Missouri S&T Library and Learning Resources

Scholars' Mine

Masters Theses

Student Theses and Dissertations

Summer 2017

Depth determination method for a trailer-truck test-bed

Aditya Thakur

Follow this and additional works at: https://scholarsmine.mst.edu/masters_theses

 Part of the Electrical and Computer Engineering Commons

Department:

Recommended Citation

Thakur, Aditya, "Depth determination method for a trailer-truck test-bed" (2017). *Masters Theses*. 7681.
https://scholarsmine.mst.edu/masters_theses/7681

This thesis is brought to you by Scholars' Mine, a service of the Missouri S&T Library and Learning Resources. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

DEPTH DETERMINATION METHOD FOR A TRAILER-TRUCK TEST-BED

by

ADITYA THAKUR

A THESIS

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

in

ELECTRICAL ENGINEERING

2017

Approved by

Dr. Levent Acar
Dr. Jagannathan Sarangapani
Dr. Randy Moss

Copyright 2017

ADITYA THAKUR

All Rights Reserved

ABSTRACT

A camera is considered as a good sensing device to obtain visual information from its surrounding environment. In a three dimensional space, a camera has a two dimensional plane. Projection of an object on this plane creates a two dimensional projection and loses information of the third dimension. A single projection is not enough to retrieve the lost third dimensional information about the object. Thus it makes it difficult to use a single camera as a sensing instrument. In this project, we have developed a method that determines the three dimensional information using a single camera. The method utilizes an assumption of the camera being in motion, allowing it to take projections on unaligned camera planes at different positions. A mathematical comparison of these projections gives us a deterministic value of depth of the surroundings from these camera planes.

ACKNOWLEDGMENTS

I would like to thank Dr. Levent Acar for being my adviser and guiding me in completing the current thesis. I also want to thank Dr. Jagannathan Sarangapani and Dr. Randy H. Moss for serving in the committee. I would like to thank my family and friends for their continued support and encouragement that pushed me towards the completion of this project. This research was supported by the Department of Electrical and Computer Engineering of the Missouri University of Science and Technology.

TABLE OF CONTENTS

Page

APPENDICES

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

A control scheme can be represented as the combination of four basic blocks that are Perception, Motion Control, Cognition Path Planing and Localization. A generalized model of this system as represented by the Book Autonomous Mobile Robots by Roland Siegwart Siegwart et al. (2011) in Figure 1.1. Perception and Localization are the blocks that we are focusing on in this research topic. A general approach for the perception is by the use of multiple sensors, such as ultrasonic range finder, LIDAR, etc. These sensors are easier to use but have problems of having limited field of view. These sensors are also susceptible to noise. A camera is a better alternative for perception. The data we retrieve from the camera is closer to human vision. But as in humans, we require two eyes to get better perception, a camera based system needs to have more than one camera. This makes the system costly. To avoid this drawback, a single camera approach for estimating depth of the environment is developed in this project. The information obtained through this method is then further used for localization and Map building.

Numerous work has been done to develop methods to compute the third dimension information using cameras. Adelson, et al. in Adelson and Wang (1992) have developed a method to obtain the depth by the use of a plenoptic camera. Another method in Hermans et al. (2009) uses a sliding projection to estimate the depth. Although this method gives good results, it is computationally expensive. There have been a few hybrid methods such as Matusik et al. (2000); Slabaugh et al. (2002). These methods make a rough estimation of 3D geometry and mix it with the traditional image rendering algorithms to obtain accurate results. Saxena, et al. in Saxena et al. (2005) use a probabilistic model on monocular images. In Esteban and Schmitt (2004), Esteban uses silhouette and stereo fusion for 3D object modeling. A few other methods use two aligned images to estimate the depth information using disparity maps as in Shadbolt (2003); Jain et al. (1995). One of these

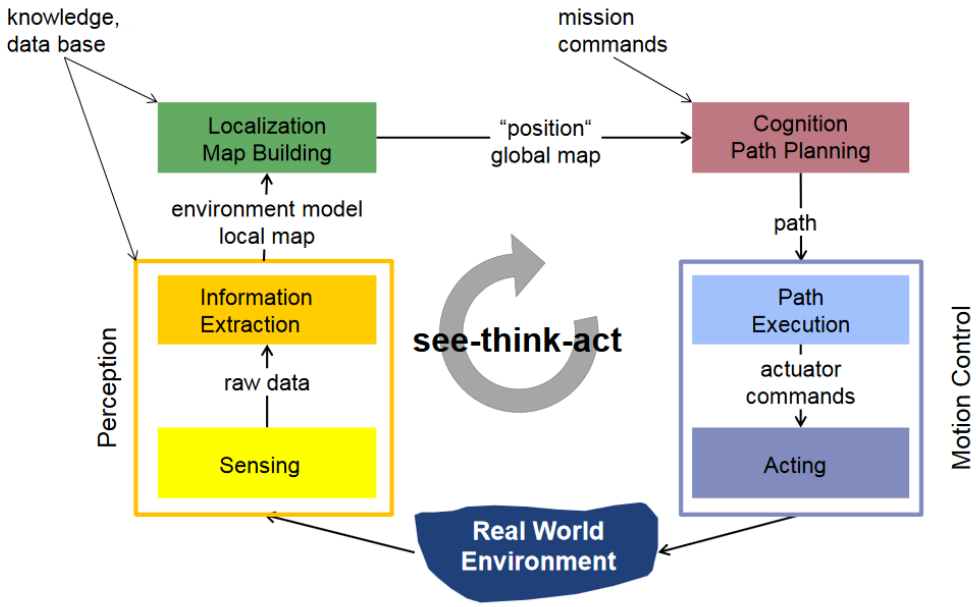

Figure 1.1. Reference Control Scheme for Mobile Robot Systems

methods by Jain, et al. in Jain et al. (1995), uses a human eye-like detection technique on two parallel images estimates the depth of an object. This method, being good for depth estimation, has a few limitations such as having more than one aligned camera planes. This method may or may not require more than one camera but because of that it can also be computationally expensive for the cases where the motion is not so linear.

Nowadays, there exist many system which move in a three dimensional space without adrones Apvrille et al. (2014). These systems still do not use the camera as a sensor of perceiving depth because of the limitations of currently existing methods. To avoid this problem, a new approach is developed by the use of single camera that moves through a three dimensional space. The limitations of requiring an aligned camera plane does not exist for this method.

The hardware setup we have used here is a miniaturized truck model that was designed by Pravin Dhake in Dhake (2007) and Robert Woodley in Woodley and Acar (1999). The test-bed they designed was not compatible with PCIe modules and thus, it was up-

graded to accommodate the PCIe modules. Using these modules, a stacked test-bed was designed. All the peripherals including motor driver, camera and sensors are connected to this test-bed. Using the information gathered from these peripherals, extra information about its surroundings is calculated. This information is then stored for further processing of complex calculation of object retrieval and map building.

This thesis is divided into two sections. In first part we will see how the data acquisition is done on this test-bed and in the later section we will see how the surroundings are mapped using the acquired data.

2. HARDWARE

Earlier, the hardware was designed around the PCI-104 bus. The modules from that test-bed were compatible with only PC/104 bus and the newer modules were compatible with the PCIexpress bus. This inconsistency was causing a conflict in communication and required extra hardware for interfacing. To avoid this, the modules were replaced and upgraded by adding Data Acquisition and Motor Driver Modules that supported the PCIe bus. The hardware list for the test bed is given below in Table 2.1,

2.1. CONTROLLER MODULE PFM-945C

The hardware was designed around the PFM-945C Motherboard Aaeon's, which is a Processor module designed for Intel Atom N270 Processor. The module has about 1GB of Memory and an expansion bus to support both PCI-104 and PCI-104-E bus. The older hardware was designed for PC/104 form factor based modules and it had legacy hardware issues. AAEON PFM 945C has clock speed of 1.6 Gigahertz. This board requires a power supply of 12V and 1.5 Amperes. The Mother board can be seen in a Figure 2.1 below,

Table 2.1. Hardware in Computer System of Test-bed

Type of Module	Name of Module
Controller Board	AAEON PFM-945C
Data Acquisition	RTD DM35425HR
Motor Controller	RTD DM35820HR
Frame Grabber	Sensoray 911
Storage Board	SSD-104 SATA

Figure 2.1. Controller Module PFM-945C

2.2. DATA ACQUISITION BOARD - RTD DM35425HR

The DM35425 is a software configurable high-speed, 12-bit data acquisition module in the PCIe/104 form factor. This module provides 16 differential or 32 single-ended analog input channels, with programmable gain and variable input range. The DM35425 also features four 12-bit high-speed analog outputs with programmable output range, and a 32-bit port of digital I/O. This module can acquire the data from all the analog sensors including the potentiometers and the IMU unit. The module can be seen in Figure 2.2.

2.3. DIGITAL I/O BOARD - DM35820

The DM35820HR is designed to provide high speed digital I/O for PCI/104-Express Systems. It interfaces with the PCI or PCI Express bus and uses large FIFOs and DMA transfers to allow for efficient data management. Several peripherals, including Pulse Width Modulators, Incremental Encoders, and Programmable Clocks are also provided with this

Figure 2.2. Data Acquisition Board DM35425

module. The Encoder is directly connected to this board which measures the distance covered by the system. The Digital I/O's helps us to provide a control signal to the Motor Driver Circuitry. The module can be seen in Figure 2.3.

