, right image as reference



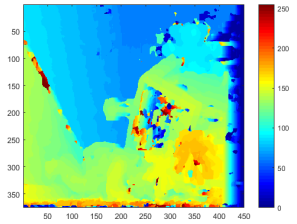
Window size 5, left image as reference



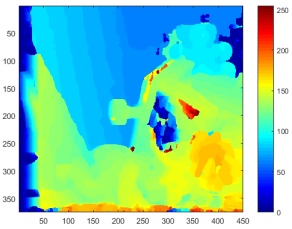
Window size 5, right image as reference



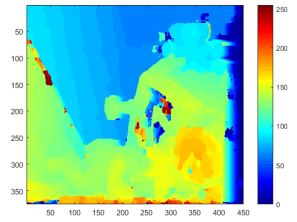
Window size 9, left image as reference



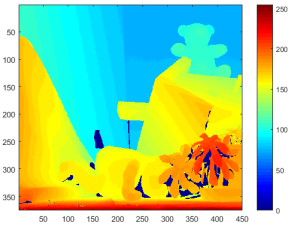
Window size 9, right image as reference



Window size 15, left image as reference



Window size 15, right image as reference



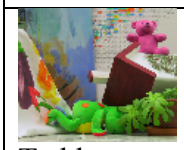
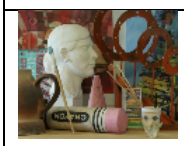
Ground truth

Figure 5-11 The Teddy at different square window sizes census (disparity levels up-scaled from 64 to 256)

As displayed in Figure 5-11, these results are generated by using BM (L/R check mask is not applied) with an up-scaled disparity map of the ground truth from 64 to 256. A 3 by 3 window medium filter is used to smoothen the results. From these disparity maps that are generated, it can be seen that the quality of matching in the BM method depends on the selection of the window sizes. This phenomenon can be observed near the chimney in Figure 5-11. The best window size for the Teddy image is found to be 5 by 5.

Other than the Teddy image, four datasets of Middlebury are also evaluated using the BM, and the results are shown in Table 5-3. It can be seen that BM will not perform well in the congested scene with many homogeneous surfaces (Con and Art images) even with a small window applied. Marking the unreliable pixels will significantly improve the accuracy of depth estimation of the valid matches and reduce the RMSE. However, some features may be excluded after applying the mask. The same datasets will be used to evaluate the SGM, and the findings, as well as the performance of SGM on the lab images, will be discussed later.

Table 5-3 Results of applying windows-based BM method in Middlebury stereo datasets

Test image	Window size	$PerLeft_{CK}$	$RMSE_{CK}$	$RMSE_{All}$	$PerCorr_{CK}$	$PerCorr_{All}$
 Teddy	3 by 3	78.12	6.4735	10.9162	89.86	66.01
	5 by 5	80.32	6.0512	10.7736	87.19	70.72
	15 by 15	83.43	5.9709	10.4511	86.19	71.96
 Con	3 by 3	57.32	13.1575	21.4515	46.06	5.76
	5 by 5	57.84	13.3457	21.3875	45.85	6.01
	15 by 15	59.54	14.6734	22.4500	45.29	6.80
 Art	3 by 3	57.35	10.6732	19.8637	62.21	26.25
	5 by 5	57.15	10.6773	20.1000	60.58	23.99
	15 by 15	58.07	12.5243	21.3632	54.34	17.76
 Books	3 by 3	57.70	11.2178	17.6360	60.80	25.64
	5 by 5	59.14	11.1465	17.8696	60.05	25.57
	15 by 15	61.16	12.6269	18.4516	57.10	23.88
 Cloth	3 by 3	57.01	7.3160	14.4909	69.44	35.72
	5 by 5	57.45	7.7669	14.7929	67.78	33.50
	15 by 15	59.41	10.3162	16.4174	56.34	21.08

Notation:
$PerLeft_{CK}$: Percentage of the reliable pixel marked by L/R consistency check
$RMSE_{CK}$: RMSE of calculated disparity map versus ground truth after applying L/R check mask
$RMSE_{All}$: RMSE of calculated disparity map versus ground truth
$PerCorr_{CK}$: Percentage of pixels that have correct disparity levels within threshold =2, after applying L/R check mask
$PerCorr_{All}$: Percentage of pixels that have correct disparity levels within threshold =2

5.4.3 Semi-Global Matching (SGM) method

5.4.3.1 Introduction

In 2006, Hirschmuller proposed the Semi-Global Matching (SGM) method for photogrammetry and the aerial survey, which eliminates the streaking artifacts that are produced in DP by performing systematically eight/sixteen scanning paths for a pixel. The

SGM aims to minimize the global energy function shown in Equation 5.3. Similar to the global energy function mentioned in the literature review, a global function consists of two parts. The first is the pixelwise matching cost and the second part constrains the local smoothness. Correspondingly, in Equation 5.3, $C\left(p, D_p\right)$ describes the matching cost on a pixel p according to the given disparity D_p. The second term sums all small penalty P_1 for the pixels whose disparity difference is at most one pixel within the neighbor of p. The third term sums all large penalty P_2 for all neighbor pixels with a disparity difference that is higher than one [111].

$$E(D)=\sum_{p}\left(C\left(p, D_p\right)+\sum_{q \in N_p} P_1 T\left[D_p-D_q=1\right]+\sum_{q \in N_p} P_2 T\left[D_p-D_q>1\right]\right) \text{ [68]} \quad (5.3)$$

5.4.3.2 Birchfield-Tomasi (BT) pixelwise cost matching calculation

The first step of dense disparity estimation is to determine the matching cost. The Birchfield-Tomasi (BT) method is employed as a cost calculation method to construct the DSI as suggested by [112]. This approach is insensitive to sampling effects and dissimilarity is computed by measuring how well a pixel in the left (reference) image will fit the linear interpolated region surrounding the possible correspondence in the right (matching) image [66]. The dissimilarity of a point in the left image x_L and a point in the right image x_R is described as Equation 5.4. The search space for x_L and x_R will be constrained along two corresponding epipolar lines.

$$d(x_L, x_R) = \min\left\{\overline{d}(x_L, x_R, I_L, I_R), \overline{d}(x_R, x_L, I_R, I_L)\right\} \text{ [66]} \tag{5.4}$$

where

$$\begin{aligned} I_R^- &= \frac{1}{2}\left(I_R(x_R) + I_R(x_R - 1)\right), I_R^+ = \frac{1}{2}\left(I_R(x_R) + I_R(x_R + 1)\right) \\ I_L^- &= \frac{1}{2}\left(I_L(x_L) + I_L(x_L - 1)\right), I_L^+ = \frac{1}{2}\left(I_L(x_L) + I_L(x_L + 1)\right) \end{aligned} \tag{5.5}$$

$$\begin{aligned} I_{\min L} &= \min\left\{I_R^-, I_R^+, I_R(x_R)\right\}, I_{\max L} = \max\left\{I_R^-, I_R^+, I_R(x_R)\right\} \\ I_{\min R} &= \min\left\{I_L^-, I_L^+, I_L(x_L)\right\}, I_{\max R} = \max\left\{I_L^-, I_L^+, I_L(x_L)\right\} \end{aligned} \tag{5.6}$$

$$\begin{aligned} \overline{d}(x_L, x_R, I_L, I_R) &= \max\left\{0, I_L(x_L) - I_{\max R}, I_{\min R} - I_L(x_L)\right\} \\ \overline{d}(x_R, x_L, I_R, I_L) &= \max\left\{0, I_R(x_R) - I_{\max L}, I_{\min L} - I_R(x_R)\right\} \end{aligned} \tag{5.7}$$

Similarly, the symmetric quantity $d(x_R, x_L)$ can be easily derived. However, to compute $d(x_L, x_R)$ and $d(x_R, x_L)$, there may be a tedious implementation procedure to generate a DSI matrix. Then, a vectorized BT method is proposed as follows. It only contains two for loops in the BT pixelwise matching computation.

1) Select a column for the left image as x_L, *i.e.* *n*-th column in left (reference) image
2) Find the corresponding (*n*-1) -th column as x_R in right image (matching) that may contain pixels that match to the pixels in *n*-th column of the left image at disparity level 1. As shown in Figure 5-12 (a), each box contains 3 column vectors and the vertical arrow above the box represents the center column. Additionally, the center column vector of the red box (where the center column vector is at (*n*-1)-th column), the green box and purple may contain pixels that match the pixels in the center column vector of the blue box in the left image at disparity level 1, 2, and 3,

respectively. The locations of each box in the right image of Figure 5-12 (a) are offset by one column.

3) Extract the *n*-th column vector and its neighboring column vectors at (*n*-1)-th and (*n*+1)-th column in the left image. Extract the column vectors at (*n*-2)-th, (*n*-1)-th, and *n*-th column in the right image and partition these 6 vectors into 6 matrices as shown in Figure 5-12 (b).
4) Then apply Equation 5.4 on 6 matrices as illustrated in Figure 5-12 (b). The dissimilarities about the *n*-th column vector in the left image at disparity level 1 is found.
5) Similarly, step 3 and step 4 can be applied to find the dissimilarities about the *n*-th column vector in the left image at disparity level 2, *i.e.* the green box shown in Figure 5-12 (c).
6) Step 3 to step 4 can be grouped into a 3D matrix operation, in which the red, green and purple boxes in Figure 5-12 may be allocated as slices along the disparity axis.
7) Loop the above procedures for *n* in the range of 1 to the width of the image to obtain a DSI.