2.4. STORAGE BOARD - SSD 104 SATA

SSD-104 SATA is the dual memory storage device supported by PCI-104 and PCIe-104 bus. It allows 2 mSATA SSD cards. The SSD cards improves the performance of the system. In this storage, we have installed the Debian OS which is patched with RTAI as in Raghavan (2014), to act like a real time system. The image grabber also captures and stores frames from the camera plane to the same storage. The SSD being faster storage system keeps the system performance from failing down. The Figure 2.4 shows the module SSD-104.

Figure 2.3. Digital I/O Board DM35820

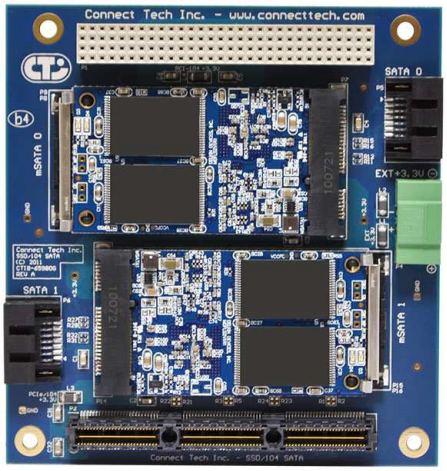

Figure 2.4. SSD Storage Module SSD-104 SATA

2.5. FRAME GRABBER - SENSORAY 911

Sensoray 911 Sensoray is four channel frame grabber that can simultaneously capture data from four image and audio channels. Here we are using just a single camera and thus the other channels are not used. It can capture 120 FPS with NTSC format or 100

FPS with PAL format. There is also a provision from I/O signals which can be used as a regular Digital Input/ Outputs. The frame grabber stores the data constantly on the storage module and it can be processed and stored in any format. This module can be seen in the Figure 2.5.

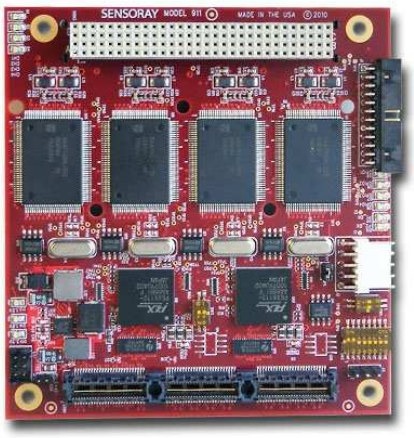

Figure 2.5. Frame Grabber - Sensoray 911

2.6. SENSORS

In this section, the sensors that are connected to the embedded computer are described.

2.6.1. Camera - Sony EVI. Sony EVI-D30 is an NTSC color camera as seen in Figure 2.6. We will be using this camera to capture frames while our system is on the move. The camera through an RCA port is connected to the frame grabber module mentioned

before. Camera has a pan and tilt setting using which it can be moved to take images from multiple angles. It is a high speed camera with auto focus and a 12X zoom. This helps us to achieve a faster performance for our overall system.

Figure 2.6. Camera - Sony-EVI-D30

2.6.2. Analog Accelerometer - DE ACCM3D. The analog accelerometer which works on a 3.3- volt power source provides three analog channels which provide pitch, roll and yaw motion measurement. This accelerometer is attached horizontally to the truck test-bed and it is used to find the turning of the truck along its axis. According to the angle, it generates voltage in the range or 0 to 3.3-volts and this voltage is given to the Data Acquisition board. The board then provides the digital output between the range of 0-4096. These values are then converted into the angle by which the truck has rotated. The accelerometer can be seen in Figure 2.7. The accelerometer gives 3.3 volts as the max voltage and the 1.66 point is considered a 0g point. The sensitivity of the given Accelerometer is 0.333v/g. Using this information, the motion can be calculated by using Equation 2.1

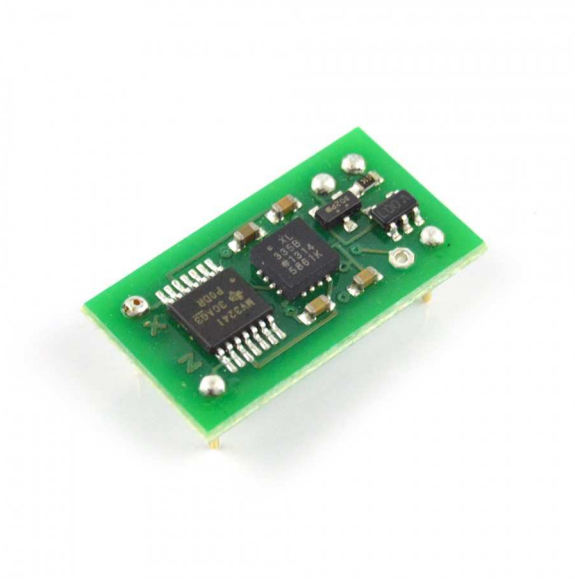

Figure 2.7. Accelerometer

Figure 2.8. Potentiometer

$$\delta = (\frac{3.3 * ADC_{val}}{4095} - 1.66)\frac{1}{0.333} \tag{2.1}$$

Figure 2.9. Servo Motor

2.6.3. Potentiometer. A potentiometer is attached to the joint where the Trailer is connected to the head of the truck. The potentiometer has a rotary play and it is used to measure the angle of rotation of the trailer with respect to the truck. The potentiometer provides an analog voltage which converts it to an angle after it provides it to the Data Acquisition board. The Potentiometer is showed in Figure 2.8. The possible angle is from -170 to 170 degrees and thus the ADC data is converted using Equation 2.2.

$$\delta_1 = (\frac{340 * ADC_{val}}{4095} - 170)\frac{\pi}{180}\text{radians} \tag{2.2}$$

2.6.4. Servo Motor. Servo motor is connected to the steering wheel and even though it is an actuator, it is also a sensor. Airtronics Servo Motor was used to implement the steering mechanism. The servo motor can be seen in Figure 2.9. The motor also has a potentiometer inside it. The analog signal from this potentiometer is used to measure the angle of rotation of the steering wheel. This data is given to the ADC of Data Acquisition Board. It converts the data into a Digital 12-bit value. This 12-bit value is then further substitutated in Equation 2.3 to obtain the actual angle. This will give us the angle in range

Table 2.2. Operating Range of All Hardware Components

Component	Data Rate	Operating Range	Update Frequency
Encoder	200MHz	0 – 5V	12.5MHz
Accelerometer	12-bit Digital from ADC	0 – 5V	1.25MHz
Trailer Potentiometer	12-bit Digital from ADC	0 – 5V	1.25MHz
Servo Potentiometer	12-bit Digital from ADC	0 – 5V	1.25MHz
Servo Motor		0 – 6V	25MHz
Camera	130fps	6V	2000Hz
Driving Motor		0 – 12V	25MHz

of -15 to 15 degrees

$$\delta_2 = (\frac{15 * ADC_{val}}{4095} - 30)\frac{\pi}{180}\text{radians} \tag{2.3}$$

2.6.5. Operating Range of Hardware Components. The maximum operating range of all connected hardware components is in Table 2.2.

3. SETUP OF TRUCK

The hardware in Section 2 was compiled together to construct a $1 : 16^{th}$ scale model of a truck and its trailer. The constructed model can be seen in Figure 3.1. The sensors except camera, allow the truck to measure own position from the starting point, and the trajectory it is going to follow.

Figure 3.1. Trailer-Truck Test-bed

Figure 3.2 shows the stackable embedded computer which is connected to the sensors. The representative block diagram of this system can be seen in Figure 3.3. The Data Acquisition board is connected to all the sensors including the accelerometer and the Potentiometer. The Digital I/O board provides PWM signals to the Motor Driver and the Servo Motor which is used to steer the truck. The Digital I/O board also has 16-bit Timer Counters which are used to measure the pulses generated by the Rotary Encoder.

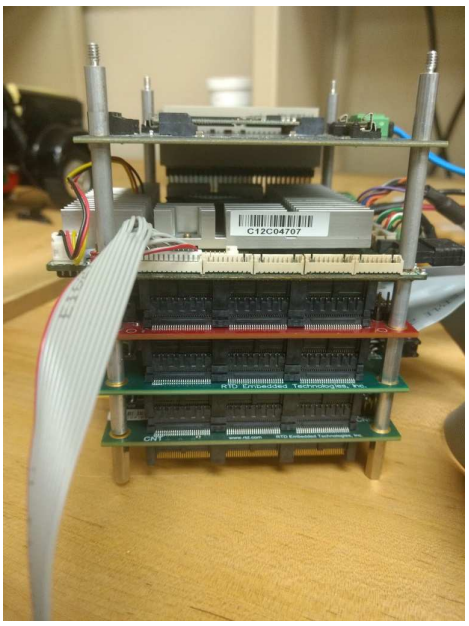

Figure 3.2. Stacked Embedded Computer System

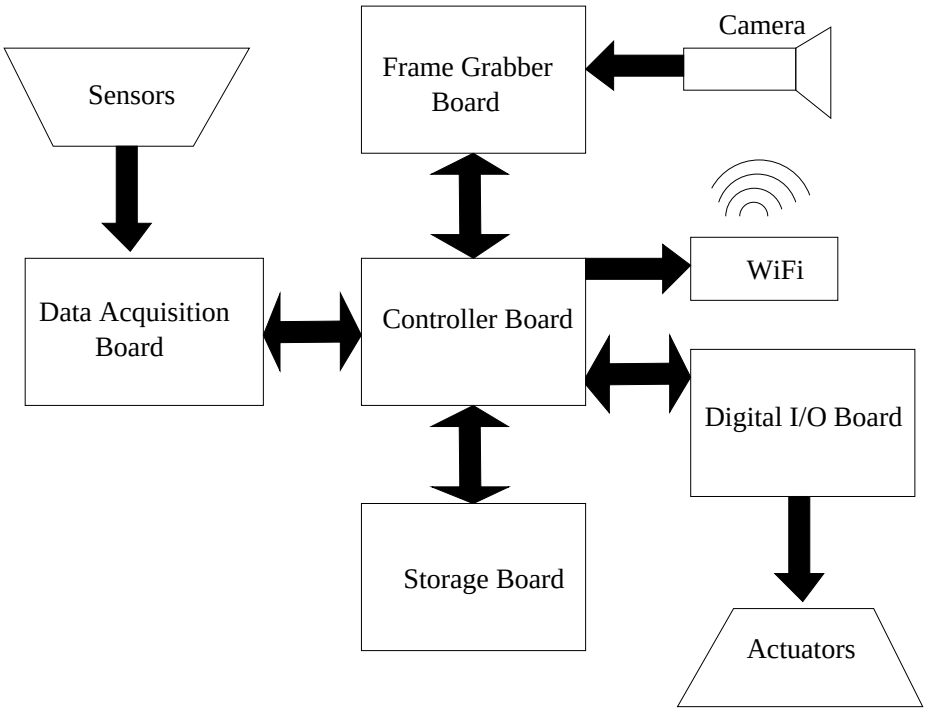

Figure 3.3. Block Diagram of Test-bed

The trailer truck system has USB ports and Ethernet ports too. Using which, the truck can be connected to the internet through a secure gateway. The truck has an on-board LAMP Server as in Timberlake (2010), which has Linux as operating system, Apache as

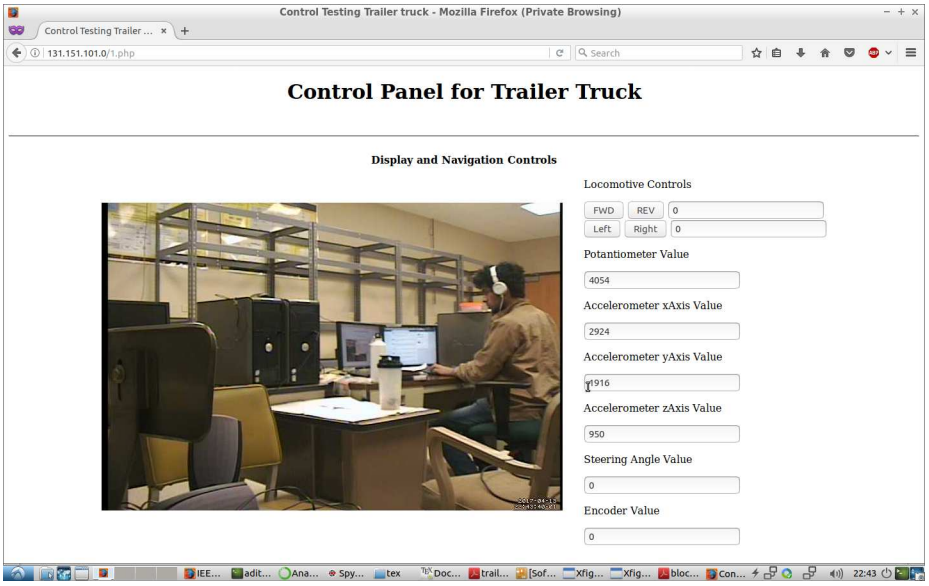

Figure 3.4. Control Panel of Truck

the web server, MySQL as the relational database and PHP as the object oriented scripting language. The video from the camera is shared with this server so the controller can have a constant video feed coming from the truck.

The system runs on a lightweight variant of a Debian Operating system called Xubuntu. The OS was chosen for its minimalism and thus it is not heavy on the resources. The kernel that comes with the Debian is not suitable for the real-time applications as it lacks the required scheduling capabilities of a real-time scheduler. To fix that, the kernel was patched with the Real-time Application Interface as in (RTAI) Mantegazza et al. (2000), which enables the operating system to behave like a real time system.

The control panel allows the user to drive the truck. The control panel has buttons to adjust its speed and also steer the truck. The values from the sensors are also collected and showed on the same panel for making it available as an API for designing a remote application as a future work.

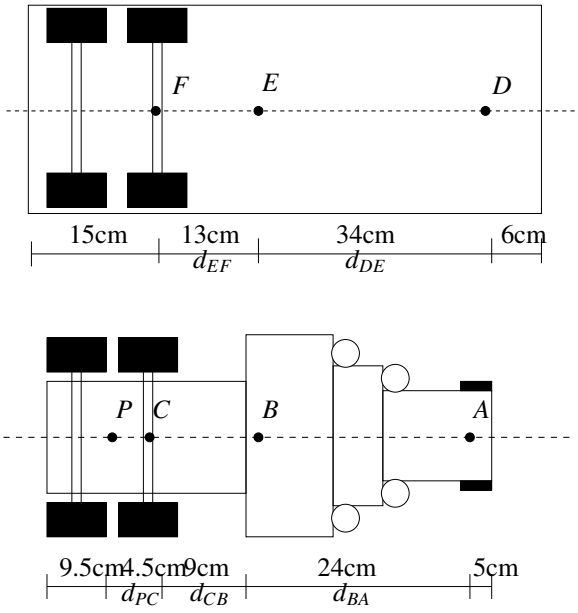

Figure 3.5. Dimensions of the Trailer Truck

Figure 3.5 shows us the points of interest and the dimensions of the truck and the trailer connected to it. Variable d is used to denote the dimensions in centimeters and the subscript denotes the difference between two points. For example, d_{CB} will show the distance between point B and C. The points P and D are the pivot points where the trailer is connected to the truck. A is the center for the steering tires and C is the center for the driving tires. Points B and E denote the centers of gravity for truck and the trailer, respectively.

The truck and the trailer can be reduced into a one dimensional model where the modules are reduced into a single line through their center. The reduced model and the related variables can be seen in Figure 3.6. Velocity and acceleration components are shown in two directions. One is along the direction of the truck and second component is in the perpendicular direction to the truck. The i^{th} component points towards the direction of Truck and j^{th} component points towards the perpendicular. For example, v_{D_j} is the component of velocity which is perpendicular from point D.

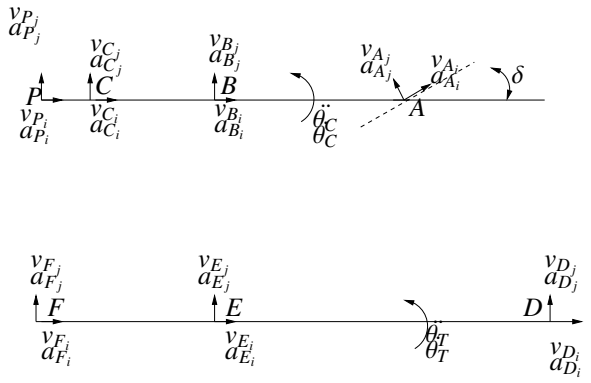

Figure 3.6. 1-dimensional Representation of the Truck

Angle δ is the angle of steering of the steering tires. In motion, the same angle is achieved by the truck while turning, with respect to the truck. The center of gravity of the truck is at point B and the camera is placed between points D and E. In next section, the points around the truck will be obtained in the three dimensional space. But the points will be denoted from the point of the camera and not from the trucks perspective. Thus it is required to convert these points into the trucks coordinate system, that is, the global coordinate system. The transformation of these points can be given by,

$$\boldsymbol{x_o} = [H_{s\to o}]\boldsymbol{x}_s \tag{3.1}$$

where H is the transformation matrix which converts the points from midpoint of D and E to the global coordinate system at point B.

$$\vec{H} = \begin{bmatrix} R & \vec{t} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\delta) & 0 & \sin(\delta) & t_1 \\ 0 & 1 & 0 & t_2 \\ -\sin(\delta) & 0 & \cos(\delta) & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

where, $\boldsymbol{x}_s$ is a three dimensional vector represented as a homogeneous vector with four components to accommodate the 4×4 Transformation matrix.