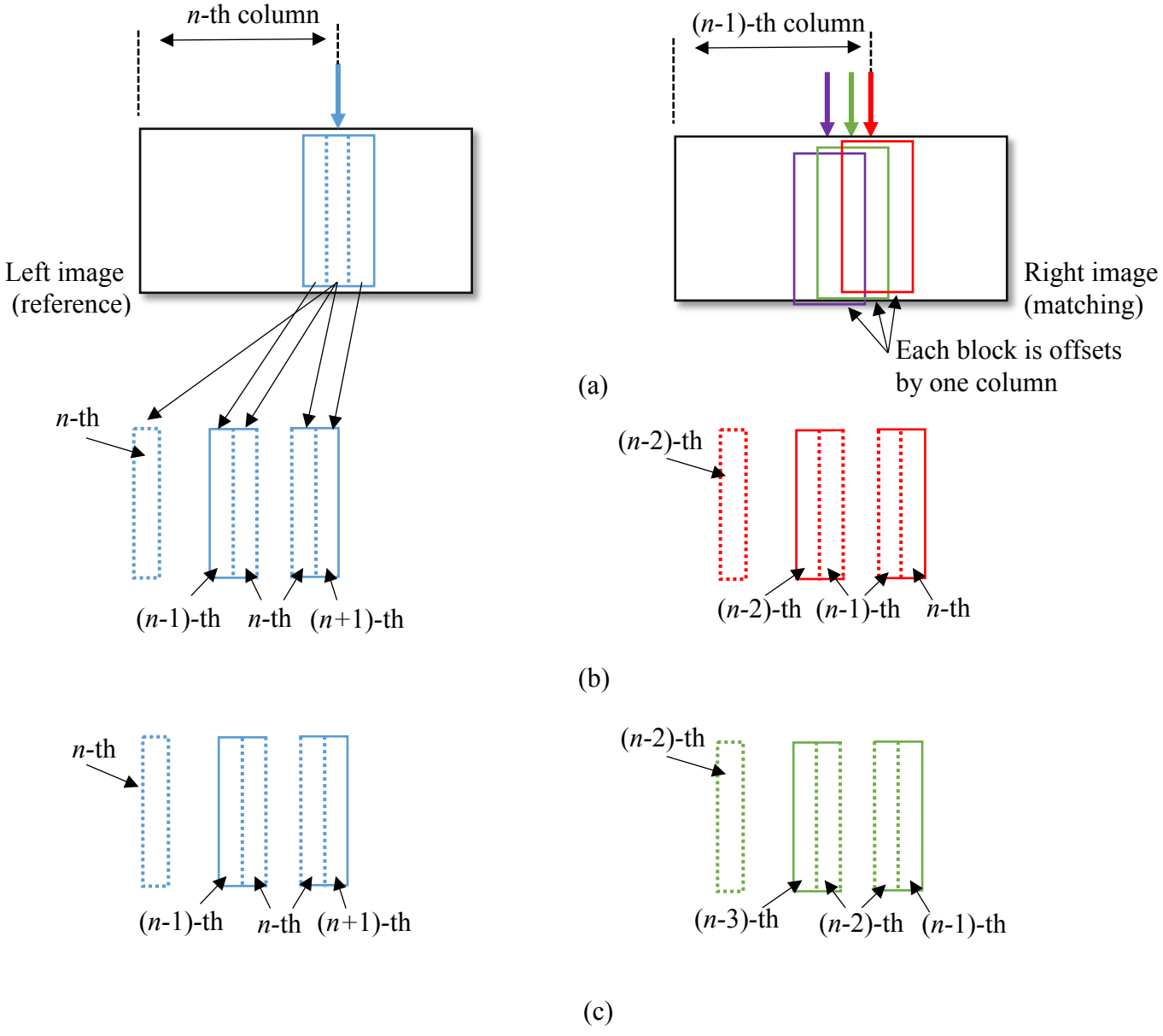


Figure 5-12 Vectorized BT pixel wise matching method

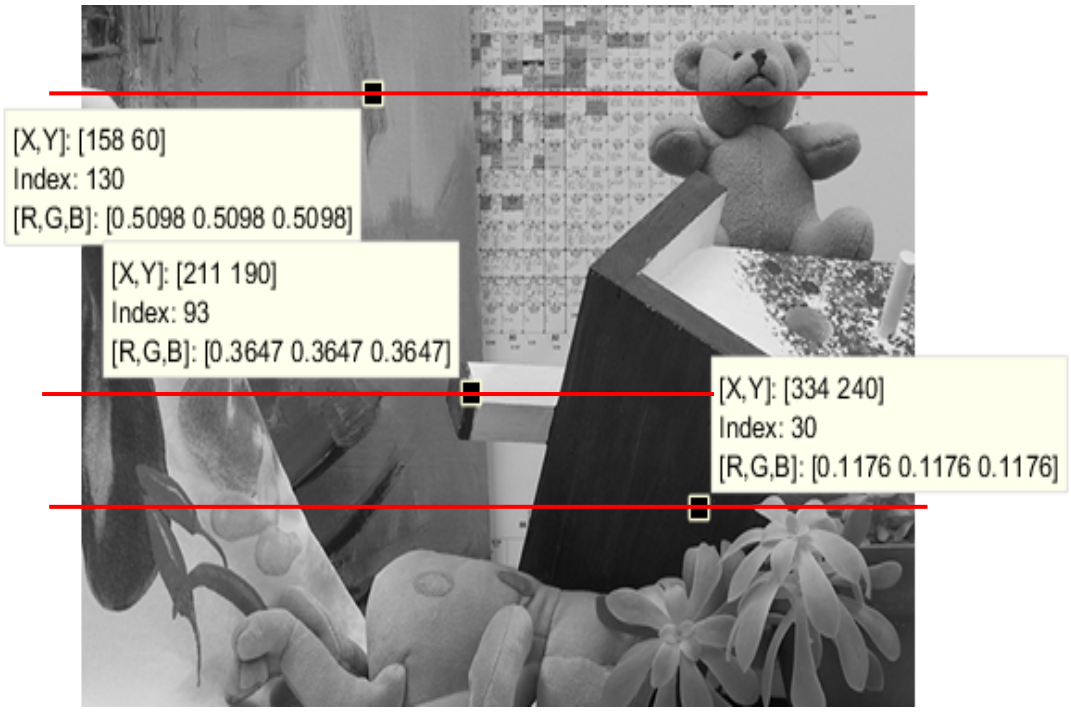


Figure 5-13 Teddy image for matching cost and DSI illustration

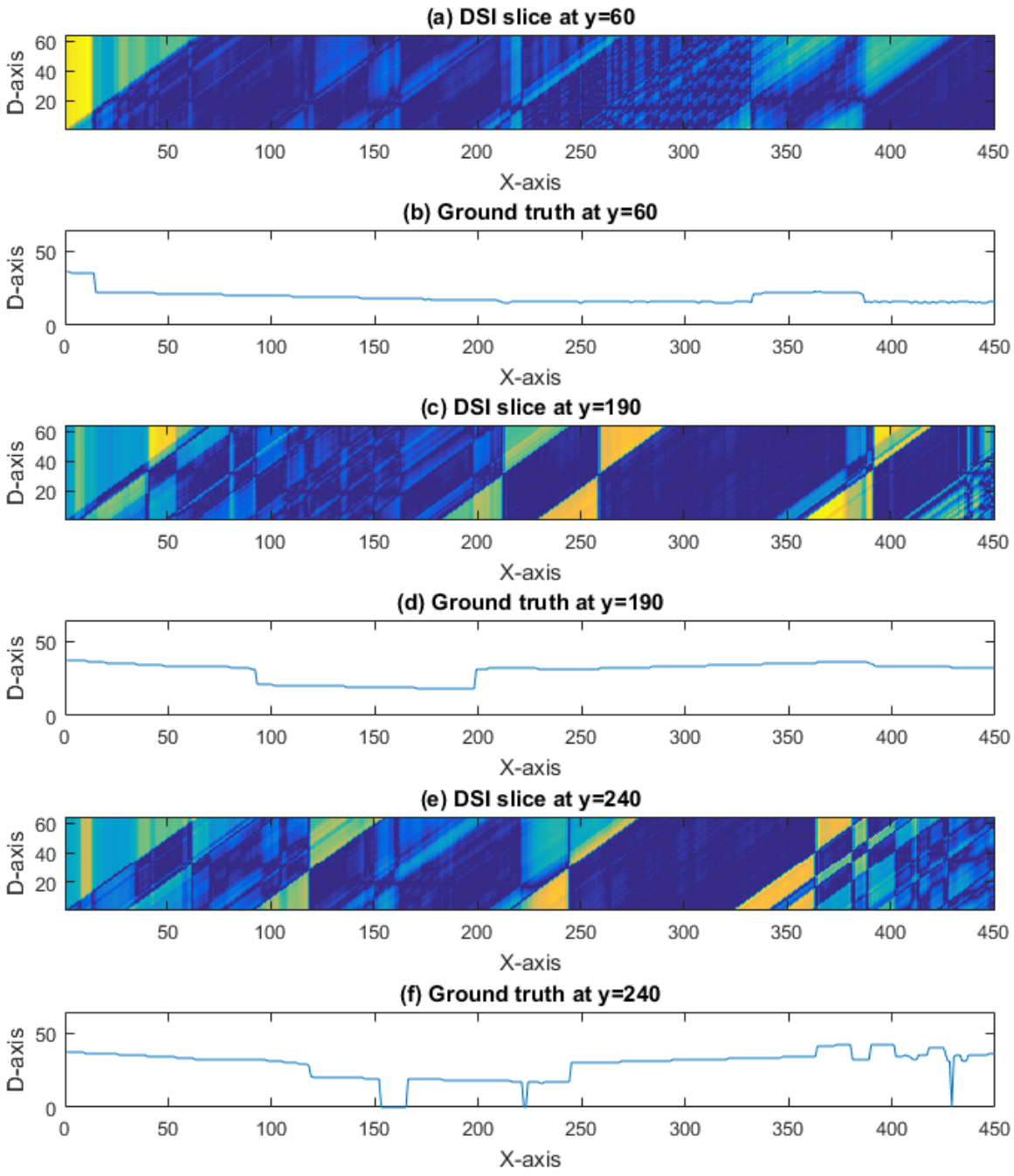


Figure 5-14 Comparison of calculated DSI to the ground truth in the Teddy image

After applying the vectorized BT pixelwise matching on the Teddy image in Figure 5-14, three selected slices of DSI, which are cut according to the locations marked in Figure 5-13, are shown in the image. The darker segments indicate that there may be a high likelihood that the disparity value will be found near that disparity level, while small bands that almost blend into the DSI background, like from 280 to 320 along the *x-axis* in (c) and (e), are usually the result of texture-less regions. Comparing the estimated DSI to the ground truth, they seem to be well matched. However, in pixelwise matching cost, the smoothness constraints of neighboring disparities are generally not applied yet, resulting in ambiguity and mismatches. The next step will solve this problem by aggregation.

5.4.3.3 Cost aggregation

Equation 5.3 describes a NP-complement problem [68] to find the disparity image D that minimize the energy function $E(D)$. Meanwhile, DP can be performed in 1D to minimize the energy function row by row efficiently while the results suffer streaking effects. To unconstrain the 1D search problem, the cost shall be aggregated from all of the directions in 1D equally. This leads to the creation of SGM [68]. As shown in Equation 5.8, the aggregation process can be described recursively. The first term describes the matching cost that is calculated from TB or MI. The second term is the minimal path costs of the previous pixel. The third term is the overall minimum path cost of the previous pixel, and it prevents the rapidly increasing path cost from the second term. Notably, the second term requires the minimum value of the previous path cost at the same disparity *d,* previous path cost at disparity *d*-1 with a small penalty*,* previous path cost at disparity *d*+1 with a small penalty, and a minimum value of previous path cost that excludes the disparity at *d-1*, *d*

and $d+1$ with a large penalty. The second term with penalties enforces the smoothness near the neighboring pixels of p.

$$L_r(p,d) = C(p,d) + \min(L_r(p-r,d), \\ L_r(p-r,d-1)+P_1, \\ L_r(p-r,d+1)+P_1, \\ \min_i L_r(p-r,i)+P_2) - \min_k L_r(p-r,k) \tag{5.8}$$

$$S(p,d) = \sum_t L_r(p,d) \tag{5.9}$$

Once the eight/sixteen path costs have been calculated, the WTA is applied to the results of the sum of all path costs in Equation 5.9.

$$
\begin{aligned}
L_{r(x,y)}\left(p(x,y),d\right) = C\left(p(x,y),d\right) + \min(&L_{r(x,y)}\left(\left[p(x,y)-r(c,r)\right],d\right),\\
&L_{r(x,y)}\left(\left[p(x,y)-r(c,r)\right],d-1\right)+P_1,\\
&L_{r(x,y)}\left(\left[p(x,y)-r(c,r)\right],d+1\right)+P_1,\\
&\min_i L_{r(x,y)}\left(\left[p(x,y)-r(c,r)\right],i\right)+P_2)\\
-&\min_k L_{r(x,y)}\left(\left[p(x,y)-r(c,r)\right],k\right)
\end{aligned}
\qquad (5.10)
$$

where:

$C\left(p(x,y),d\right)$ is the matching cost that calculated from TB pixelwise matching at current pixel $p(x,y)$ and at disparity level d.

$\left[p(x,y)-r(c,r)\right]$ is defined as the location of the previous pixel in the scanning path.

$r(c,r)$ represents a scanning path, *i.e.*, point 3 in Figure 5-15 is $r(1,0)$.

$L_{r(x,y)}\left(p(x,y),d\right)$ is path cost at $p(x,y)$ at *d*-th (same) disparity level.

$L_{r(x,y)}\left(\left[p(x,y)-r(x,y)\right],d\right)$ is cost at $\left[p(x,y)-r(x,y)\right]$ at *d*-th disparity level.

$\min_i L_{r(x,y)}\left(\left[p(x,y)-r(x,y)\right],i\right)$ is the minimum cost at $\left[p(x,y)-r(x,y)\right]$ excluding $d, d+1, d-1$ th disparity levels.

$\min_k L_{r(x,y)}\left(\left[p(x,y)-r(x,y)\right],k\right)$ is the local minimum cost of the previous pixel along disparity axis in DSI.

P_1 is a pre-defined small penalty.

$$
P_2 = \frac{P_2'}{\left|I_{\left[p(x,y)-r(x,y)\right]} - I_{p(x,y)}\right|} \quad (5.11)
$$

[68] is the large penalty, where P_2' is a pre-defined value.

Table 5-4 Lookup table for the scanning paths

r (index)	x coord. (Col)	y coord. (Row)	Start corner		r (index)	x coord. (Col)	y coord. (Row)	Start corner
1	-1	0	Top		9	-2	-1	Top
2	0	-1	Top		10	-1	-2	Top
3	+1	0	Bottom		11	+1	-2	Bottom
4	0	+1	Bottom		12	+2	-1	Bottom
5	-1	-1	Top		13	+2	+1	Bottom
6	+1	-1	Bottom		14	+1	+2	Bottom
7	+1	+1	Bottom		15	-1	+2	Top
8	-1	+1	Top		16	-2	+1	Top

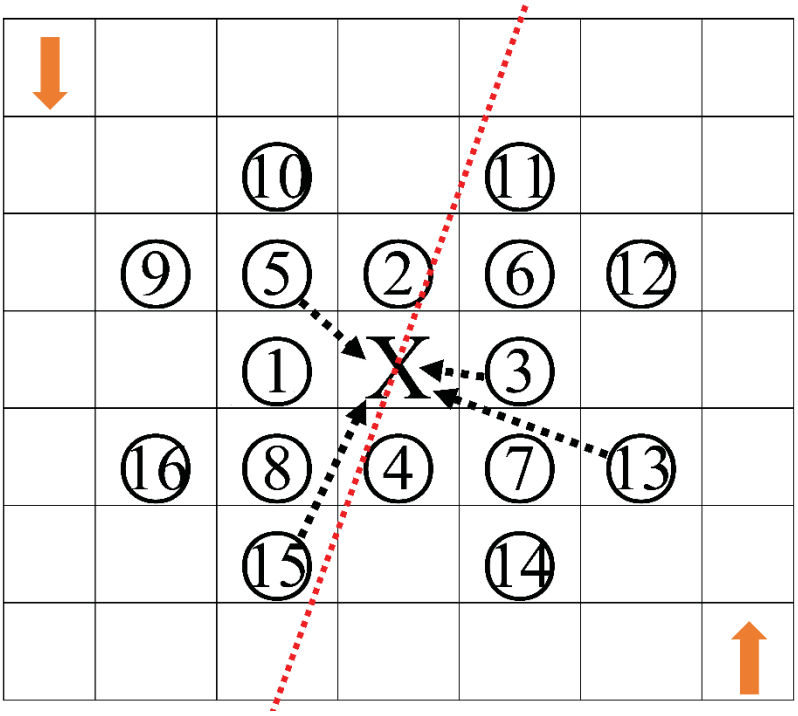
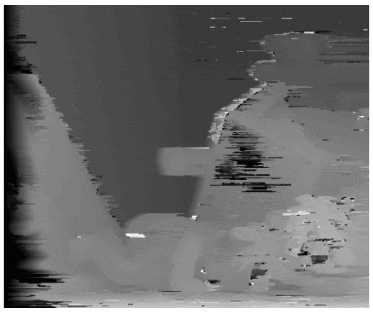
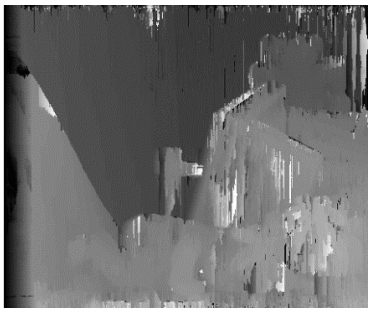
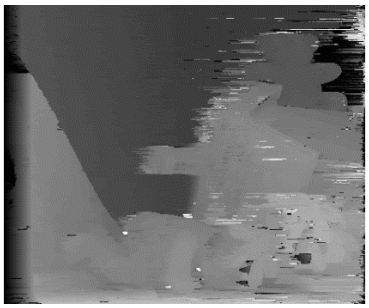
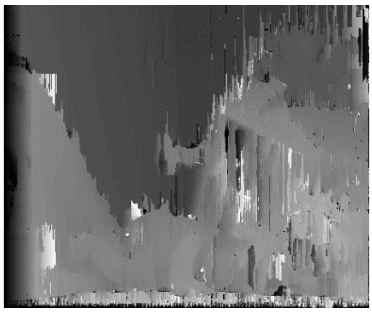


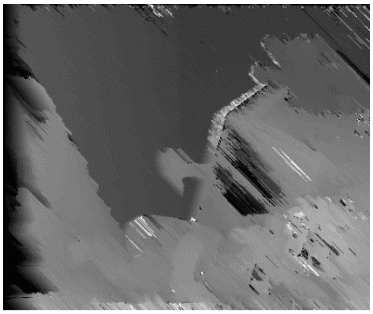
Figure 5-15 Sixteen scan path directions in a discrete pixel domain (X and its sixteen neighbors)