3.1. TRACKING THE TRUCK TRAJECTORY

Figure 3.7, shows the different points on the truck with respect to the global coordinate system. For the sake of simplicity, it is assumed that initially, the truck has its center of gravity at global origin and it parallel to the x-axis. Thus the pose angle is 0 initially. The coordinates X_1, Y_1, X_2, Y_2 and X_3, Y_3 can be represented by the following Equations. The angle $\theta_1(t)$ with respect to the global origin is addition of steering angle and the angle δ which is,

$$\theta_1(t) = \phi_1(t) + \int_0^t v_1(\sigma) \sin(\phi_1(\sigma)) d\sigma \tag{3.3}$$

The angle δ_1 is thus as mentioned in Equation 3.3, such that

$$\delta_1(t) = \frac{1}{l_1} \int_0^t v_1(\sigma) \sin(\phi_1(\sigma)) d\sigma \tag{3.4}$$

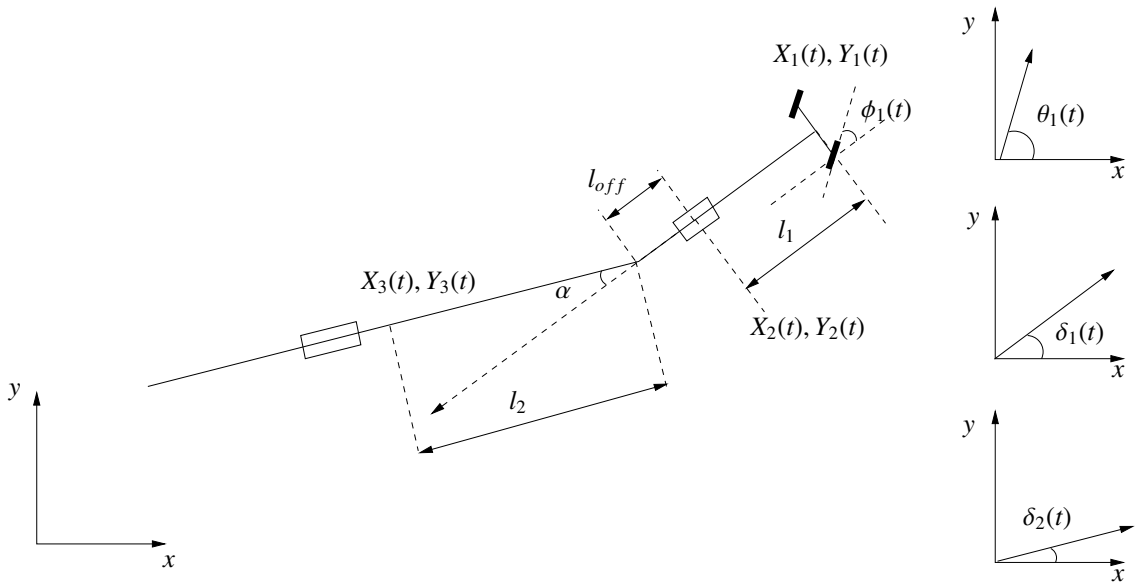

Figure 3.7. Analysis of Multiple Points on the Truck

Where, the velocity v_1 is the velocity from the driving wheels. the coordinates X_1, Y_1 can be obtained using angle θ_1, such that

$$\begin{aligned} X_1(t) &= \int_0^t v_1(\sigma)\cos(\theta_1(\sigma))d\sigma \\ Y_1(t) &= \int_0^t v_1(\sigma)\sin(\theta_1(\sigma))d\sigma \end{aligned} \tag{3.5}$$

The coordinates obtained are of the point where the center of gravity is located. We are interested in the point where the camera is located that is X_3, Y_3. To obtain that information, we first need the pivot point, that is X_2, Y_2,

$$\begin{aligned} X_2(t) &= X_1(t) - l_1 cos(\delta_1(t)) \\ Y_2(t) &= Y_1(t) - l_1 sin(\delta_1(t)) \end{aligned} \tag{3.6}$$

For obtaining the X_3, Y_3, we need the angle generated by the trialer with respect to the global coordinates.

$$\alpha = \delta_1(t) - \delta_2 \tag{3.7}$$

Where the angle α is the angle between trailer and truck. This angle is known from the feedback of potentiometer.

$$\begin{aligned} X_3(t) &= X_2(t) - l_2 cos(\delta_2(t)) \\ Y_3(t) &= Y_2(t) - l_2 sin(\delta_2(t)) \end{aligned} \tag{3.8}$$

Thus the final equation for the coordinates X_3, Y_3 are written, such that

$$\begin{aligned} X_3(t) &= \int_0^t v_1(\sigma) \cos(\theta_1(\sigma)) d\sigma - l_1 cos(\delta_1(t) - l_2 cos(\delta_2(t)) \\ Y_3(t) &= \int_0^t v_1(\sigma) \sin(\theta_1(\sigma)) d\sigma - l_1 sin(\delta_1(t)) - l_2 sin(\delta_2(t)) \end{aligned} \tag{3.9}$$

4. INTRODUCTION TO THE DEPTH DETERMINATION METHOD

With robotics and automation gaining prominence in the tech world, efficient and fast processing of visual data has become increasingly important. In the field of robot vision, various types of sensors, such as ultrasonic and LIDAR, are available to detect objects in their surroundings Besl (1988); Benet et al. (2002). Although these sensors are easy to use, their limitations such as limited sensing field and noise render them inconvenient for larger unknown environments.

A camera is considered a good sensing device to obtain visual information from its surrounding environment. In a three dimensional environment, most cameras provide only two dimensional projections at fixed locations. Thus additional methods are required to determine information about the third dimension.

In this section, we have developed a method that overcomes the limitations of requiring aligned camera planes. Our method determines the three dimensional information of an object from its projections on camera planes at two different locations. The camera takes a picture of an object at one location and then takes another picture of the same object from another location. Comparing these two projections, we present a method that mathematically provides a deterministic value for the three dimensional information of the object.

4.1. INFORMATION LOSS IN PROJECTIONS

To represent the object and the camera locations and incorporate possible rotation and translation movements, we denote a vector $\boldsymbol{x}$ in three dimensional space as $\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$. We assume that the global origin is at $\boldsymbol{o} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ and the local normal vector in the camera coordinate system is $\boldsymbol{n} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. When the camera takes a picture, the picture is a two dimensional projection of a three dimensional object on the camera plane.

In this process, we lose the information about the third dimension of the object. By taking the projection of the point $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$ onto the α camera plane we get

$$
\begin{aligned}
P_\alpha[\boldsymbol{x}] &= \left(I - \boldsymbol{n}_\alpha \boldsymbol{n}_\alpha{}^T\right)\boldsymbol{x} \\
&= \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x_{o_1} \\ x_{o_2} \\ x_{o_3} \end{bmatrix} \\
&= \begin{bmatrix} x_{\alpha_1} \\ x_{\alpha_2} \\ 0 \end{bmatrix}
\end{aligned}
\tag{4.1}
$$

As Equation (4.1) shows, when we take a projection of point $\boldsymbol{x}$ on a plane α with center at $\boldsymbol{c_\alpha}$ and normal vector $\boldsymbol{n_\alpha}$ perpendicular to the first two dimensions. A projection will lack the third dimension information. A point $\boldsymbol{x}$ can have multiple projections on multiple planes. For example, the projections of the point $\boldsymbol{x}$ on planes α and β are,

$$
P_\alpha\left[\ \boldsymbol{x}\ \right] = \begin{bmatrix} x_{\alpha_1} \\ x_{\alpha_2} \\ 0 \end{bmatrix}, \text{ and } P_\beta\left[\ \boldsymbol{x}\ \right] = \begin{bmatrix} x_{\beta_1} \\ x_{\beta_2} \\ 0 \end{bmatrix}, \text{respectively.} \tag{4.2}
$$

where $x_{\alpha_1}, x_{\alpha_2}$ and x_{β_1}, x_{β_2} are in local coordinates.

5. BINOCULAR DISPARITY IMAGE DEPTH ESTIMATION METHOD

Two planes α and β are represented by their centers $\boldsymbol{c_\alpha}$ and $\boldsymbol{c_\beta}$ and their normal vectors $\boldsymbol{n_\alpha}$ and $\boldsymbol{n_\beta}$, respectively. The center of the first plane is also assumed as the origin of the global coordinate system. In this method, we assume that the planes are perfectly aligned with each other, such that their normal vectors are translations in one dimension only. To obtain the third dimensional information using the Binocular Disparity Method, as described in Trucco and Verri (1998), we take the projection of a point on the image plane. A point $\boldsymbol{x_o}$ is described in a three dimensions with coordinates, such that $\boldsymbol{x_o} = \begin{bmatrix} x_{o_1} & x_{o_2} & x_{o_3} \end{bmatrix}^T$. The projection, with respect to the α plane, gives coordinates $P_\alpha \begin{bmatrix} \boldsymbol{x}_o \end{bmatrix} = \begin{bmatrix} x_{\alpha 1} & x_{\alpha 2} & 0 \end{bmatrix}^T$. Here the third dimension coordinate is 0 as described in Section 4.1. The $d_{1\to 2}$ is the relative distance between the centers of these two planes.

Figure 5.2 shows the camera configuration for the Stereo Disparity Method. f is the distance from the camera plane to each of the lenses, and is also the focal length of the cameras. Theorem 1 gives a relative approach to find the third dimension value of a point by finding a relationship between the actual point and the diminished image due to the lenses.

5.1. DEMONSTRATION OF THE BINOCULAR DISPARITY METHOD

The Binocular Disparity Method works with a single perfectly aligned camera sliding along on one of the dimensions of the camera plane or with a pair of perfectly aligned cameras that are placed along a line. In this method, we need to determine the focal length of the camera lenses in pixels. We obtain this information by calibrating the camera and by obtaining its intrinsic matrix as in Zhang (2000). Additionally, the camera measures

the first and second dimensions as the projection of the point on the camera planes α and β. Figure 5.1 shows the setup for calculations, where the distance between the cameras is $d_{1\to 2}$.

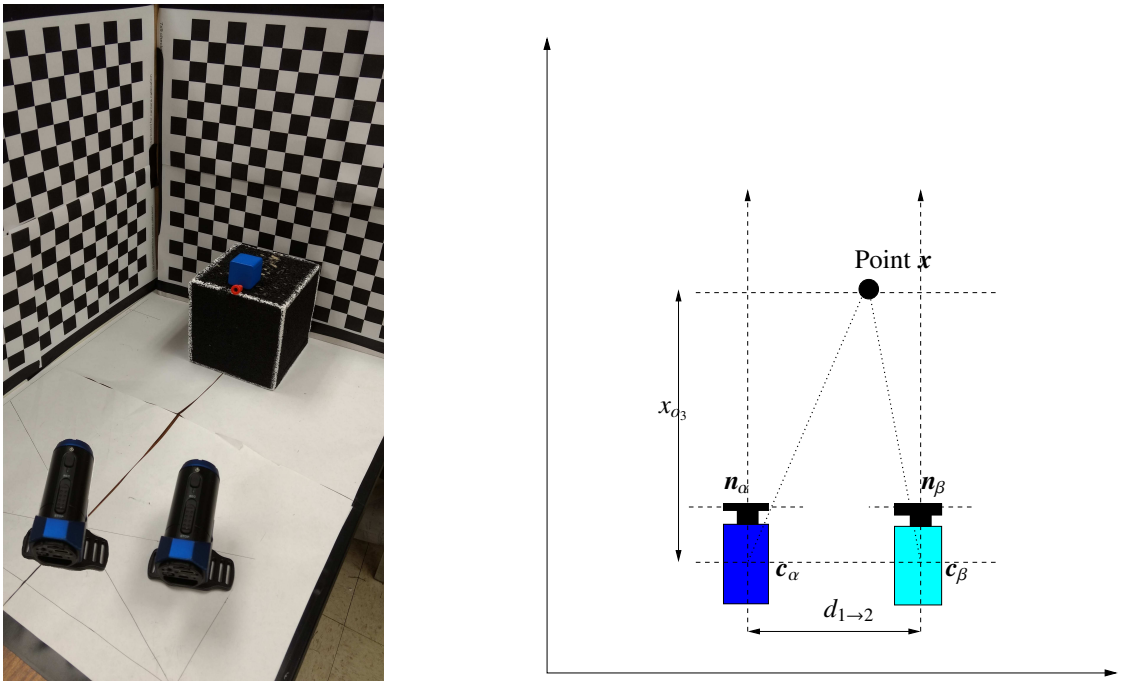

Figure 5.1. Experimental Setup for the Binocular Disparity Method

Theorem 1. *Given two perfectly aligned camera planes separated in one dimension*

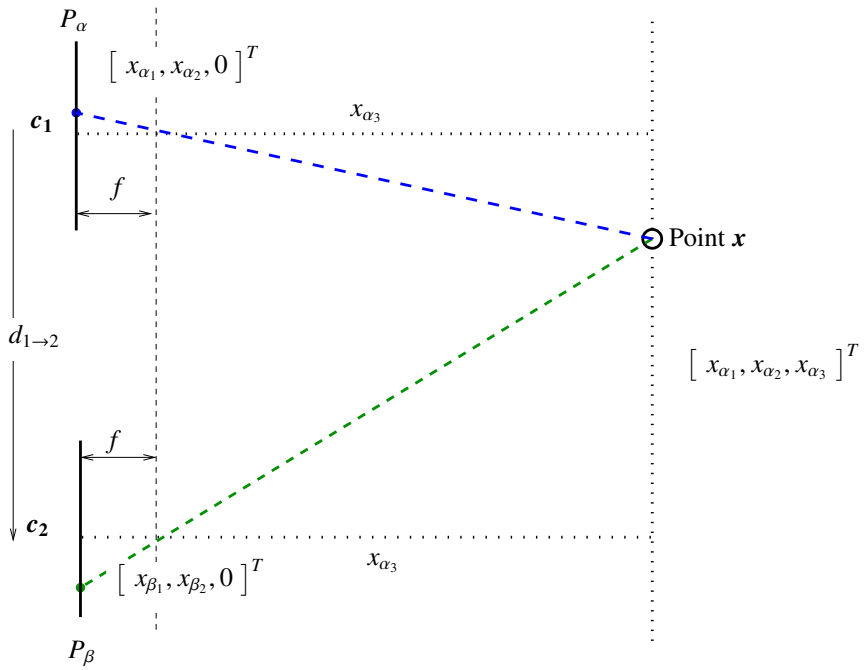

Figure 5.2. Projection of Point on Multiple Image Planes

the depth from plane of an object can be obtained by

$$\boldsymbol{x_\alpha} = \begin{bmatrix} \frac{x_{o_3}}{f} x_{\alpha_1} \\ \frac{x_{o_3}}{f} x_{\alpha_2} \\ \frac{d_{1\to 2}}{x_{\alpha_1} - x_{\beta_1}} f \end{bmatrix}. \tag{5.1}$$

Proof From Figure 5.2, we can observe that

$$\frac{x_{\alpha_1}}{f} = \frac{x_{o_1}}{x_{o_3}}, \quad \text{and} \quad \frac{x_{\beta_1}}{f} = \frac{d_{1\to 2} - x_{\emptyset_1}}{x_{o_3}} \tag{5.2}$$

Thus ,

$$x_{\alpha_1} = \frac{x_{o_1}}{x_{o_3}} f, \qquad x_{\beta_1} = \frac{x_{o_1} - d_{1\to 2}}{x_{o_3}} f \tag{5.3}$$

and

$$x_{\alpha_2} = \frac{x_{o_2}}{x_{o_3}} f, \qquad x_{\beta_2} = \frac{x_{o_2}}{x_{o_3}} f \tag{5.4}$$

Calculating disparity using the above equations, such that

$$x_{\alpha_1} - x_{\beta_1} = \frac{x_{o_1}}{x_{o_3}} f - \left(\frac{x_{o_1} - d_{1\to 2}}{x_{o_3}} \right) f = \frac{d_{1\to 2}}{x_{o_3}} f. \tag{5.5}$$

Thus, the depth or the 3rd dimensional information of given point is

$$x_{o_3} = \frac{d_{1\to 2}}{x_{\alpha_1} - x_{\beta_1}} f. \tag{5.6}$$

Using the depth information, all the actual coordinates of the point $\boldsymbol{x}$ can be obtained as follows

$$x_{o_1} = \frac{x_{o_3}}{f} x_{\alpha_1} \qquad \text{and} \qquad x_{o_2} = \frac{x_{o_3}}{f} x_{\alpha_2}. \tag{5.7}$$

So the actual point from the perspective of first plane can be represented as

$$\boldsymbol{x}_\alpha = \begin{bmatrix} x_{\alpha_1} \frac{x_{o_3}}{f} \\ x_{\alpha_2} \frac{x_{o_3}}{f} \\ \frac{d_{1\to 2}}{x_{\alpha_1} - x_{\beta_1}} f \end{bmatrix}. \tag{5.8}$$

□

Definition 1. *The method to calculate the coordinates of a point in the three dimensional space using Theorem* 1 *is called the 'Binocular Disparity Method'.*

For demonstration of this method, we used an Ion Air Pro camera. We utilized the same camera on the method described in Section 6. We placed a round red object as our point of interest. We calculated the first and second coordinate values in pixels by observing the point in multiple images. Figure 5.3 shows three images of the object taken from multiple positions.

Figure 5.3. Images for the Binocular Disparity Method

From the images in Figure 5.3, we get the pixel data in three camera planes as,

$$\boldsymbol{x}_\alpha{}^* = \begin{bmatrix} 1547 \\ y_\alpha{}^* \\ 0 \end{bmatrix}, \text{ and } \boldsymbol{x}_\beta{}^* = \begin{bmatrix} 1145 \\ y_\beta{}^* \\ 0 \end{bmatrix}, \text{ and } \boldsymbol{x}_\gamma{}^* = \begin{bmatrix} 667 \\ y_\gamma{}^* \\ 0 \end{bmatrix}, \tag{5.9}$$

To adjust the pixel data to the center of image plane from the left top corner, we subtract half of the image from the raw data, such that

$$
\begin{aligned}
\boldsymbol{x}_\alpha &= \begin{bmatrix} 1547 - 1024 \\ x^*_{\alpha_2} - 768 \\ 0 \end{bmatrix} = \begin{bmatrix} 523 \\ x_{\alpha_2} \\ 0 \end{bmatrix} \\
\boldsymbol{x}_\beta &= \begin{bmatrix} 1145 - 1024 \\ x^*_{\beta_2} - 768 \\ 0 \end{bmatrix} = \begin{bmatrix} 121 \\ x_{\beta_2} \\ 0 \end{bmatrix} \\
\boldsymbol{x}_\gamma &= \begin{bmatrix} 667 - 1024 \\ x^*_{\gamma_2} - 768 \\ 0 \end{bmatrix} = \begin{bmatrix} -557 \\ x_{\gamma_2} \\ 0 \end{bmatrix}
\end{aligned} \tag{5.10}
$$

We calculated the focal length of camera in pixels from the calibration matrix which was 970 pixels. For Image A and B, the camera is translated by 19 cm and for Image A and C it is translated by 30 cm. We put this information into Equation(5.8),

$$
x_{\alpha_3} = \frac{0.3 \times 970}{523 + 557}\,\mathrm{m} = 0.2694\,\mathrm{m}, \tag{5.11}
$$

which is very close to the actual depth of about 28 cm. To validate the results, we rechecked the calculations with Image A and Image C, we get

$$
x_{\alpha_3} = \frac{0.19 \times 970}{121 + 557}\,\mathrm{m} = 0.2718\,\mathrm{m}. \tag{5.12}
$$

We used these equations with the data provided by Middleburry in Scharstein et al. (2014). Figure 5.4 shows set of these images. The features from α plane were matched with the features in β plane. We used the algorithm from the Binocular Disparity method on these features to obtain the depth of all the points. The points in the set of images can be seen represented in three dimensional space in Figure 5.6.