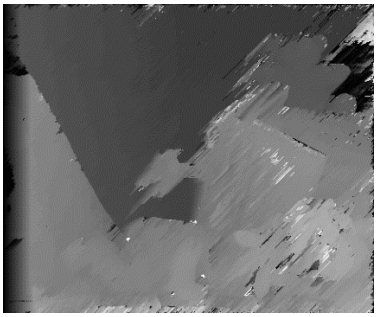

Path 1

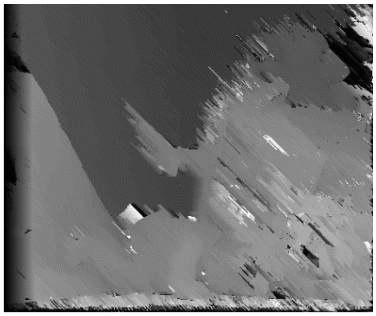

Path 2

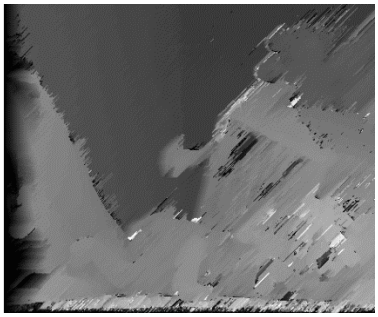

Path 3

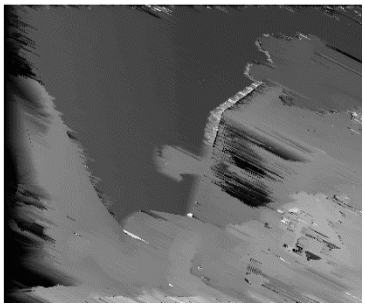

Path 4

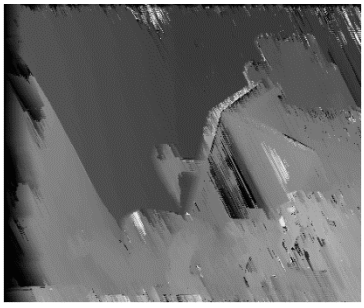

Path 5

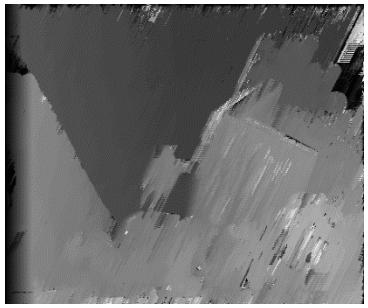

Path 6

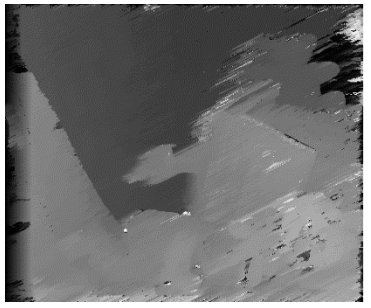

Path 7

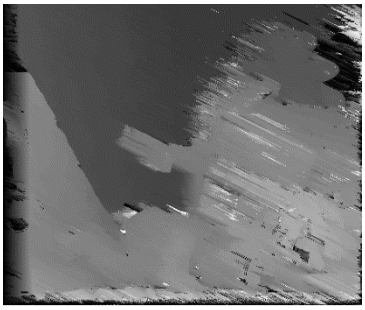

Path 8

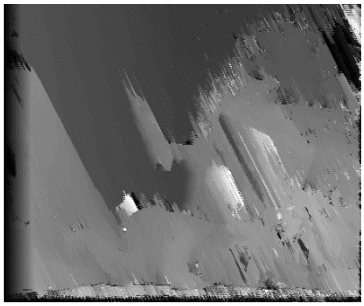

Path 9

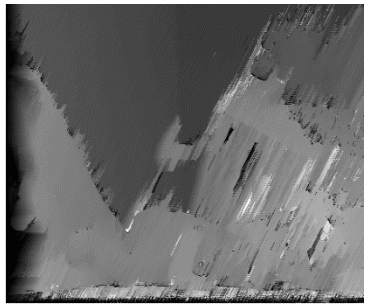

Path 10


Path 11


Path 12


Path 13


Path 14


Path 15

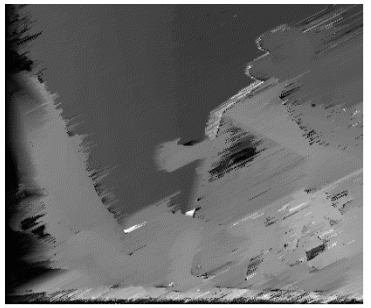

Path 16

Figure 5-16 Sixteen scanning paths on the Teddy image

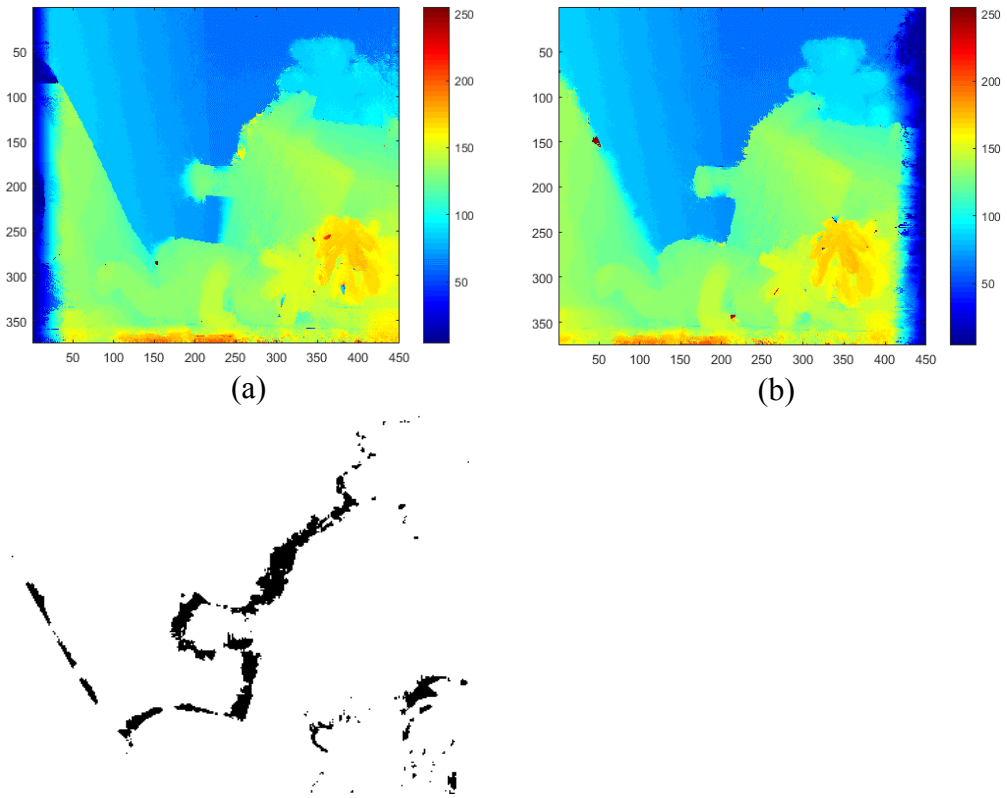
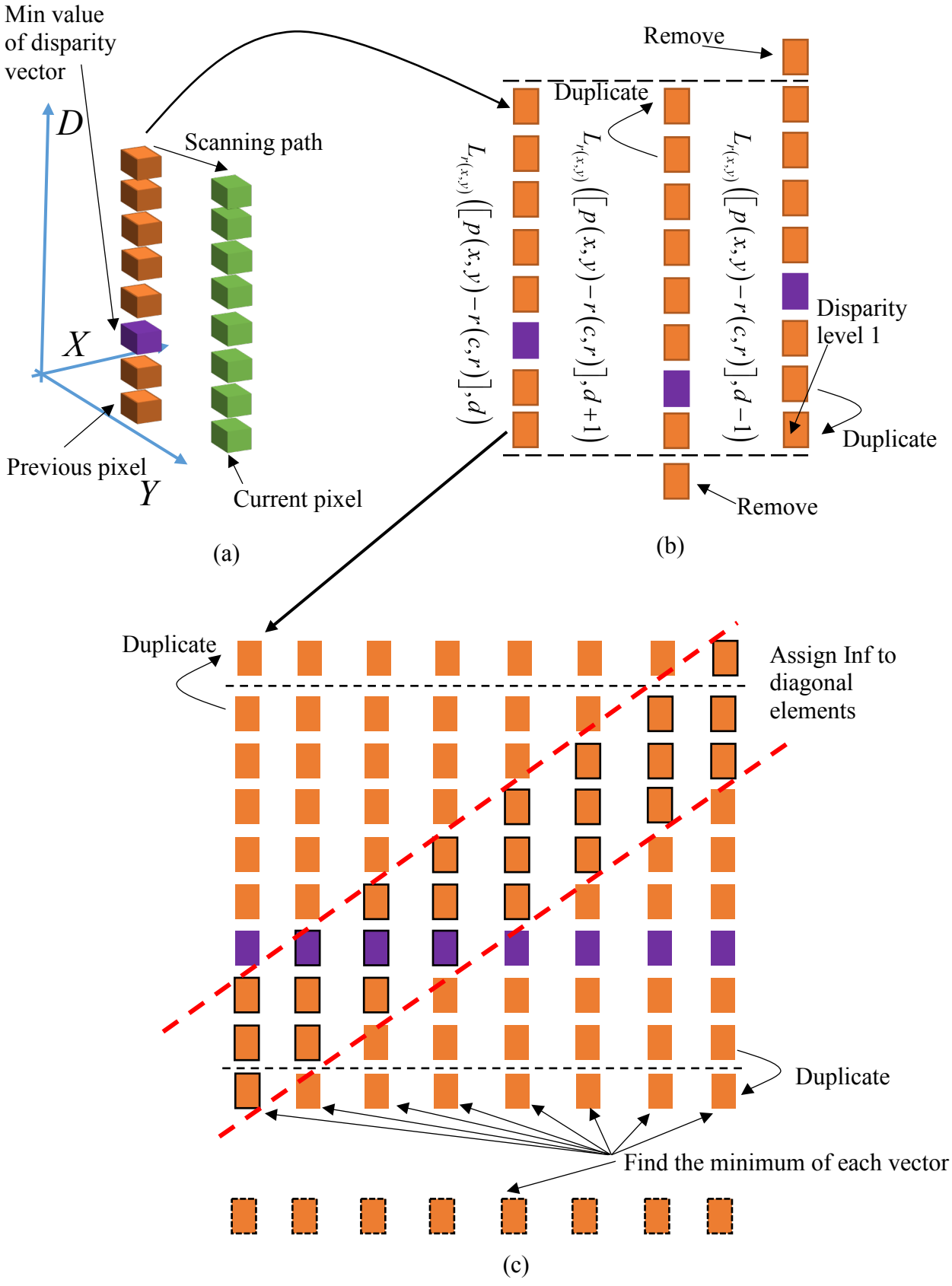


Figure 5-17 Teddy disparity map generated by SGM, (a) left image as reference image, (b) right image as reference image, (c) L/R consistency check mask that generated by (a) and (b) and black regions represent unreliable pixels

To aggregate the initial matching costs that are calculated by the BT pixelwise method, the SGM only recursively calculates the path costs instead of matching costs. Equation 5.8 can be easily interpolated as Equation 5.10 in the implementation. A lookup table is accordingly created because of discrete image space as listed in Table 5-4. In the implementation, the aggregation will be performed in two passes. The first pass starts from the top left corner and uses the location 1, 2, 5, 8, 9, 10, 15, 16 as previous path costs in Figure 5-15. The second pass starts from the bottom right corner and uses location 3, 4, 6, 7, 11, 12, 13, and 14 as previous path costs. The initial path costs for the borders are directly adapted from the TB pixelwise matching cost.


Min value of disparity vector
D
X
Y
Scanning path
Previous pixel
Current pixel
(a)
Remove
Duplicate
Disparity level 1
Duplicate
Remove
(b)
Duplicate
Assign Inf to diagonal elements
Duplicate
Find the minimum of each vector
(c)

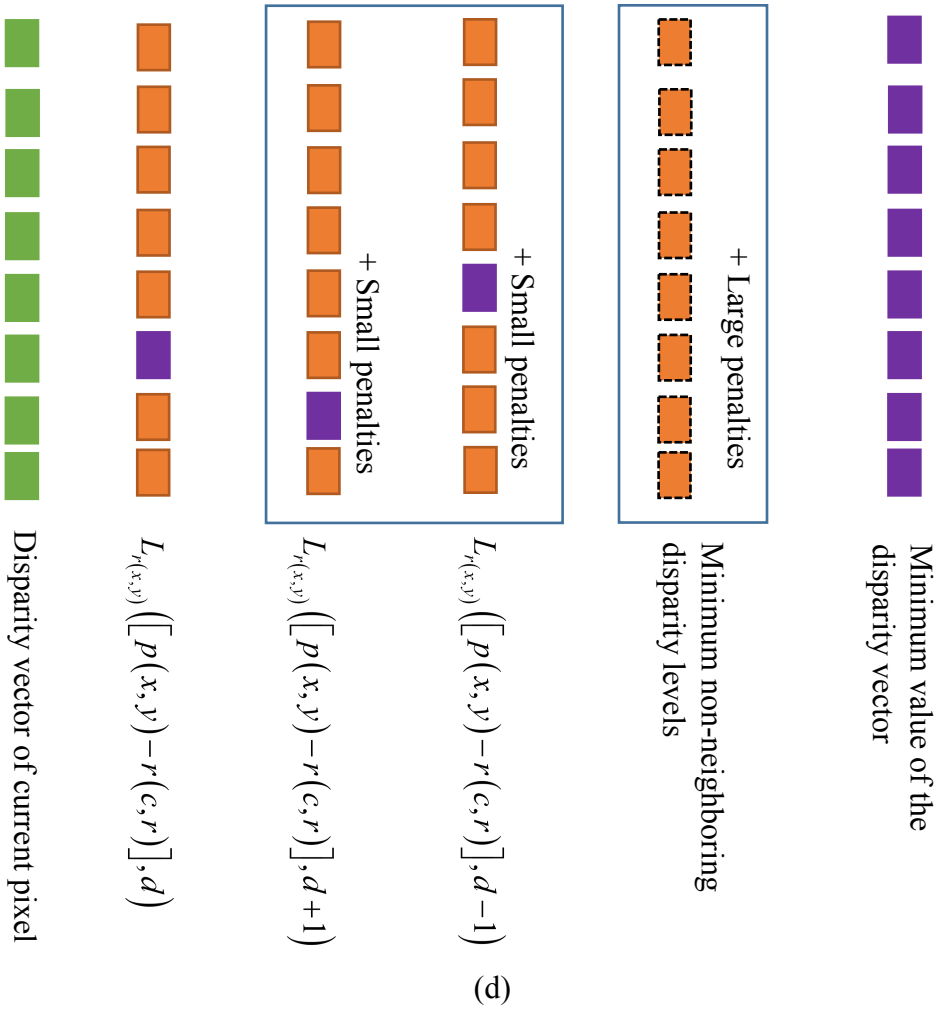


Figure 5-18 Vectorized single path scanning in SGM about a pixel

Generally, in order to perform a path scanning, at least four nested "for" loops are required to scan through paths, columns, rows, and disparity levels, respectively. To reduce the computation complexity, a vectorized and parallel SGM algorithm is proposed. In a single path aggregation, only two "for" loops are used in calculating Equation 5.8, and using one "parfor" loop to parallel the single path scanning. Then, the scanning results of 16 paths are combined eventually, since each scanning path will produce an aggregated DSI. The

advantage of this new method avoids the scanning of the disparity levels. The proposed method can be outlined as follows.

1) Load the lookup table of the scanning paths in the memory as listed in Table 5-4. Load the DSI calculated by BT pixelwise matching.
2) Initialize a "parfor" (parallel for) loop, so that, all of the 16 scanning paths will be executed concurrently.
3) In single path scanning, for a pixel, extract a column vector of the previous pixel about the disparity axis in the scanning path, *i.e.,* $L_{r(x,y)}\left(\left[p(x,y)-r(c,r)\right],d\right)$ as shown in Figure 5-18 (a).
4) Find the minimum value of the disparity vector of the previous pixel. This minimum value only needs to be calculated once for the term $\min_k L_{r(x,y)}\left(\left[p(x,y)-r(c,r)\right],k\right)$ in Equation 5.10 for one pixel location.
5) Locate $L_{r(x,y)}\left(\left[p(x,y)-r(c,r)\right],d-1\right)$ and $L_{r(x,y)}\left(\left[p(x,y)-r(c,r)\right],d+1\right)$ by shifting $L_{r(x,y)}\left(\left[p(x,y)-r(c,r)\right],d\right)$ up or down by one space, and pad the empty space by the nearest value as shown in Figure 5-18 (b). Then the small penalties can be applied to the previous pixel's disparity levels in a vector form.
6) Duplicate the disparity vector of the previous pixel into a matrix, and pad the first row and the last row by using the nearest column value as shown in Figure 5-18 (c).
7) Create a logical diagonal matrix as listed in Table 5-5. This is used to assign "Inf" (infinity) values to exclude $L_{r(x,y)}\left(\left[p(x,y)-r(x,y)\right],d\right)$ and its neighbors in disparity vectors.

8) Find the minimum values of each column vector, which will result in a row vector. Rotate this row vector to a column vector.

9) Now, the disparity vector, about a pixel $p(x, y)$, can be found by Equation 5.10 using the vectors listed in Figure 5-18 (d).

10) Repeat step 3 to 9 for each pixel in the DSI for single path scanning. The use of two "for" loops will be applied here. If the previous pixel is out of the border, then use the current pixel matching cost values.

11) Similarly, the single path scanning will begin from the bottom right corner.

12) The single path scanning can be performed in parallel. Save the DSI onto local storage and release the memory once a scanning path is completed.

13) After all of the 16 paths have been scanned, all 16 DSI are loaded from the local storage and they are merged together by summing them as illustrated in Equation 5.9.

14) Apply WTA to the merged DSI and generate a disparity map.

15) Repeat step 1 to step 14 to generate a disparity map for right image.

16) By comparing the two disparity maps, a binary mask that marks reliable and unreliable pixels is generated for L/R consistency check.

17) Apply this L/R check mask and the 2D medium filter (3 by 3) to the reference image disparity. A more accurate disparity map is then produced.

Table 5-5 A diagonal binary mask that excludes the neighboring costs in DSI

0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	1
0	0	0	0	0	1	1	1
0	0	0	0	1	1	1	0
0	0	0	1	1	1	0	0
0	0	1	1	1	0	0	0
0	1	1	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0

Another ingenuity in SGM is to use an adaptive penalty system that was proposed by Hirschmuller [68] to preserve edges at discontinuity regions. As in Equation 5.11, it shows an adaptive method to assign a large penalty P_2 with the change of intensity between the current and previous pixel. Thus, a smaller penalty will be allocated to the previous path cost near edge regions (discontinuities), and the sudden change of disparity (depth) can be highlighted among all previous path costs in the second term of Equation 5.8. This results in a clear disparity step. However, if the current pixel and previous pixel are within the smooth area with similar disparities, this large penalty will be ignored. It is required that P_1 shall always be smaller than $P_2^{'}$. Finally, Figure 5-16 shows all 16 scanning paths on the Teddy image, and the disparity map is shown in Figure 5-17 with the L/R consistency check mask.

5.4.3.4 Penalty function selection and evaluation of SGM method using Middlebury dataset

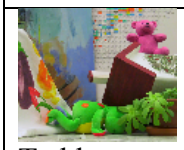
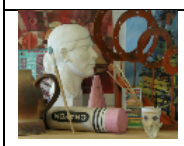
In addition to the adaptive penalty method, three different penalty functions are investigated by [113], and the results demonstrate that the original adaptive penalty

assignment method works moderately well among the three candidates. In this section, another method proposed by Hu [114], which assigns constant P_1 at edge pixels and P_2 at non-edge pixels, is further exploited, and three different penalty functions are evaluated.

1) Using a Canny edge detector [44] to deter edges, and assign the small penalty P_1 =5 to previous path costs as P_2 at the edge pixels, while at non-edge pixels, the adaptive penalty method is used for $P_2^{'}$=600
2) Same as adaptive method proposed by Hirschmuller using P_1=5, $P_2^{'}$=600
3) Using the constant penalties, such as P_1=5, P_2=60

After examining the results in Table 5-6, the results of three different methods are comparable. With the conclusions made by the study in [113], it is safe to conclude that the adaptive penalty function proposed by Hirschmuller is robust and modification is not required in the calculation. Since it encodes the small penalty for discontinuities, edges are not obliged to have special labeling in the computation.

Table 5-6 Comparison of three penalty functions using SGM method

Test image	Test method	$PerLeft_{CK}$	$RMSE_{CK}$	$RMSE_{All}$	$PerCorr_{CK}$	$PerCorr_{All}$
 Teddy	1	89.08	5.2408	8.6585	89.86	80.11
	2	90.06	5.3656	8.3685	88.51	79.59
	3	90.90	5.5544	8.4684	86.55	78.19
 Con	1	89.74	5.1592	9.7986	87.20	77.67
	2	90.04	5.0538	9.6258	88.14	79.13
	3	91.12	5.2708	9.2928	87.17	79.32
 Art	1	76.56	5.4445	15.4981	78.80	58.10
	2	77.13	5.9266	15.0321	75.50	55.55
	3	80.23	5.9619	14.6726	75.36	57.54
 Books	1	82.66	4.3742	10.5975	85.26	70.41
	2	84.38	4.3601	10.2108	84.01	69.98
	3	85.89	4.1319	10.1614	84.33	71.00
 Cloth	1	90.25	1.1637	9.3344	99.43	89.77
	2	90.36	1.1251	9.0521	99.52	90.00
	3	90.41	1.1138	8.8466	99.49	90.44

Notation: Same as Table 5-3

5.4.4 Comparison of using BM and SGM in Middlebury stereo datasets

The qualitative evaluation results of the BM method and SGM method are given in Table 5-3 and Table 5-6, respectively. When comparing the calculated disparity map against the ground truth, the SGM method produces a significantly higher correct rate than the BM method even without using L/R check mask, especially in the Con and Art images. The SGM method also preserves a more valid pixel in the disparity map after L/R check mask

is applied. The RMSEs are lower in SGM in general. In the following discussion, the focus is on applying the BM and SGM in real world data.

5.4.5 3D and 2D reconstructions

Dense disparity map estimation facilitates accurate 3D and 2D scene reconstructions. The lab image datasets are serviced as input images to evaluate two dense stereo matching methods in the real world scenarios. In SGM, two penalties are defined as P_1=30, $P_2^{'}$=1000, and the window size is determined to be 49 by 49 in the BM method by trial and error. The estimated depth from the markers to the left camera is illustrated in Table 5-7. The 3D reconstruction is accomplished by the given disparity map and camera calibration and then rendered by the Matlab Computer Vision Toolbox. For robots to detect the nearby obstacles, the 2D binary maps are generated along the horizontal plane of each marker using Equation 2.5. The cameras with/without an IR filter yield similar results. Thus, only the results obtained from the NonIR filter camera are presented here.

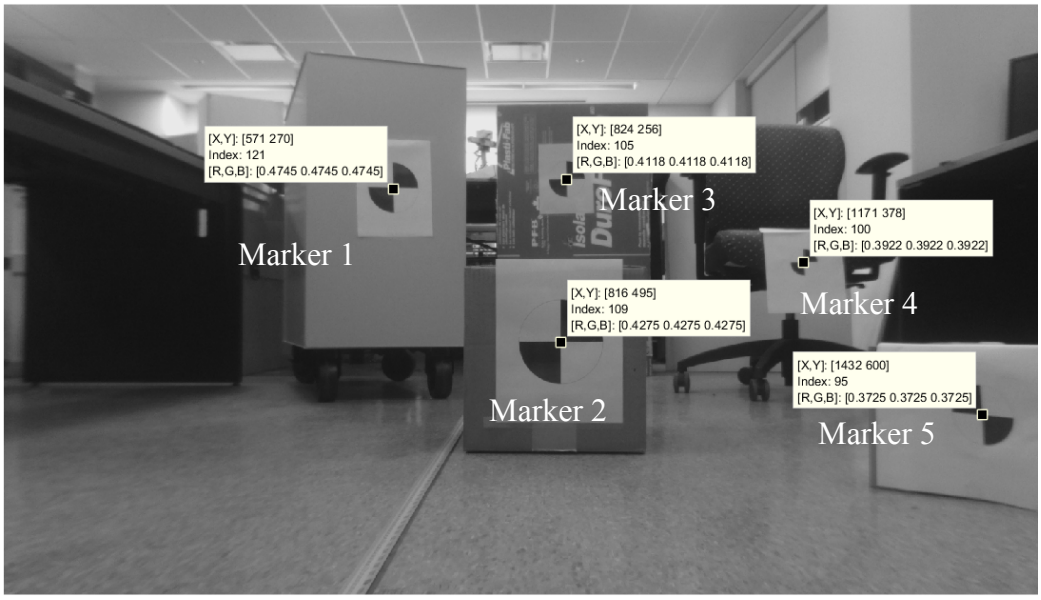


Figure 5-19 Marker positions in the lab image

Table 5-7 Five marker positions respect to left camera by stereo triangulation and real depth measurements (all units are in meters)

	Real depth	Images without an IR filter			Images with an IR filter		
		BM method	SGM 8 paths	SGM 16 paths	BM method	SGM 8 paths	SGM 16 paths
Marker 1	2.55	2.598	2.598	2.598	2.512	2.512	2.487
Marker 2	1.50	1.556	1.556	1.556	1.483	1.483	1.483
Marker 3	3.60	3.665	3.665	3.665	3.624	3.573	3.624
Marker 4	2.50	2.524	2.524	2.524	2.487	2.478	2.487
Marker 5	1.10	1.119	1.115	1.115	1.098	1.098	1.098

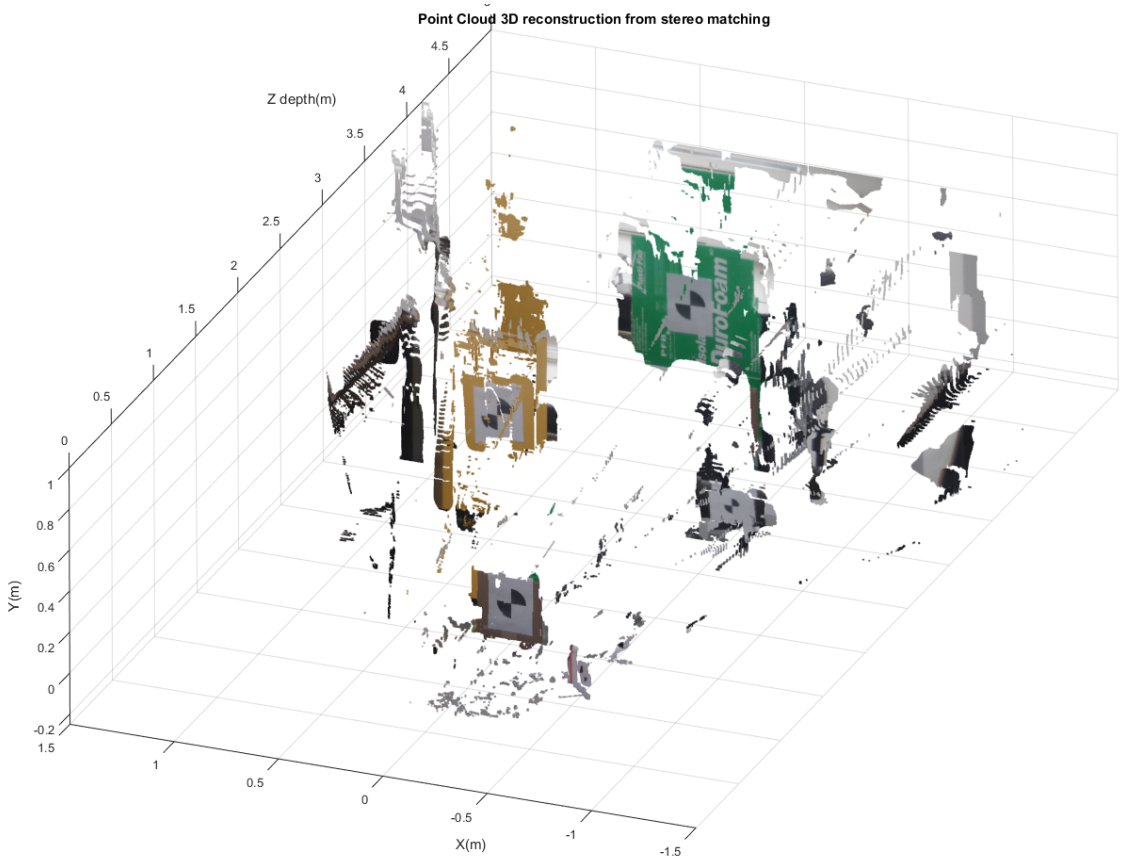


Figure 5-20 A 3D rendered reconstruction from BM method on NonIR camera image

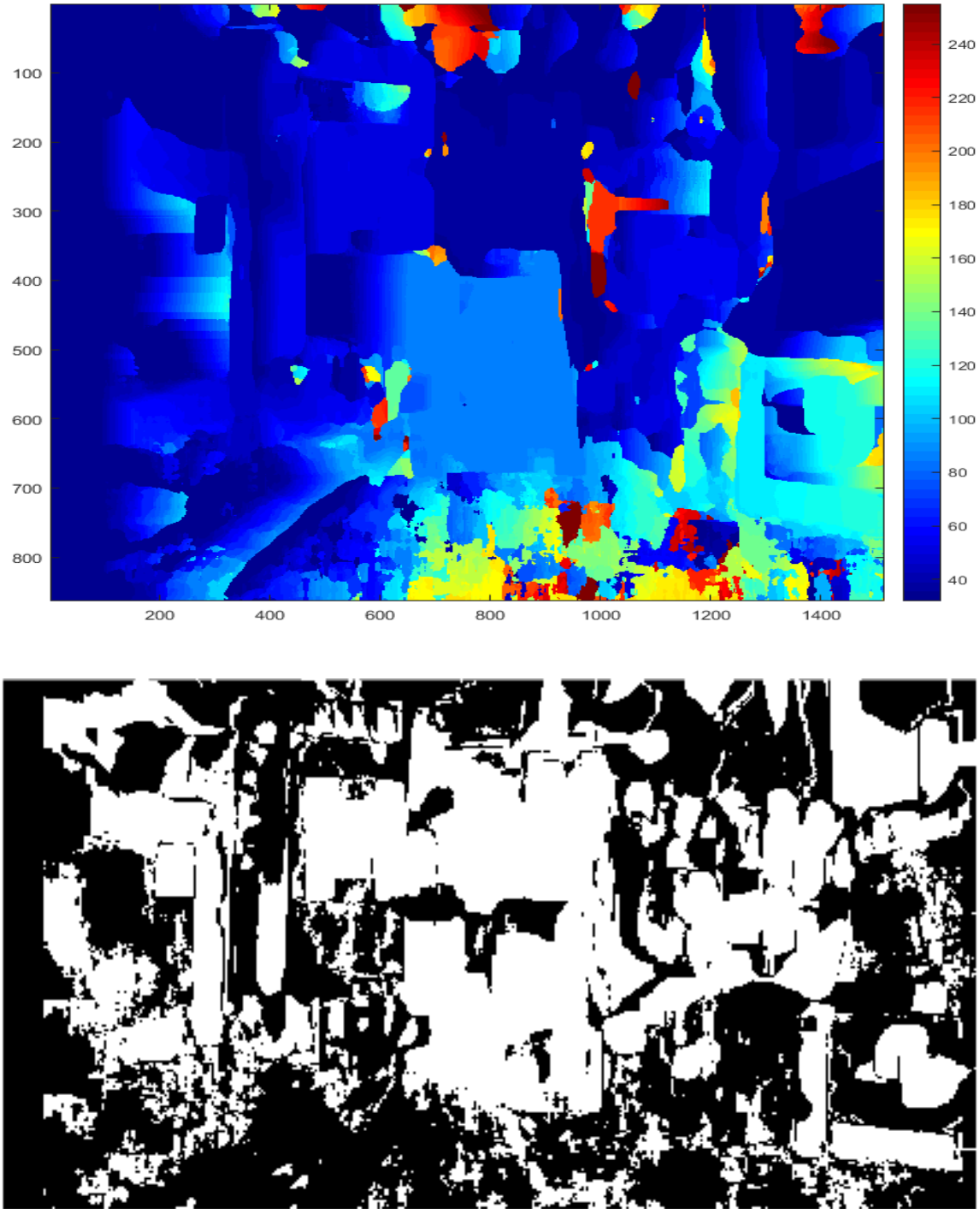


Figure 5-21 Top image: estimated disparity map from eight paths BM on NonIR camera image (disparity range normalized to 1 to 256, R/L check not yet applied); Bottom image: R/L check mask, white regions represent valid matches.