Figure 5.4. Camera plane α

Figure 5.5. Camera plane β

5.2. LIMITATIONS OF THE METHOD

For the Binocular Disparity method, the projections are to be taken at the equal set of intervals. The planes have to be perfectly aligned to each other for this method to work. This limits the use of this method as it starts to work only for small changes in position. Another limitation for this method is when the distance between the object and the cameras is less than 30 times of the distance between camera planes, this method fails, as in Kytö et al. (2011). In addition, if this method is used with a single camera, there is a need of rotating the camera planes mathematically for each iteration.

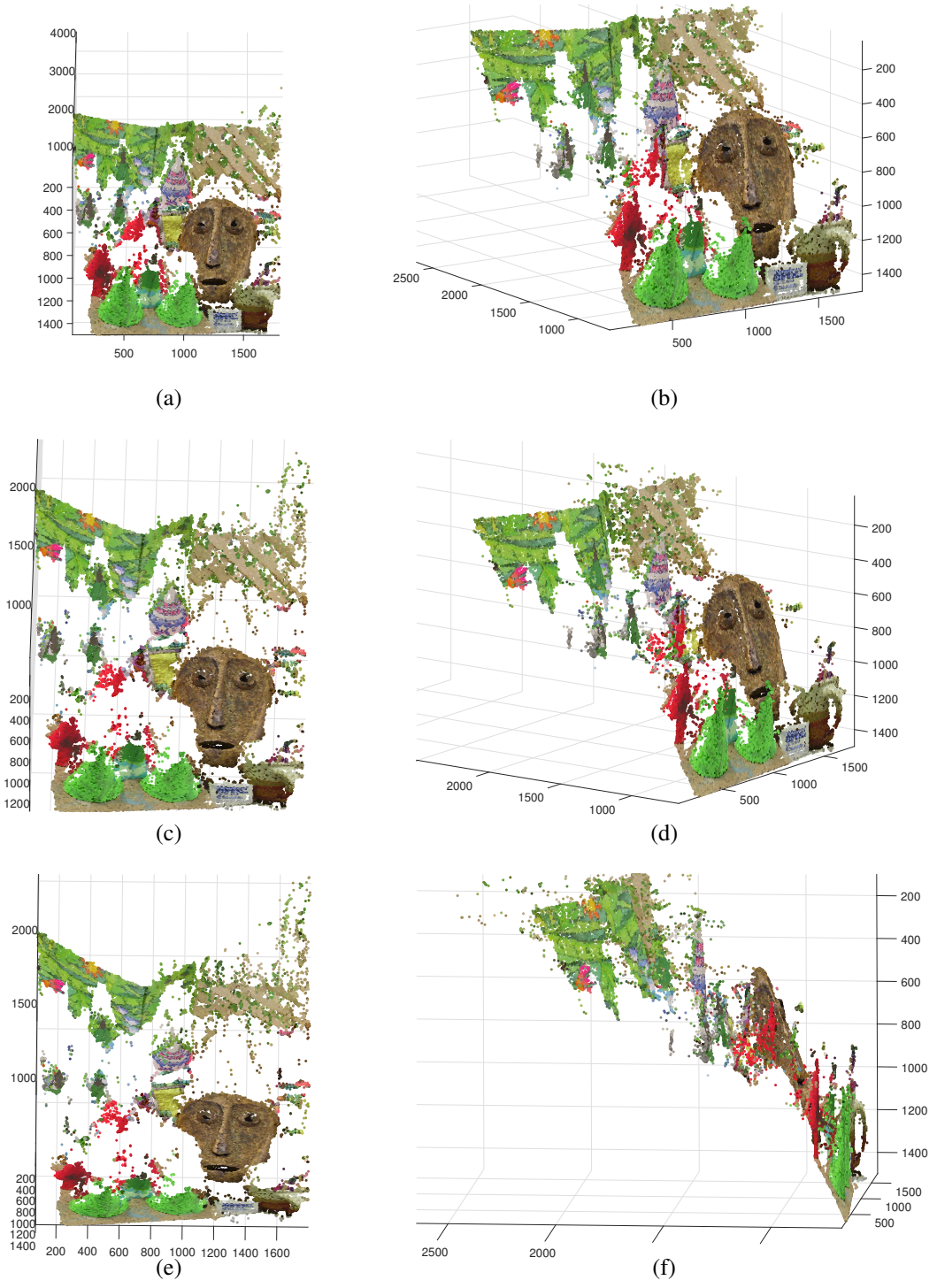

Figure 5.6. Display of Resulted Points from Binocular Disparity Method in three Dimensional Space (a),(c),(e) from front to top and (b),(d),(f) from front to side

6. OBTAINING DEPTH FROM UNALIGNED PLANES METHOD

In this method, we use properties of projection from different perspectives to obtain the third dimension information. The two camera locations can be practically anywhere that eliminates one of the limitations of the earlier method, the particular alignment of the two cameras. We assume that the projection of a point $\boldsymbol{x}$ is being projected on two unaligned planes α and β as in Figure 6.1. The planes are represented by centers $\boldsymbol{c_\alpha}$ and $\boldsymbol{c_\beta}$ with normal vectors $\boldsymbol{n_\alpha}$ and $\boldsymbol{n_\beta}$, respectively. There can be angular difference present between these planes in contrast to the earlier method. We represent the angular difference in terms of normal vectors, such that

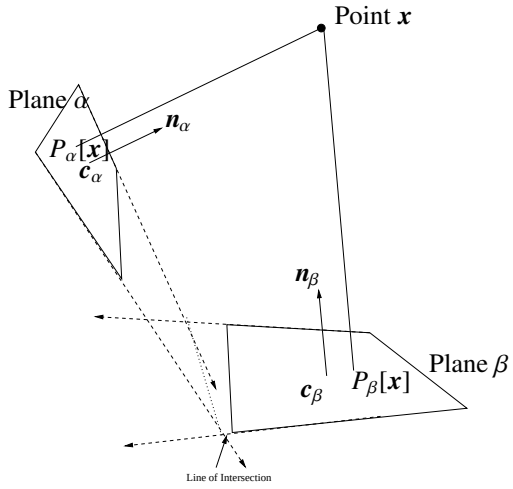

Figure 6.1. Projection of Point on Unaligned Image Planes

$$\phi = \cos^{-1}\left(\frac{\boldsymbol{i}_{1_\alpha}.\boldsymbol{i}_{1_\beta}}{||\boldsymbol{i}_{1_\alpha}|||\boldsymbol{i}_{1_\beta}||}\right), \quad \psi = \cos^{-1}\left(\frac{\boldsymbol{i}_{2_\alpha}.\boldsymbol{i}_{2_\beta}}{||\boldsymbol{i}_{2_\alpha}|||\boldsymbol{i}_{2_\beta}||}\right), \quad \theta = \cos^{-1}\left(\frac{\boldsymbol{i}_{3_\alpha}.\boldsymbol{i}_{3_\beta}}{||\boldsymbol{i}_{3_\alpha}|||\boldsymbol{i}_{3_\beta}||}\right) \tag{6.1}$$

where, the $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$ components are the projections of the normal vectors $\boldsymbol{n_\alpha}$ and $\boldsymbol{n_\beta}$ on the global planes x, y and z, respectively. We obtain these projection vectors from the usual basis vectors $\boldsymbol{e_1}$, $\boldsymbol{e_2}$ and $\boldsymbol{e_3}$,such that for second and third coordinate plane,

$$\boldsymbol{i}_{1_\alpha} = (I_{3\times3} - \boldsymbol{e_1}\boldsymbol{e_1}^T)\boldsymbol{n_\alpha}, \quad \boldsymbol{i}_{1_\beta} = (I_{3\times3} - \boldsymbol{e_1}\boldsymbol{e_1}^T)\boldsymbol{n_\beta} \tag{6.2}$$

for first and third coordinate plane,

$$\boldsymbol{i}_{2_\alpha} = (I_{3\times3} - \boldsymbol{e_2}\boldsymbol{e_2}^T)\boldsymbol{n_\alpha}, \quad \boldsymbol{i}_{2_\beta} = (I_{3\times3} - \boldsymbol{e_2}\boldsymbol{e_2}^T)\boldsymbol{n_\beta} \tag{6.3}$$

and for first and second coordinate plane,

$$\boldsymbol{i}_{3_\alpha} = (I_{3\times3} - \boldsymbol{e_3}\boldsymbol{e_3}^T)\boldsymbol{n_\alpha}, \quad \boldsymbol{i}_{3_\beta} = (I_{3\times3} - \boldsymbol{e_3}\boldsymbol{e_3}^T)\boldsymbol{n_\beta} \tag{6.4}$$

The perpendicular drawn from the projections along the plane towards the line of intersection and the projection vectors coming from the point $\boldsymbol{x}$ are co-planar from Theorem 2

Theorem 2. *The Projection vectors from a point $\boldsymbol{x}$ on planes α and β, and the perpendicular vectors along the plane from these projections drawn towards the line of projection are co-planar.*

Proof Two vectors always intersect in a line as long as they are not parallel. Let the planes be specified in Hessian normal form, then the line of intersection must be perpendicular to both $\boldsymbol{n}_\alpha$ and $\boldsymbol{n}_\beta$, which means it is parallel to

$$a = \boldsymbol{n}_\alpha \times \boldsymbol{n}_\beta \tag{6.5}$$

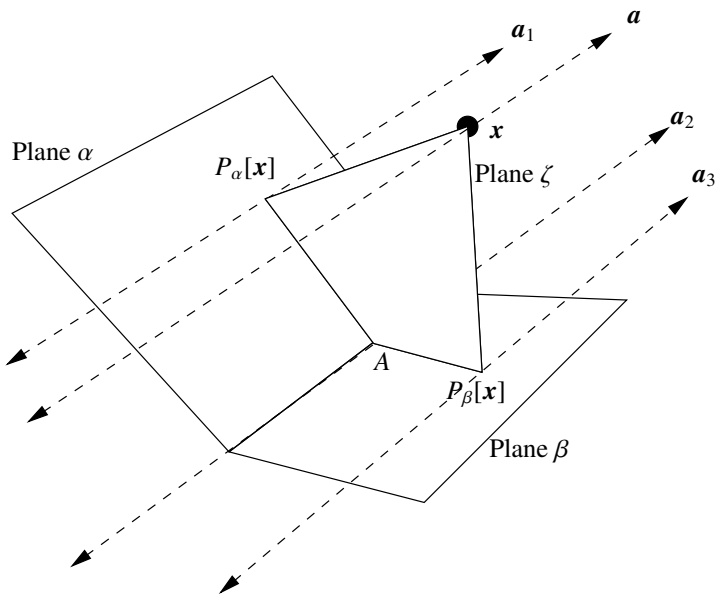

Figure 6.2. Coplanarity of Projection Vectors and its Components

The projection vectors points towards the same direction as the normal vectors and intersect at point $\boldsymbol{x}$. Thus the Equation (6.5) can be written as,

$$a = P_\alpha[\boldsymbol{x}] \times P_\beta[\boldsymbol{x}] \tag{6.6}$$

These three points $\boldsymbol{x}, P_\alpha[\boldsymbol{x}]$ and $P_\beta[\boldsymbol{x}]$ form a plane ζ. Line of intersection and the vector orthogonal to the plane ζ are thus parallel.

Let vector $\boldsymbol{a_2}$ be the line of intersection of planes and vectors $\boldsymbol{a_1}$ and $\boldsymbol{a_3}$ be the orthogonal vectors drawn at projections $P_\alpha[\boldsymbol{x}]$ and $P_\beta[\boldsymbol{x}]$, respectively as in Figure 6.2, such that

$$\boldsymbol{a} \quad \| \quad \boldsymbol{a}_1 \quad \| \quad \boldsymbol{a}_2 \quad \| \quad \boldsymbol{a}_3 \tag{6.7}$$

Thus any co-planar vector drawn from the projection meets the line of intersection $\boldsymbol{a_2}$ is a perpendicular to it. □

Corollary 1. *The perpendicular vectors drawn along the planes towards the line of intersection from the points of projection vectors, which are co-planar to the actual vectors of projection, intersect the line of intersection at the same point.*

Proof The vectors drawn perpendicular towards the line of intersection are co-planar to plane ζ. Let these points meet the line of intersection at points A_1 and A_2, such as

$$\overrightarrow{P_\alpha[\boldsymbol{x}]\, A_1} \perp \boldsymbol{a_2} \quad , \quad \overrightarrow{P_\alpha[\boldsymbol{x}]\, A_2} \perp \boldsymbol{a_2} \tag{6.8}$$

The line of intersection $\boldsymbol{a_2}$ is orthogonal to the plane ζ. Thus there can not exist two different points on $\boldsymbol{a_2}$ which can be co-planar to plane ζ. Thus the points A_1 and A_2 are the same points. □

The camera planes are unaligned. Thus the planes can be rotated by three possible angles of rotation. The three dimensional Rotation matrix R is represented by

$$R = R_x R_y R_z, \tag{6.9}$$

where R is the overall rotation matrix, and R_x, R_y, and R_z are the rotation matrices along each axis. The combined rotational matrix R is represented as

$$R = \begin{bmatrix} \cos(\theta)\cos(\psi) & -\sin(\theta) & \cos(\theta)\sin(\psi) \\ \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) & \cos(\phi)\cos(\theta) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) \\ \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \sin(\phi)\cos(\theta) & \sin(\phi)\sin(\theta)\sin(\psi) - \cos(\phi)\cos(\psi) \end{bmatrix} \tag{6.10}$$

The angles θ,ϕ and ψ can be obtained using Equation(6.1). The translation vector $\boldsymbol{t}$ can be represented as,

$$\boldsymbol{t} = \begin{bmatrix} c_{\alpha_1} - c_{\beta_1} \\ c_{\alpha_2} - c_{\beta_2} \\ c_{\alpha_3} - c_{\beta_3} \end{bmatrix}, \tag{6.11}$$

A combination of R and $\boldsymbol{t}$ forms a Transformation matrix H. A transformation matrix H can be represented in the form $H_{\alpha\to\beta}$, where transformation happens from coordinate system α to coordinate system β.

A real world example for visualizing this method we can consider. A robot equipped with a camera is traversing through its surroundings. It takes two projections of a point object from two different positions with unaligned camera planes. We know the centers $\boldsymbol{c_\alpha}$ and $\boldsymbol{c_\beta}$ of the camera planes with respect to the global co-ordinate system. We can calculate angular difference between the camera planes using Equation(6.1).

Theorem 3. *A point $\boldsymbol{x}$ has projections on unaligned camera planes α and β. The projections are $P_\alpha[\boldsymbol{x}] = \begin{bmatrix} x_{\alpha_1} & x_{\alpha_2} & 0 \end{bmatrix}^T$ and $P_\beta[\boldsymbol{x}] = \begin{bmatrix} x_{\beta_1} & x_{\beta_2} & 0 \end{bmatrix}^T$. The planes are separated by translation vector $\boldsymbol{t}$ and rotated by angles ϕ, ψ and θ, forming the Rotation Matrix R and a combined transformation matrix $H_{\alpha\to\beta}$. $H_{\alpha\to\beta}$ transforms the point vectors from α coordinate system to β coordinate system. The depth of point $\boldsymbol{x}$ from α plane can be estimated by the ratio of Euclidean distance between projection of $P_\alpha[\boldsymbol{x}]$ on β plane $P_\beta[H_{\alpha\to o}P_\alpha[\boldsymbol{x}]]$ and the projection of $\boldsymbol{x}$ on β plane $P_\beta[\boldsymbol{x}]$ with the sine of angle $\measuredangle\mu$, where $\measuredangle\nu$ is the angle between the planes at virtual intersection of camera planes.*

$$x_{\alpha_3} = \frac{||P_\beta[H_{\alpha\to\beta}P_\alpha[\boldsymbol{x}]] - P_\beta[\boldsymbol{x}||]}{\sin(\measuredangle\nu)} \tag{6.12}$$

Proof

As seen in Figure 6.3, the plane with center $\boldsymbol{c_\alpha}$ and normal vector $\boldsymbol{n_\alpha}$ is plane α, and plane with center $\boldsymbol{c_\beta}$ and normal vector $\boldsymbol{n_\beta}$ is plane β. The point $\boldsymbol{x}$ is projected on plane α and β at points $P_\alpha[\boldsymbol{x}]$ and $P_\beta[\boldsymbol{x}]$ respectively.

The point $P_\alpha[\boldsymbol{x}]$ is further projected on plane β at point $P_\beta[H_{\alpha\to\beta}P_\alpha[\boldsymbol{x}]]$.

The planes intersect at a line vector $\boldsymbol{a_2}$, as in Figure 6.2. There is a point E on the line of intersection. The vectors of projections are originated at point $\boldsymbol{x}$. This forms a co-planar quadrilateral with vectors of projection, joining the planes at 90 degrees each as in Theorem 2. This plane is called plane ζ.