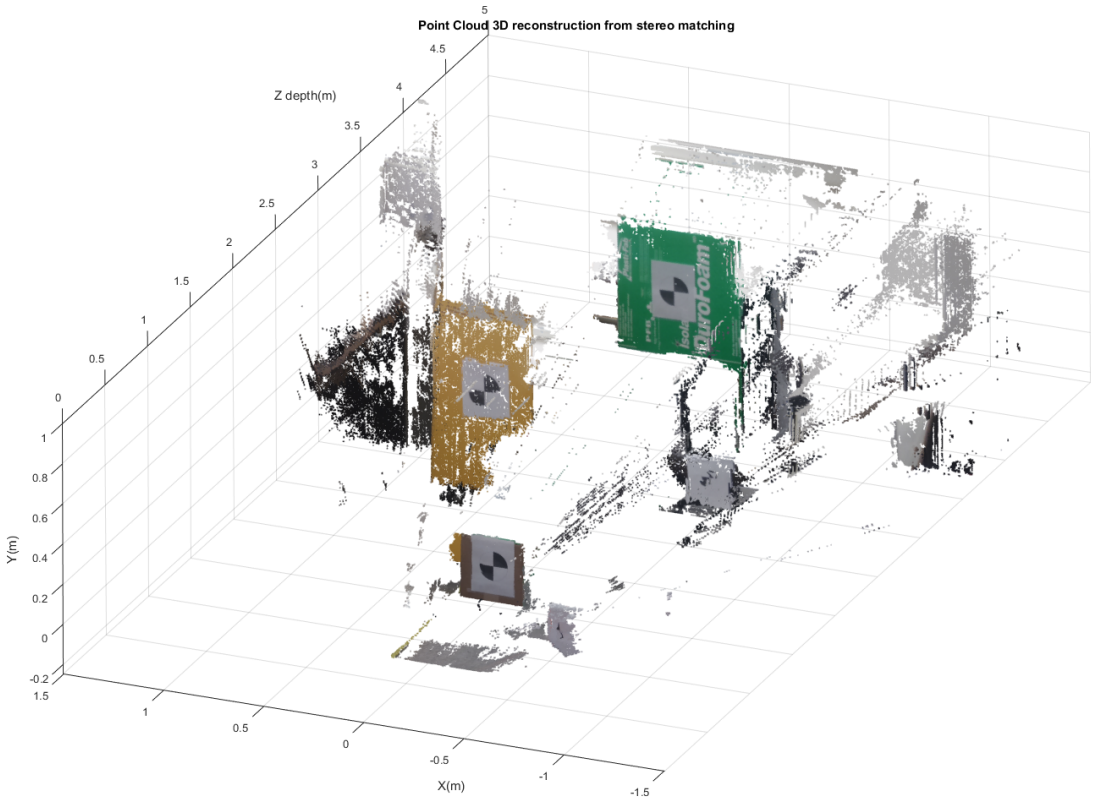


Figure 5-22 A 3D rendered reconstruction from eight paths SGM on NonIR camera image

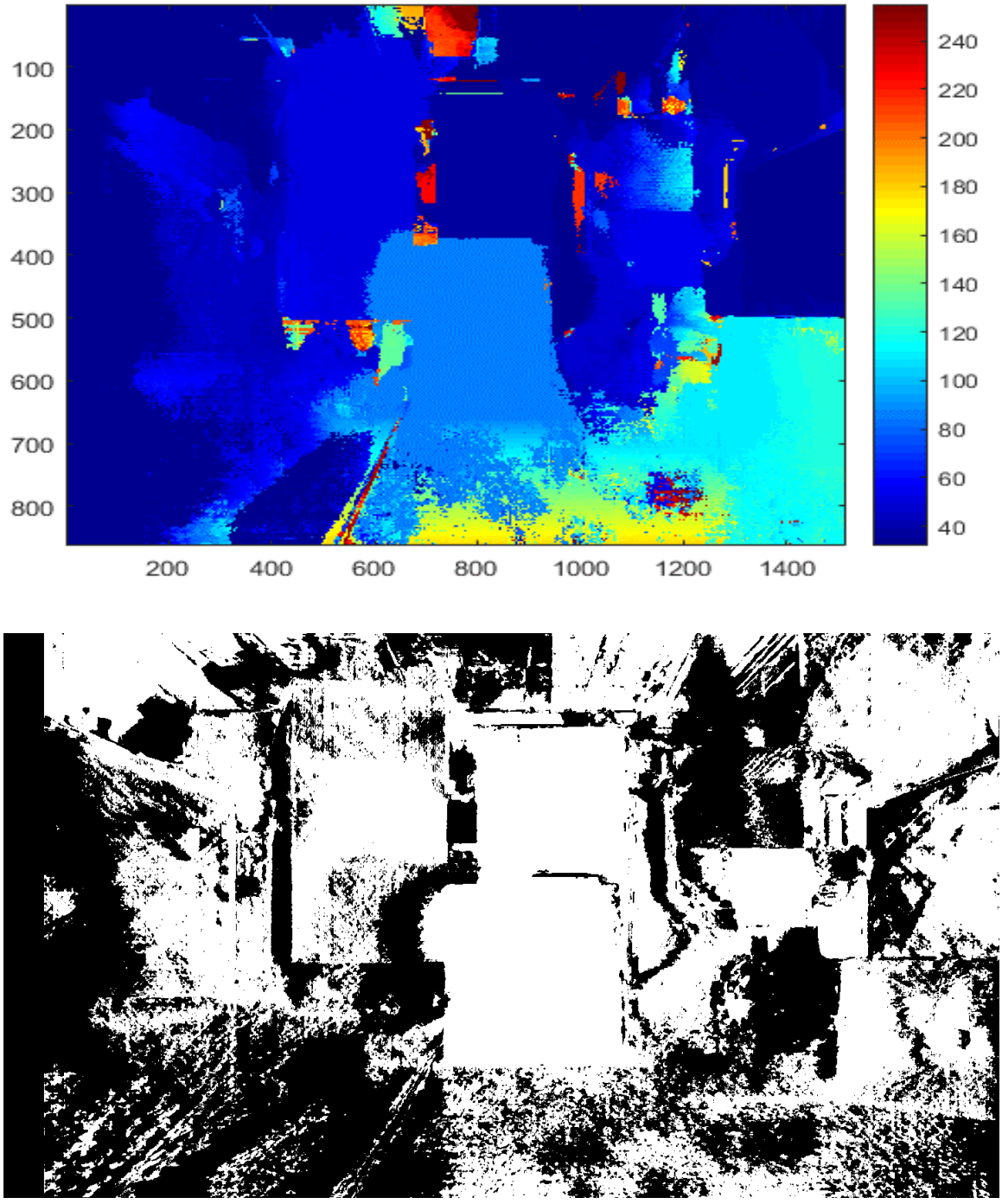


Figure 5-23 Top image: estimated disparity map from SGM method on NonIR camera (disparity range normalized to 1 to 256, R/L check not yet applied); Bottom image: R/L check mask, white regions represent valid matches.

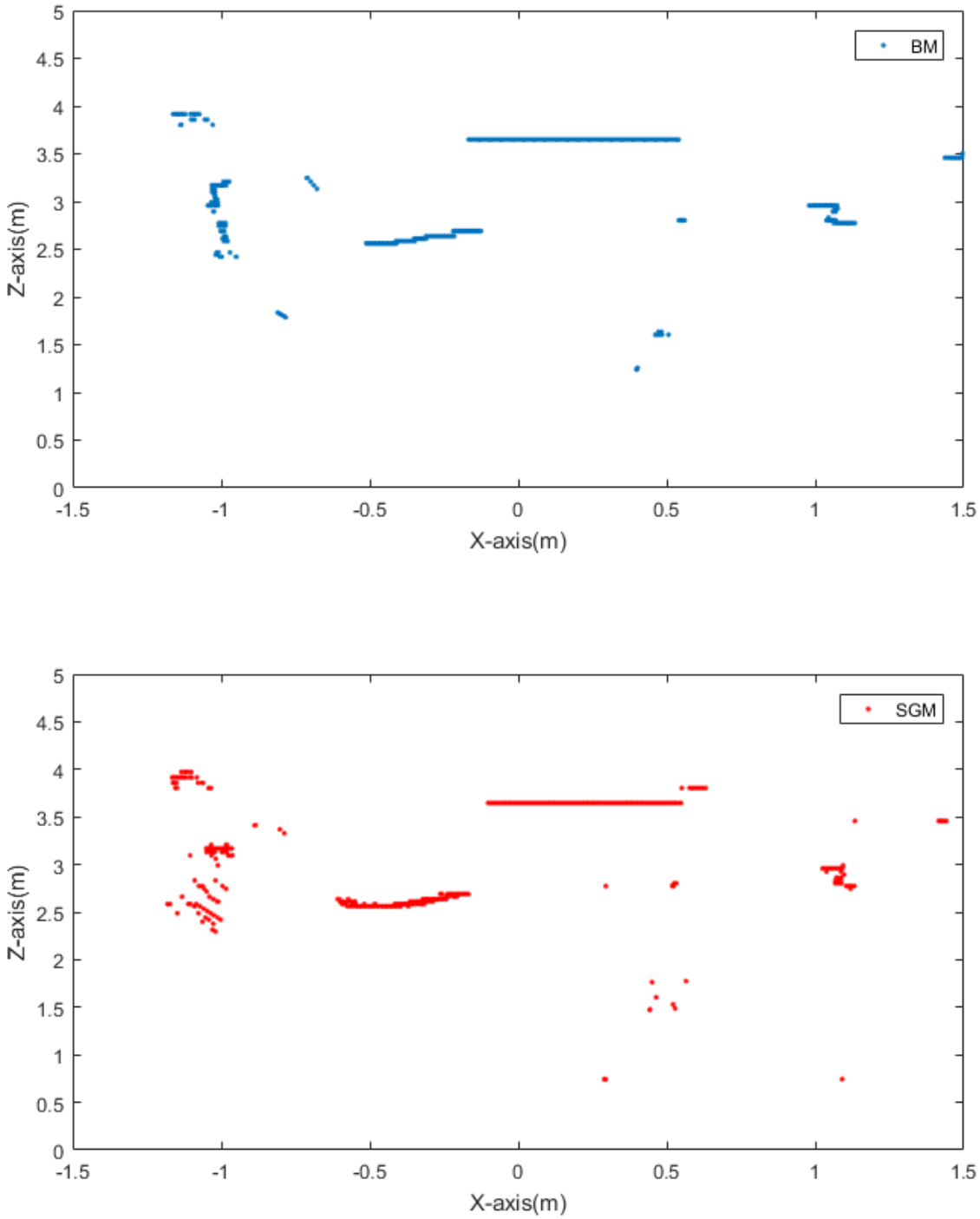


Figure 5-24 2D reconstruction using the results from BM and SGM depth estimations on NonIR images at horizontal plane of marker 1

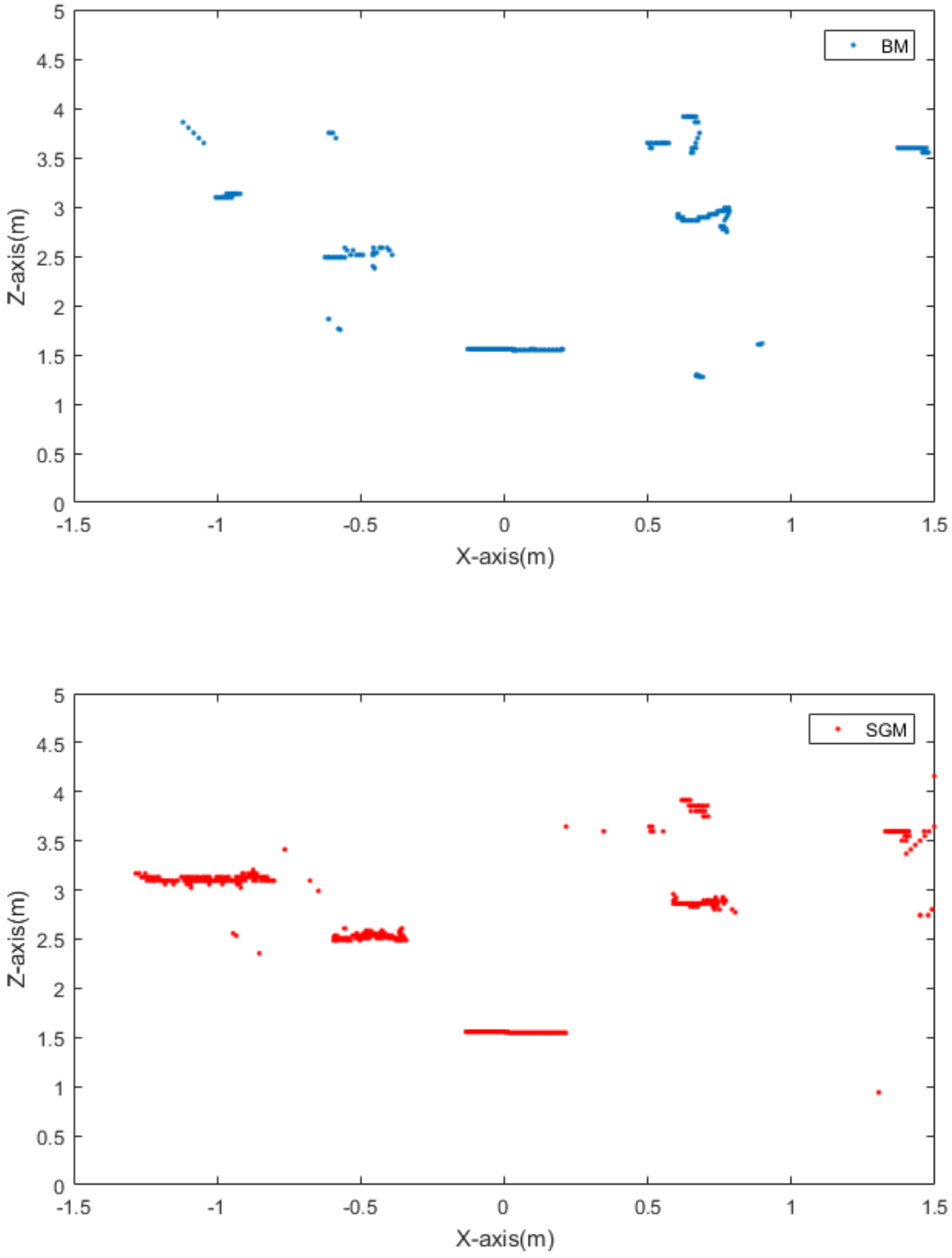


Figure 5-25 2D reconstruction using the results from BM and SGM depth estimations on NonIR images at horizontal plane of marker 2

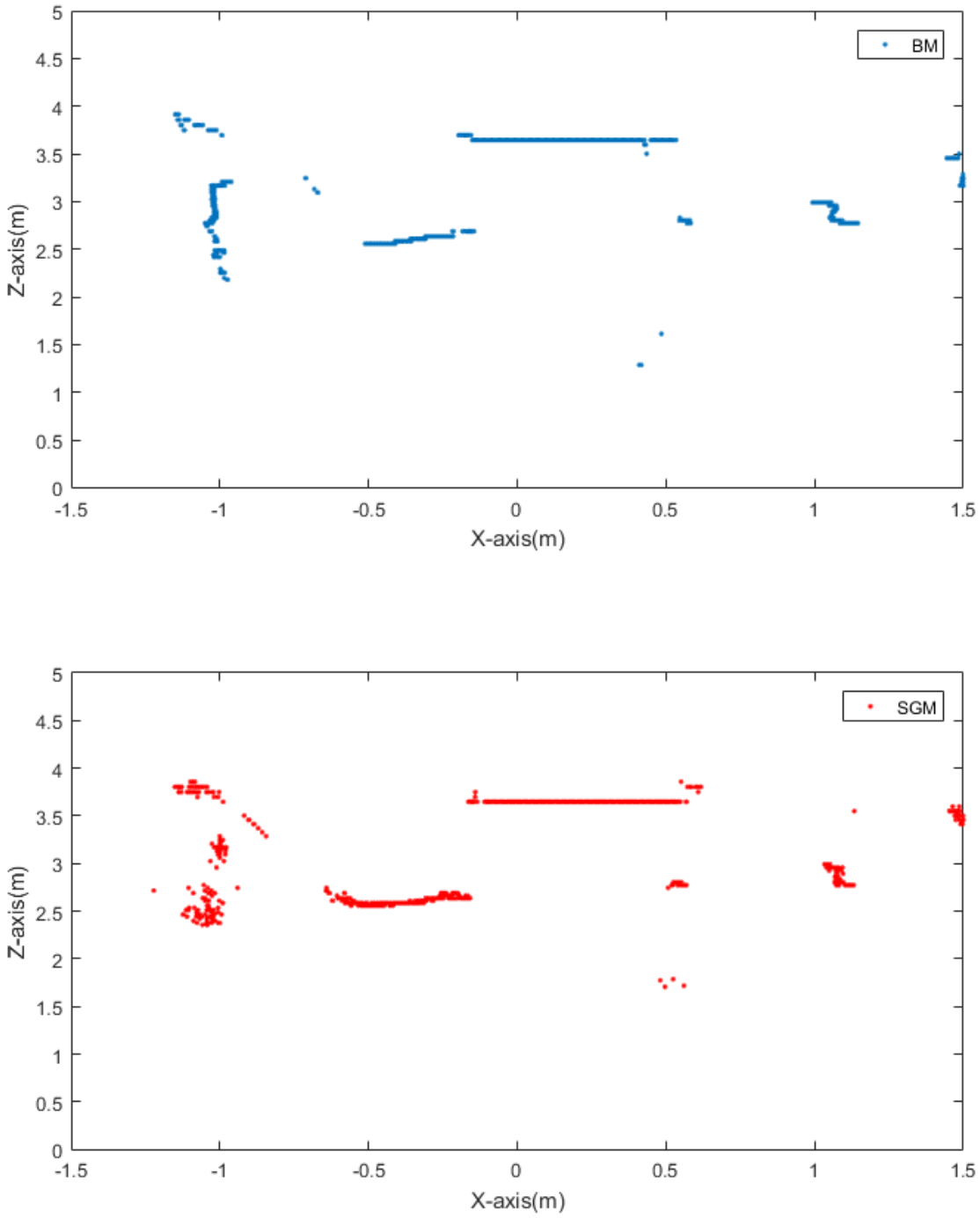


Figure 5-26 2D reconstruction using the results from BM and SGM depth estimations on NonIR images at horizontal plane of marker 3

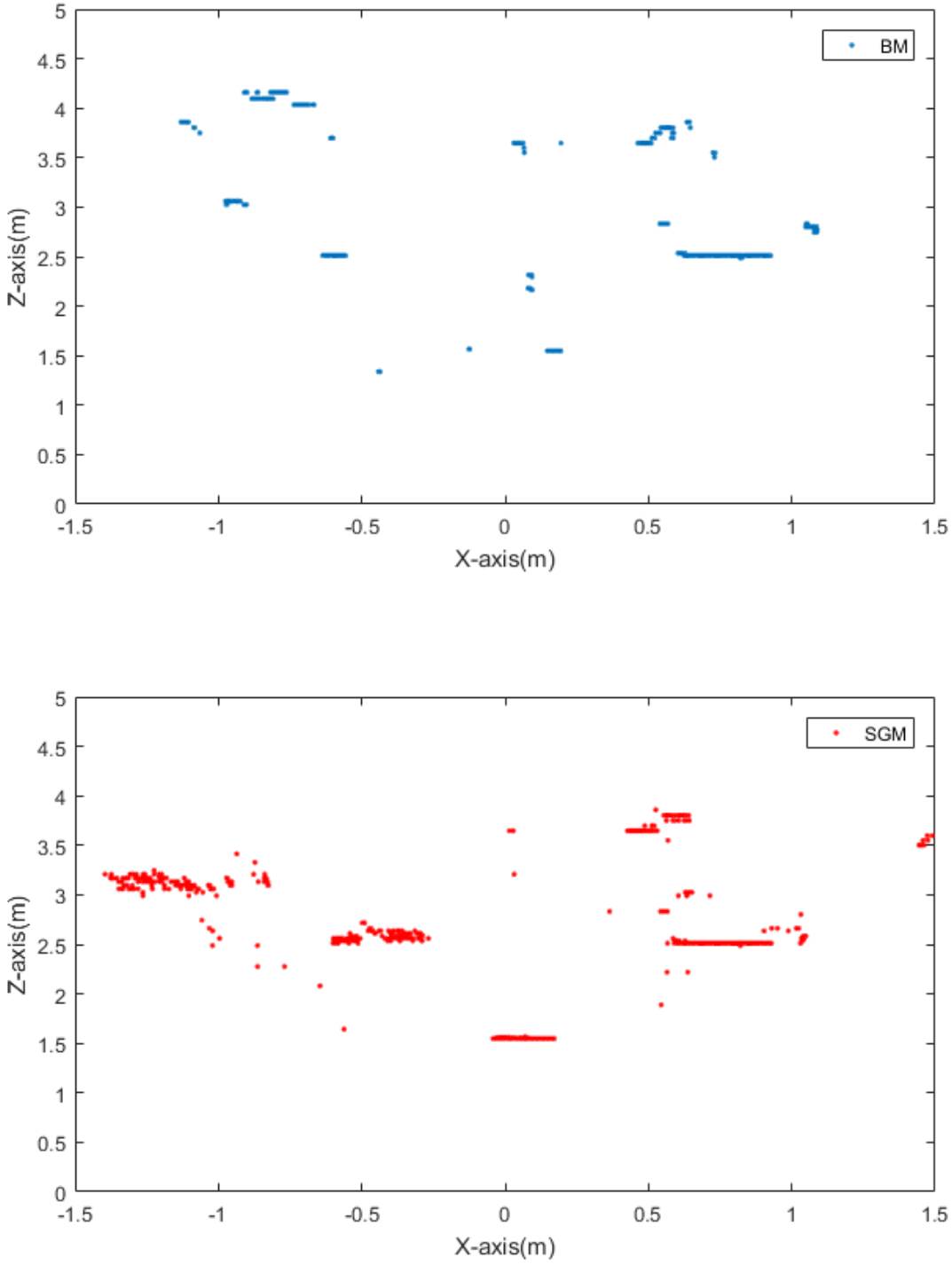


Figure 5-27 2D reconstruction using the results from BM and SGM depth estimations on NonIR images at horizontal plane of marker 4

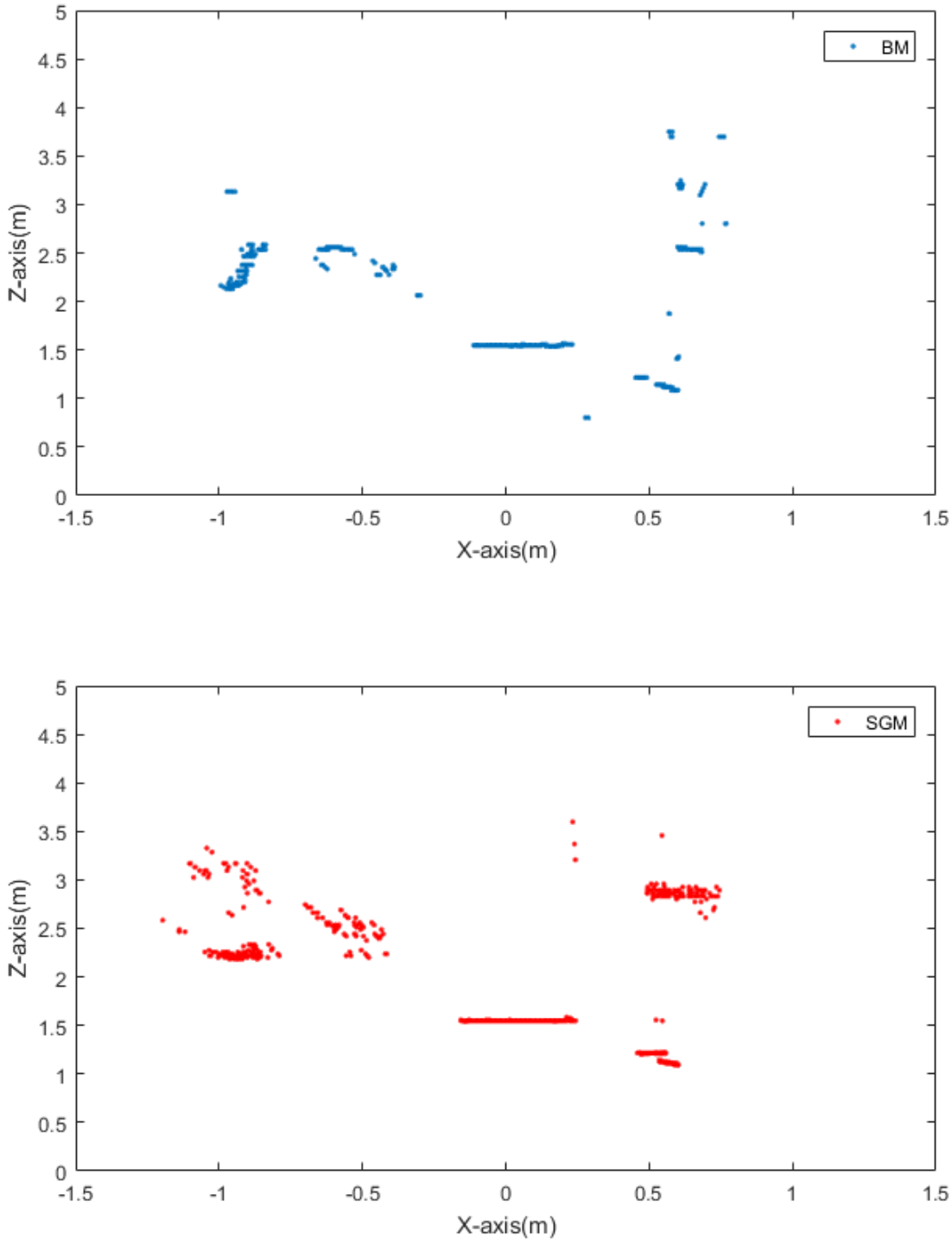


Figure 5-28 2D reconstruction using the results from BM and SGM depth estimations on NonIR images at horizontal plane of marker 5

5.4.6 Computational complexity

The code for BM and SGM are both developed in a Matlab environment, which is not the optimal solution to process the image data. The elapse time for each method is documented in Table 5-8. Note that NonIR images have higher resolution on rectified images than the images with an IR filter, resulting in longer processing time. In each test case, each image went through the workflow outlined in Figure 5-1 once, and thus, the processing time will be doubled for a pair of stereo images. Table 5-8 shows the results of processing a pair of stereo images so that the two disparity maps can be used for a L/R consistency check in the refinement stage. Also, the code in Matlab is not fully optimized. The tests were conducted on an Intel i7-5820K 6C12T, 64GB memory, Windows7 computer with Matlab2016b. Moreover, six workers were used in Matlab for SGM parallel processing.

Table 5-8 Elapsed calculation time

	Images without IR filter (HR)			Images with IR filter (HR)		
	BM method	SGM 8 paths	SGM 16 paths	BM method	SGM 8 paths	SGM 16 paths
Elapse time (seconds)	231.90	4075.90	6658.83	215.88	3856.46	5799.77

5.5 Discussion of 2D/3D triangulation by using dense stereo matching method in real world data.

The BM method and SGM method are investigated throughout this chapter. From the results obtained using Middlebury stereo datasets in Table 5-3 and Table 5-6, it is observed that SGM is outperformed in the structured environments and scenes with repeated patterns than the BM method. On the other hand, the BM method is sensitive to window size, which will become a practical problem when the robot is working in the real world environment.

Thus, advanced algorithms that adjust window sizes based on scenes may need to be used. Additionally, from the results of SGM, it seems that SGM is insensitive to the penalty value selections, as long as the small penalty P_1 is sufficiently small and the maximum value of large adaptive penalty P_2 is less than P_1 . Thus $P_2^{'}$ can be a number slightly greater than the *maximum intensity variation* (usually 256 in grayscale image) $\times P_1$.

In the real world data, from visual inspection, BM method works surprisingly better than the Middlebury stereo datasets. It can be seen in Table 5-7, that the BM and SGM yielded similar results within centimeter accuracy. However, the results in Table 5-7 only represent point features, and the accurate results further prove the importance of camera calibration. Results that are more interesting can be found from the 3D reconstructions of the scene and L/R check masks from Figure 5-20 to Figure 5-23. SGM performs reasonably well in the homogeneous surfaces and slant surfaces, since it takes advantage of using the smoothness term in the global energy function. In SGM, a small region on the top left of the box (marker 2) is marked as valid. However, that region is an occluded area, which means no match should be found around that region. This mismatch may be caused by the special positioning of the two objects, which are combined occlusions and discontinuities on one edge. While in 3D reconstructions, the results from BM is cruder than those from the SGM. In Figure 5-21, many mismatches are excluded from L/R consistency check mask and result in holes and cuts in the image. The BM method achieves better matching results in the texture rich regions on the insulating foam than the texture-less regions in the trolley. In terms of L/R check mask, comparing Figure 5-21 and Figure 5-23, it clearly shows that

SGM surpasses the BM. The mask generated by SGM distinctly outlines the edges and large surfaces.

However, these six figures are mainly for visualization. The binary maps that are created from disparity maps from 16 paths SGM scanning and calculated by Equation 2.5 are shown in Figure 5-24 to Figure 5-28. These five figures represent five horizontal slices that cut through the five different markers. As it can be seen in all of the slices, both matching methods successfully located the fronto-parallel surfaces. The SGM is better on slant surfaces, and it provides denser point clouds in Figure 5-26 and Figure 5-28 at marker 3 and marker 5, respectively. Figure 5-27 is a special case where the horizontal slice of marker 4 is near the top surface of the box of marker 2. The SGM was able to locate the discontinuity. In both methods, the chair with highly dense repetitive patterns is not recognizable. The BM that has larger window sizes will result in mismatches on that area, and L/R check will exclude these mismatches. Similarly, SGM also excludes a significant amount of mismatch on the chair backrest. Considering the real world data, after applying the L/R check mask, there are still some noise or matches appear in the 2D map. Finally, it can be expected that the BM and SGM will have similar basic depth estimation capability in the real world environment, but SGM outperforms BM in texture-less areas and distinguishing edges. For surfaces with fine repetitive patterns, both methods will not work correctly, and only partial fronto-parallel surface is identified.

Other than performance results, the SGM is much more difficult to implement than the BM. Many vectorization optimization techniques are applied in the Matlab code, and it requires more memory storage space (width × length × disparity range) to store at least eight paths during the parallel processing (Matlab Parallel Processing Toolbox™). A high computational cost is the main reason that the full resolution images captured by Raspberry Pi camera modules are not used in this thesis work. As illustrated in Table 5-7, the BM method took significantly less time than SGM, and it can be considered to be a tradeoff between performance and calculation time cost. In most cases, eight aggregation paths are sufficient as shown in Figure 5-29. At the center of marker 2 (816,495) of the NonIR lab image, eight paths are adequate to prove that the minimum cost occurred at disparity level 171.

In conclusion, both the BM method and SGM are capable of providing accurate 2D/3D depth measurements after stereo image triangulation, and they will work in general purpose when the illumination in the working environment is sufficient, however some features or objects are only partially recognizable in both methods. BM is more suitable for a resource-limited system, while SGM is more robust to work on a texture-less region and areas where there are many discontinuities presented. The computational cost of SGM is much higher than BM.

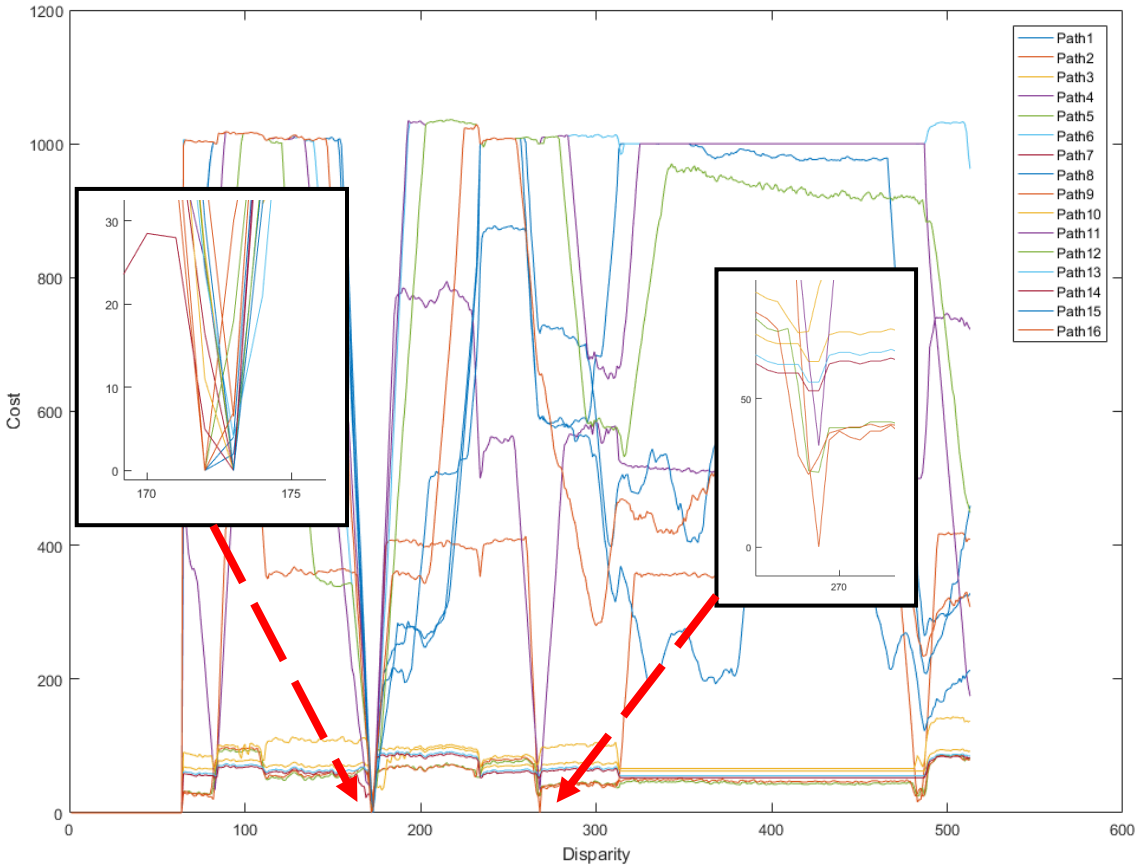


Figure 5-29 Eight paths versus sixteen paths in SGM on NonIR lab image

One solution is to process only one strip of the image to the robot. Thus, a 2.5D space can be reconstructed. Secondly, the image can be further downsampled, but this may affect the resolution of matching results. As shown in Figure 5-30, the images are down-sampled by 50% and surfaces at a distance start to lost accuracy. Estimations that are more uncertain are observed.

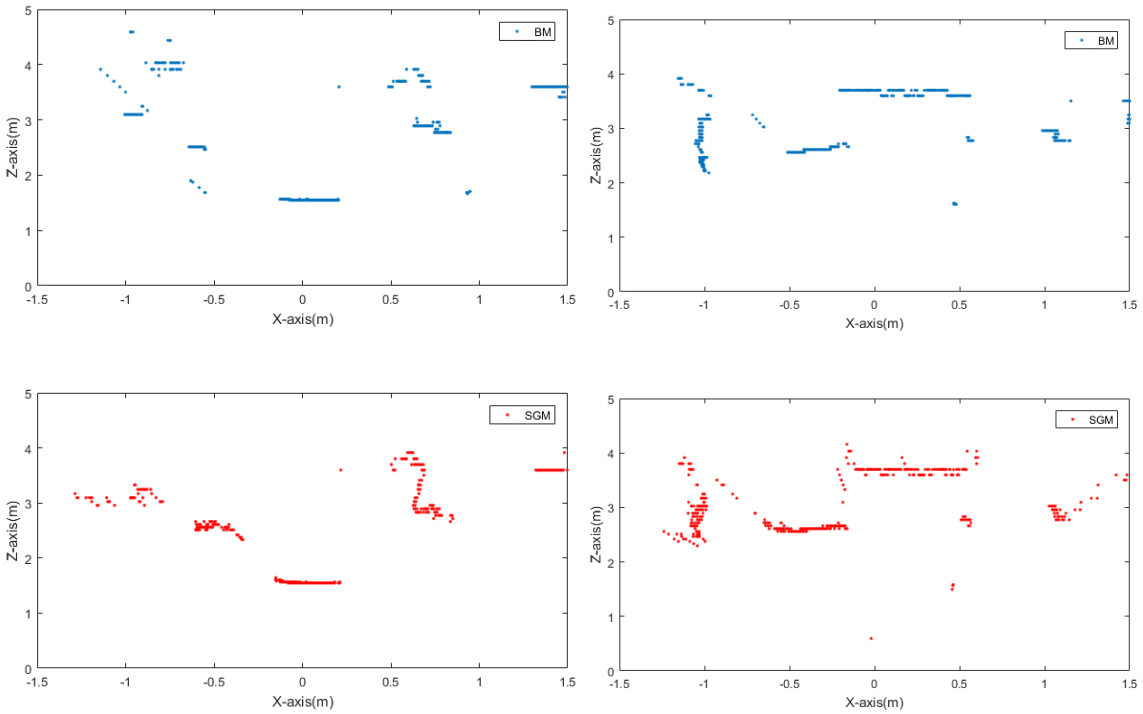


Figure 5-30 2D reconstructions of down-sampled NonIR image (by 50%) on the horizontal plane of marker 2 (left) and marker 3 (right)

Chapter 6. Future work

The future research work can address the following aspects:

1) In this thesis, the software-based SGM is implemented in Matlab. For future work, the most important task is to investigate the real-time hardware-based SGM solution.

2) In this thesis, the stereo camera rig is sufficient to capture images in the static scene, which means that the robot is in a stationary condition for stereo matching algorithm testing and evaluation. Alternatively, the video mode of the camera modules shall be tested and possible frame and triggering synchronization problems need to be identified. If the current Raspberry Pi camera module setup is unable to work in the real-time environment, a new stereo camera rig needs to be built.

3) It is possible to combine local features with a dense disparity map to create a continuous 3D scenes reconstruction.

4) A high-resolution 3D laser scanner must be acquired for the comparison of vison-based depth estimations and the ground truth measurements.

Chapter 7. Conclusion

In the first part of this thesis, a hardware framework, mainly based on open-source components, is proposed for indoor robotics research. The kinematics of the differential drive system are developed with hardware implementation by using the proportional-integral controller on the Arduino 2560 and the L298P motor controller. High-speed and low-speed communication are established using 802.11AC bridged routers and Xbee modules. Two Raspberry Pis, each one with a camera module, act as a stereo image acquisition device. It can be concluded that it is possible to use open-source components in the research work due to their modular design, ease of programming, wide availability and inexpensive price. However, advanced applications such as those in image and video domains, especially in real-time applications, open-source parts may not be preferred because they commonly lack support and have system complexity. Regarding software, the idea of using a hypervisor in the onboard computer is introduced. This is because of its high versatility in the management of multiple operation systems.

The rest of the thesis work focuses on building a disparity map and stereo vision triangulation for indoor robotic applications. The goal is to investigate the feasibility of using cheap stereo cameras as range sensors for indoor robots. The process can be described as identifying the corresponding points in two images and retrieving their displacements to reconstruct the geometry of the scenes as a depth map. This is one of the most challenging and fundamental problems in computer vision. Four stereo matching methods are studied, implemented and tested using Middlebury stereo datasets and lab images that are captured by Raspberry Pi camera modules. The lab images are then

calibrated and rectified by the Matlab Computer Vision Toolbox. Two feature-based matching algorithms, that are capable of estimating sparse disparity maps, are not advised since they only provide sparse point cloud maps. Then other two dense matching methods are further explored. The vectorized window-based block matching method and the vectorized semi-global matching method are proposed, implemented and tested on the Middlebury stereo datasets, with the ground truth provided and lab images to emulate real world conditions. The advantage of semi-global matching method, over other global iterative optimization methods, is that semi-global matching method has deterministic runtime and it runs recursively. It is concluded that both of the algorithms will provide basic disparity maps in the indoor enviromment, however some features or objects are partially recogniziable in the map. The semi-global matching method outperforms in texture-less regions, and it is more capable of detecting disparity differences near the edges due to its adaptive penalty system that enforces local smoothness in the global energy function. The tradeoff between accuracy and computational complexity must also be considered. The semi-global matching method has the tendency to consume more resources than the block matching method, and the possible solutions are proposed as well. These include hardware acceleration or reducing the search space. In terms of depth estimations, the infrared blocked and non-blocked Raspberry Pi camera modules yielded comparable results.

Inferring information about three dimensions from two-dimensional images is a non-trivial task, and it will continue to be one of the most challenging research problems in computer vision and even in robotic science.

List of References

[1] O. Fouad, "Dynamic Mapping And Positioning Using Stereo Panoramic Cameras," M.A.Sc. thesis, Carleton University, Canada, 2014.

[2] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Cambridge, Massachusetts: MIT Press, 2004.

[3] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping (SLAM): Part I," *IEEE Robotics and Automation Magazine*, vol. 13, no. 6, pp. 99–108, 2006.

[4] T. Bailey and H. Durrant-Whyte, "Simultaneous Localization and Mapping (SLAM) Part II," *IEEE Robotics and Automation Magazine*, vol. 13, no. 9, pp. 108–117, 2006.

[5] U. Pilz, W. Gropengießer, F. Walder, J. Witt, and H. Werner, "Quadrocopter localization using RTK-GPS and vision-based trajectory tracking," in *Proceedings of the International Conference on Intelligent Robotics and Applications*, Aachen, Germany, 2011, pp. 12–21.

[6] C. Forster, M. Faessler, F. Fontana, M. Werlberger, and D. Scaramuzza, "Continuous on-board monocular-vision-based elevation mapping applied to autonomous landing of micro aerial vehicles," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Seattle, WA, USA, 2015, pp. 111–118.

[7] A. Al-kaff, Q. Meng, and D. Mart, "Monocular vision-based obstacle detection / avoidance for unmanned aerial vehicles," in *Proceedings of the Intelligent Vehicles*

Symposium, Gothenburg, Sweden, 2016, no. 4, pp. 3–8.

[8] N. Fioraio and K. Konolige, "Realtime visual and point cloud SLAM," in *Proceedings of the RGB-D workshop on advanced reasoning with depth cameras at robotics: Science and Systems Conference*, Los Angeles, CA, USA, 2011.

[9] A. Huang, N. Roy, A. Bachrach, P. Henry, M. Krainin, D. Maturana, and D. Fox, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Proceedings of the International Symposium on Robotics Research*, Flagstaff, AZ, USA, 2011, vol. 2, pp. 235–252.

[10] J. Garcia and Z. Zalevsky, "Range mapping using speckle decorrelation," U.S. Patent 7,433,024 B2, 2008.

[11] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremer, and W. Burgar, "An evaluation of the RGB-D SLAM system," in *Proceedings of the IEEE International Conference on Robotics and Automation*, St Paul, MN, USA, 2012, pp. 1691–1696.

[12] K. Litomisky, "Consumer RGB-D Cameras and their applications," Rep., University of California, Riverside, 2012.

[13] W. Förstner and B. P. Wrobel, *Photogrammetric Computer Vision - Statistics, Geometry, Orientation and Reconstruction*. Switzerland: Springer International Publishing, 2016.

[14] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*. 1993.

[15] M. Adler, T. Boutell, and J. Bowler, "Portable Network Graphics (PNG)

Specification (Second Edition) Information technology — Computer graphics and image processing — Portable Network Graphics (PNG): Functional specification . ISO / IEC 15948 : 2003 (E),” 2003. [Online]. Available: https://www.w3.org/TR/PNG. [Accessed: 03-Oct-2016].

[16] O. Marques, *Practical Image and Video Processing Using MATLAB*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2011.

[17] D. Jones, “Picamera 1.12 documentation,” 2016. [Online]. Available: https://picamera.readthedocs.io/en/release-1.12/. [Accessed: 03-Oct-2016].

[18] D. Scharstein, R. Szeliski, and H. Hirschmüller, “Middlebury Stereo Datasets,” 2016. [Online]. Available: http://vision.middlebury.edu/stereo/data/. [Accessed: 03-Oct-2016].

[19] Mathworks, “Matlab convert RGB image or colormap to grayscale.” [Online]. Available: https://www.mathworks.com/help/matlab/ref/rgb2gray.html. [Accessed: 03-Oct-2016].

[20] P.K.Sinha, *Image Acquisition and Preprocessing for machine vision systems*. Bellingham, Washington: Society of Photo-Optical Instrumentation Engineers, 2012.

[21] SONY, “IMX219PQ Product information,” 2016. [Online]. Available: http://www.sony-semicon.co.jp/products_en/new_pro/april_2014/imx219_e.html. [Accessed: 03-Oct-2016].

[22] E. Trucco and A. Verri, *Introductory Techniques for 3D Computer Vision*. Upper

Saddle River, New Jersey: Prentice-Hall International, 1998.

[23] J. Torres-moreno, “Differential drive control system design for the intelligent vacuum robot project,” Rep., Carleton University, 2013.

[24] C. Wheeler, “Control algorithm implementation using arduino hardware for the intelligent robotic vacuum project,” Rep., Carleton University, 2014.

[25] K. Shojaei, A. M. Shahri, A. Tarakameh, and B. Tabibian, “Adaptive trajectory tracking control of a differential drive wheeled mobile robot,” *Robotica*, vol. 29, no. 3, pp. 391–402, 2016.

[26] U. of Michigan, “Control Tutorials for MATLAB and Simulink - Motor System,” 2016. [Online]. Available: http://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SystemModeling. [Accessed: 01-Oct-2016].

[27] B. Beauregard, “Arduino PID library,” 2011. [Online]. Available: http://brettbeauregard.com/blog/category/pid/coding/. [Accessed: 05-Oct-2016].

[28] R. Li, M. Pang, C. Zhao, G. Zhou, and L. Fang, “Monocular long-term target following on UAVs,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition Workshops*, Las Vegas, NV, USA, 2016, pp. 29–37.

[29] E. Eaton, C. Mucchiani, M. Mohan, D. Isele, J. M. Luna, and C. Clingerman, “Design of a low-cost platform for autonomous mobile service robots,” in *Proceedings of the Workshop on Autonomous Mobile Service Robots*, New York, NY, USA, 2016.

[30] E. Fouad, T. Okamoto, F. Zhou, J. Orenstein, K. Agarwal, T. Pailevanian, and J. Larson, "Design and implementation of an integrated, autonomous underwater vehicle: Dory," Rep., California Institute of Technology, 2016.

[31] "Raspberry GPIO Pinout Diagram." [Online]. Available: https://miniboard.com.ua/img/cms/raspberry_pi_circuit_note_fig2a.jpg. [Accessed: 10-Oct-2016].

[32] Python Software Foundation, "RPi.GPIO 0.6.3 A module to control Raspberry Pi GPIO channels." [Online]. Available: https://pypi.python.org/pypi/RPi.GPIO. [Accessed: 10-Oct-2016].

[33] MINI-BOX, "DCDC-NUC, 6-48V automotive power supply for NUC, 12V or 19V output," 2016. [Online]. Available: http://www.mini-box.com/DCDC-NUC. [Accessed: 10-Oct-2016].

[34] J. J. Wolken, *Light Detectors, Photoreceptors, and Imaging Systems in Nature*. New York: Oxford University Press, 1995.

[35] P. Servos and M. Goodale, "Binocular vision and the on-line control of human prehension," *Exp. brain Res.*, vol. 98, no. 1, pp. 119–27, 1994.

[36] A. J. Parker, "Binocular depth perception and the cerebral cortex," *Nat. Rev. Neurosci.*, vol. 8, no. 5, pp. 379–391, 2007.

[37] A. Yonas and C. E. Granrud, "The development of sensitivity to kinetic, binocular, and pictorial depth information in human infants," *Brain Mechanisms and Spatial Vision*. pp. 113–145, 1985.

[38] B. Timney, "Development of binocular depth perception in kittens," *Investig. Ophthalmol. Vis. Sci.*, vol. 21, no. 3, pp. 493–496, 1981.

[39] C. Hendrix and W. Barfield, "Relationship between monocular and binocular depth cues for judgements of spatial information and spatial instrument design," *Displays*, vol. 16, no. 3, pp. 103–113, 1995.

[40] J. R. Jensen, *Remote Sensing of the Environment, An Earth Resource Perspective*. Upper Saddle River, New Jersey: Prentice-Hall, Inc., 2007.

[41] N. Lazaros, G. C. Sirakoulis, and A. Gasteratos, "Review of stereo vision algorithms: from software to hardware," *Int. J. Optomechatronics*, vol. 2, no. 4, pp. 435–462, 2008.

[42] S. Barnard and M. Fischler, "Computational stereo," *ACM Comput. Surv.*, vol. 14, no. 4, pp. 553–572, 1982.

[43] R. Szeliski, *Computer Vision : Algorithms and Applications*. London: Springer-Verlag, 2010.

[44] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986.

[45] H. H. Baker, "Depth from edge and intensity based stereo," Rep. No. STAN-CS-82-930, Standford University, 1982.

[46] G. Medioni and R. Nevatia, "Segment-based stereo matching," *Comput. Vision, Graph. Image Process.*, vol. 31, no. 1, pp. 2–18, 1985.

[47] V. Venkateswar and R. Chellappa, "Hierarchical stereo and motion correspondence

using feature groupings," *Int. J. Comput. Vis.*, vol. 15, no. 3, pp. 245–269, 1995.

[48] J. Tian, N. Thacker, and A. Stancu, "Simulation of wire-frame object representations for a 3D robotic vision system," Memo. No. 2015-007, Medical School, University of Manchester, 2015.

[49] J. Tian, "Simulation of a 3D vision-based robotic system simulation of a 3D vision-based robotic system," Memo. No. 2015-004, Medical School, University of Manchester, 2015.

[50] D. G. Lowe, "Distinctive image features from scale invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[51] I. Hadda and J. Knani, "Global mapping and localization for mobile robots using stereo vision," in *Proceedings of the International Multi-Conference on Systems, Signals & Devices*, Castelldefels-Barcelona, Spain, 2013, pp. 1–6.

[52] B. Charmette, E. Royer, and F. Chausse, "Vision-based robot localization based on the efficient matching of planar features," *Mach. Vis. Appl.*, vol. 27, no. 4, pp. 415–436, 2016.

[53] K. Sharma, K. Jeong, and S. Kim, "Vision based autonomous vehicle navigation with self-organizing map feature matching technique," in *Proceedings of the International Conference on Control, Automation and Systems*, Singapore, 2011, pp. 946–949.

[54] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vis.*, vol. 47, no. 1–3, pp. 7–42,

2002.

[55] R. K. Gupta and S. Y. Cho, “A correlation-based approach for real-time stereo matching,” in *Proceedings of the International Symposium on Visual Computing*, Las Vegas, NV, USA, 2010, pp. 129–138.

[56] M. Z. Brown, D. Burschka, and G. D. Hager, “Advances in computational stereo,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 993–1008, 2003.

[57] M. J. Hannah, “Computer Matching of Areas in Stereo Images,” Memo. STAN-CS-74-438, Standford University, 1974.

[58] T. Kanade and M. Okutomi, “A stereo matching algorithm with an adaptive window: theory and experiment,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 9, pp. 920–932, 1994.

[59] A. Miron, S. Ainouz, A. Rogozan, and A. Bensrhair, “A robust cost function for stereo matching of road scenes,” *Pattern Recognit. Lett.*, vol. 38, pp. 70–77, 2014.

[60] L. Wang, M. Gong, M. Gong, and R. Yang, “How far can we go with local optimization in real-time stereo matching - A performance study on different cost aggregation approaches,” in *Proceedings of the 3rd International Symposium on 3D Data Processing, Visualization, and Transmission*, Chapel Hill, NC, USA, 2006, pp. 129–136.

[61] R. K. Gupta and S. Cho, “Window-based approach for fast stereo correspondence,” *IET Comput. Vis.*, vol. 7, no. 2, pp. 123–134, 2013.

[62] H. Hirschmuller and D. Scharstein, “Evaluation of stereo matching costs on images

with radiometric differences," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 9, pp. 1582–1599, 2009.

[63] R. Zabih and J. Woodfill, "Non-parametric Local Transforms for Computing Visual Correspondence," in *Proceedings of the European conference on computer vision*, Stockholm, Sweden, 1994, pp. 151–158.

[64] J. Banks, "Quantitative evaluation of matching methods and validity measures for stereo vision," *Int. J. Rob. Res.*, vol. 20, no. 7, pp. 512–532, 2001.

[65] R. A. Hamzah and H. Ibrahim, "Literature survey on stereo vision disparity map algorithms," *J. Sensors*, 2016.

[66] S. Birchfield and C. Tomasi, "A pixel dissimilarity measure that is insensitive to image sampling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 4, pp. 401–406, 1998.

[67] C. Tomasi and S. Birchfield, "Depth discontinuities by pixel-to-pixel stereo," *Int. J. Comput. Vis.*, vol. 35, no. 3, pp. 269–293, 1998.

[68] H. Hirschmüller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, CA, USA, 2005, pp. 807–814.

[69] C. Lawrence Zitnick and T. Kanade, "A cooperative algorithm for stereo matching and occlusion detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 7, pp. 675–684, 2000.

[70] G. Li and S. W. Zucker, "Differential geometric consistency extends stereo to curved surfaces," in *Proceedings of the European Conference on Computer Vision*, Graz, Austria, 2006, pp. 44–57.

[71] K. Mühlmann, D. Maier, R. Hesser, and R. Männer, "Calculating dense disparity maps from color stereo images, an efficient implementation," *Int. J. Comput. Vis.*, vol. 47, no. 1–3, pp. 79–88, 2002.

[72] O. Veksler, "Fast variable window for stereo correspondence using integral images," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Madison, WI, USA, 2003, pp. 556–561.

[73] J. Lu, G. Lafruit, and F. Catthoor, "Anisotropic local high-confidence voting for accurate stereo correspondence," in *Proceedings of the SPIE Electronic Imaging conference on Image Processing: Algorithms and Systems*, San Jose, CA, USA, 2008, p. 68120J.

[74] K. J. Yoon and I. S. Kweon, "Adaptive support-weight approach for correspondence search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 650–656, 2006.

[75] Z. Gu, X. Su, Y. Liu, and Q. Zhang, "Local stereo matching with adaptive support-weight, rank transform and disparity calibration," *Pattern Recognit. Lett.*, vol. 29, no. 9, pp. 1230–1235, 2008.

[76] F. Tombari, S. Mattoccia, and L. Stefano, "Segmentation-based adaptive support for accurate stereo correspondence," in *Proceedings of the Pacific-Rim Symposium on Image and Video Technology*, Santiago, Chile, 2007, pp. 427–438.

[77] F. Tombari, S. Mattoccia, L. Di Stefano, and E. Addimanda, "Classification and evaluation of cost aggregation methods for stereo correspondence," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, USA, 2008, pp. 1–8.

[78] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for Markov random fields with smoothness-based priors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 6, pp. 1068–1080, 2008.

[79] O. Veksler, "Efficient graph-based energy minimization methods in computer vision," Ph.D. thesis, Cornell University, 1999.

[80] J. Besag, "Spatial interaction and the statistical analysis of lattice systems," *J. R. Stat. Soc. - Ser. B*, pp. 192–236, 1974.

[81] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1222–1239, 2001.

[82] J. Sun, H.-Y. Shum, and N.-N. Zheng, "Stereo matching using belief propagation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 7, pp. 787–800, 2003.

[83] M. F. Tappen and W. T. Freeman, "Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters," in *Proceedings of the 9th IEEE International Conference on Computer Vision*, Nice, France, 2003, pp. 900–906.

[84] S. Roy and I. J. Cox, “A maximum-flow formulation of the N-camera stereo correspondence problem,” in *Proceedings of the 6th International Conference on Computer Vision*, Santorini, Greece, 1998, pp. 492–499.

[85] C. L. Zitnick and S. B. Kang, “Stereo for image-based rendering using image over-segmentation,” *Int. J. Comput. Vis.*, vol. 75, no. 1, pp. 49–65, 2007.

[86] M. Bleyer and M. Gelautz, “Graph-based surface reconstruction from stereo pairs using image segmentation,” in *Proceedings of the SPIE Electronic Imaging Conference*, San Jose, CA, USA, 2005, pp. 288–299.

[87] C. Zhang, Z. Li, Y. Cheng, R. Cai, H. Chao, and Y. Rui, “Meshstereo: A global stereo model with mesh alignment regularization for view interpolation,” in *Proceedings of the IEEE International Conference on Computer Vision*, Tampa, FL, USA, 2015, pp. 2057–2065.

[88] H. Hirschm, “Stereo vision in structured environments by consistent semi-global matching,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York, NY, USA, 2006, pp. 2386–2393.

[89] A. Spyropoulos and P. Mordohai, “Ensemble Classifier for Combining Stereo Matching Algorithms,” in *Proceedings of the International Conference on 3D Vision*, Washington, DC, USA, 2015, pp. 73–81.

[90] Z. Chen, X. Sun, L. Wang, Y. Yu, and C. Huang, “A deep visual correspondence embedding model for stereo matching costs,” in *Proceedings of the IEEE International Conference on Computer Vision*, Santiago, Chile, 2015, pp. 972–980.

[91] J. Žbontar and Y. Le Cun, "Computing the stereo matching cost with a convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, 2015, pp. 1592–1599.

[92] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the International Joint Conference on Artificial Intelligence*, Vancouver, BC, Canada, 1981, pp. 674–679.

[93] Y. S. Heo, K. M. Lee, and S. U. Lee, "Simultaneous color consistency and depth map estimation for radiometrically varying stereo images," in *Proceedings of the International Conference on Computer Vision*, Kyoto, Japan, 2009, pp. 1771–1778.

[94] M. Michael, J. Salmen, J. Stallkamp, and M. Schlipsing, "Real-time stereo Vision : optimizing semi-global matching," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, Gold Coast City, Australia, 2013, pp. 1197–1202.

[95] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Madison, WI, USA, 2003, pp. 195–202.

[96] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *Proceedings of the German Conference on Pattern Recognition*, Münster, Germany, 2014, pp. 31–42.

[97] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Rob. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[98] B. Tippetts, D. J. Lee, K. Lillywhite, and J. Archibald, “Review of stereo vision algorithms and their suitability for resource-limited systems,” *J. Real-Time Image Process.*, vol. 11, no. 1, pp. 5–25, 2016.

[99] T. Xu, P. Cockshott, and S. Oehler, “Acceleration of stereo-matching on multi-core CPU and GPU,” in *Proceedings of the IEEE Intl. Conf. on High Performance Computing and Communications, IEEE 6th Intl. Symp. on Cyberspace Safety and Security, IEEE 11th Intl. Conf. on Embedded Software and Syst.*, Paris, France, 2014, pp. 108–115.

[100] S. Mattoccia, “Stereo vision algorithms for FPGAs,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Portland, OR, USA, 2013, pp. 636–641.

[101] S. Gehrig, R. Stalder, and N. Schneider, “A flexible high-resolution real-time low-power stereo vision engine,” in *Proceedings of the International Conference on Computer Vision Systems*, Copenhagen, Denmark, 2015, pp. 69–79.

[102] G.-S. Hong, W. Hoe, and B.-G. Kim, “Performance analysis of matching cost for stereo matching with CUDA,” in *Proceedings of the International Conference on Computer Vision Systems*, Berlin, Heidelberg,Germany, 2015, pp. 69–79.

[103] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, 2000.

[104] M. Bajura and U. Neumann, “Dynamic registration correction in video-based augmented reality systems,” *IEEE Comput. Graph. Appl.*, vol. 15, no. 5, pp. 52–60, 1995.

[105] A. L. Janin, D. W. Mizell, and T. P. Caudell, “Calibration of head-mounted displays for augmented reality applications,” in *Proceedings of the IEEE Virtual Reality Annual International Symposium*, Seattle, WA, USA, 1993, pp. 246–255.

[106] R. Radke, *Computer Vision for Visual Effects*. New York, NY, USA: Cambridge University Press, 2012.

[107] G. L. Mariottini, E. Alunno, and D. Prattichizzo, “The epipolar geometry toolbox (EGT) for MATLAB,” Rep., University of Siena, 2004.

[108] Elinux.org, “Rpi Camera Module,” 2016. [Online]. Available: http://elinux.org/Rpi_Camera_Module#Technical_Parameters_.28v.1_board.29. [Accessed: 02-Nov-2016].

[109] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Proceedings of the Alvey vision conference*, Mancheste, UK, 1988, p. 50.

[110] V. Andrea and B. Fulkerson, “VLFeat: An open and portable library of computer vision algorithms,” in *Proceedings of the 18th ACM international conference*, Firenze, Italy, 2010, pp. 1469–1472.

[111] H. Hirschmüller, M. Buder, and I. Ernst, “Memory efficient semi-global matching,” *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. 3, no. 9, pp. 371–376, 2012.

[112] H. Hirschmuller and D. Scharstein, “Evaluation of cost functions for stereo matching,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, USA, 2007, pp. 1–8.

[113] C. Banz, P. Pirsch, and H. Blume, “Evaluation of penalty functions for semi-global matching cost aggregation,” *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. 39, p. B3, 2012.

[114] H. Hu, Y. Rzhanov, P. J. Hatcher, and R. D. Bergeron, “Binary adaptive semi-global matching based on image edges,” in *Proceedings of the 7th International Conference on Digital Image Processing*, Los Angeles,CA, USA, 2015, p. 96311D–96311D.