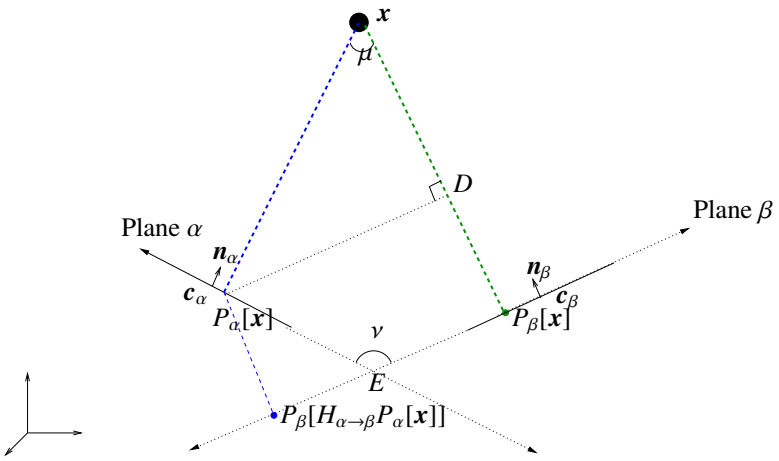

Figure 6.3. Projections of Points with respect to the World Coordinate System

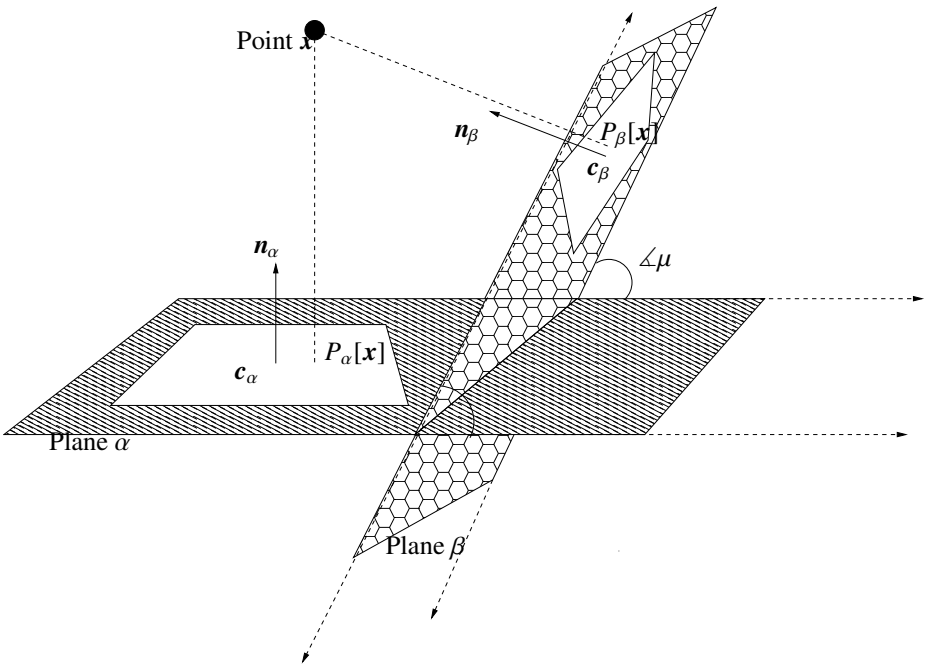

Figure 6.4. Representation of Projections with Respect to the Angle between Planes

The planes intersect each other in three dimensional space as in Figure 6.1, where Point $\boldsymbol{x}$, $P_\alpha[\boldsymbol{x}]$ and $P_\beta[\boldsymbol{x}]$ are co-planar. There also exists a point A on a line of intersection of planes α and β. This point co-planar to plane ζ as in Corollary 1, such that

$$\overrightarrow{(\boldsymbol{x}\ P_\alpha[\boldsymbol{x}])} \times \overrightarrow{(\boldsymbol{x}\ P_\beta[\boldsymbol{x}])}.\overrightarrow{\boldsymbol{x}A} = 0 \tag{6.13}$$

The angle ν is the angle between the points of projections $P_\alpha[\boldsymbol{x}]$, $P_\beta[\boldsymbol{x}]$. There is only one angle $\measuredangle\mu$ at the intersection of the planes, as in Figure 6.4. That is given by the normal vectors $\boldsymbol{n}_\alpha$ and $\boldsymbol{n}_\beta$ by the equation,

$$\measuredangle\nu = cos^{-1}\frac{\boldsymbol{n}_\alpha \boldsymbol{n}_\beta}{||\boldsymbol{n}_\alpha||||\boldsymbol{n}_\beta||} \tag{6.14}$$

Also, the plane ζ is a quadrilateral with 2 angles at 90 degrees. Thus, angle μ is obtained by,

$$\measuredangle\mu = 180^\circ - \measuredangle\nu \tag{6.15}$$

A perpendicular is drawn from point $P_\alpha[\boldsymbol{x}]$ on the line joining $\boldsymbol{x}$ and $P_\beta[\boldsymbol{x}]$. From Figure 6.3, we can see,

$$\overrightarrow{(P_\alpha[\boldsymbol{x}]\ P_\beta[H_{\alpha\to\beta}P_\alpha[\boldsymbol{x}]])} \quad || \quad \overrightarrow{(D\ P_\beta[\boldsymbol{x}])} \tag{6.16}$$

Thus the Euclidean distances between the projections is same as,

$$||P_\alpha[\boldsymbol{x}] - D|| = ||P_\beta[H_{\alpha\to\beta}P_\alpha[\boldsymbol{x}]] - P_\beta[\boldsymbol{x}]|| \tag{6.17}$$

and thus, additionally, trigonometric ratios around angle $\measuredangle\mu$ produces relationship as given in Equation (6.18)

$$||P_\alpha[\boldsymbol{x}] - \boldsymbol{x}|| = \frac{||P_\alpha[\boldsymbol{x}] - D||}{\sin(\measuredangle\nu)} \tag{6.18}$$

Substituting Equation (6.17) in Equation (6.18) we get,

$$\|P_\alpha[\boldsymbol{x}] - \boldsymbol{x}\| = \frac{\|P_\beta[H_{\alpha\to\beta}P_\alpha[\boldsymbol{x}]] - P_\beta[\boldsymbol{x}]\|}{\sin(\measuredangle\nu)} \tag{6.19}$$

Here $\|P_\alpha[\boldsymbol{x}] - \boldsymbol{x}\|$ is the depth of projection on plane α. From Equation (6.15), we get

$$sin(\measuredangle\mu) = sin(180^\circ - \measuredangle\nu) = sin(\measuredangle\nu) \tag{6.20}$$

Substituting these values in the above equation we get,

$$x_{\alpha_3} = \frac{\|P_\beta[H_{\alpha\to\beta}P_\alpha[\boldsymbol{x}]] - P_\beta[\boldsymbol{x}]\|}{\sin(\measuredangle\nu)} \tag{6.21}$$

□

Definition 2. *This method to calculate the three dimensional information of a point in the visible space of camera planes using Theorem* 3 *is named as Depth from Unaligned Planes Theorem.*

The three dimensional information of the point from plane β can be shown as,

$$\boldsymbol{x}_\alpha = \begin{bmatrix} x_{\alpha 1} \\ x_{\alpha_2} \\ \frac{\|P_\beta[H_{\alpha\to\beta}P_\alpha[\boldsymbol{x}]] - P_\beta[\boldsymbol{x}]\|}{\sin(\measuredangle\nu)} \end{bmatrix}. \tag{6.22}$$

We can further convert this point to the global coordinate system using the Rotational matrix and Translation vector $R_{\alpha\to w}$ and $\boldsymbol{t_{\alpha\to w}}$ respectively as follows:

$$\boldsymbol{x} = R_{\alpha\to w}\boldsymbol{x}_\alpha + \boldsymbol{t_{\alpha\to w}} \text{ or } \boldsymbol{x} = \mathrm{H}_{\alpha\to o}\boldsymbol{x}_\alpha \tag{6.23}$$

6.1. APPLICATION OF DEPTH FROM UNALIGNED PLANES METHOD

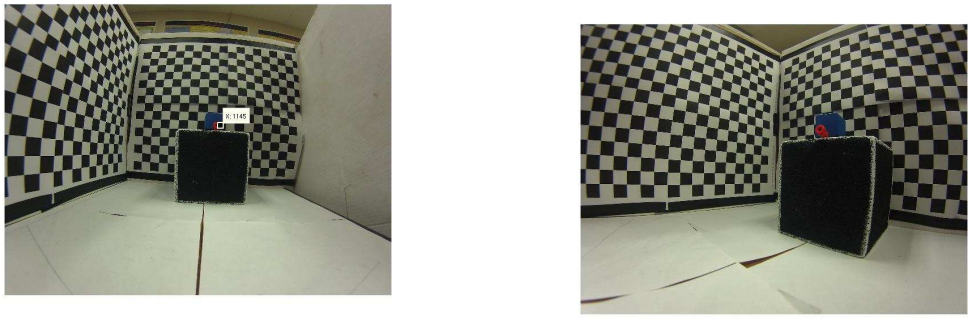

Figure 6.5. Image of the Object Taken from 1st and 2nd Camera Planes

First example, we choose the camera planes that are aligned at an angle of 45 degrees with each other, as in Figure 6.6. The representation of experimental setup is demonstrated in Figure 6.6. The center of plane α is at $\boldsymbol{c}_\alpha = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$, and the center of plane β is at $\boldsymbol{c}_\beta = \begin{bmatrix} 170 & 0 & 170 \end{bmatrix}^T$. The projections of point $\boldsymbol{x}$ on plane α is at $P_\alpha[\boldsymbol{x}] = \begin{bmatrix} 25 & 0 & 0 \end{bmatrix}^T$ and on plane β is at $P_\beta[\boldsymbol{x}] = \begin{bmatrix} -42 & 0 & 0 \end{bmatrix}^T$. The angle ν between the camera planes is 45 degrees. Thus putting it in the Equation(6.22), we get,

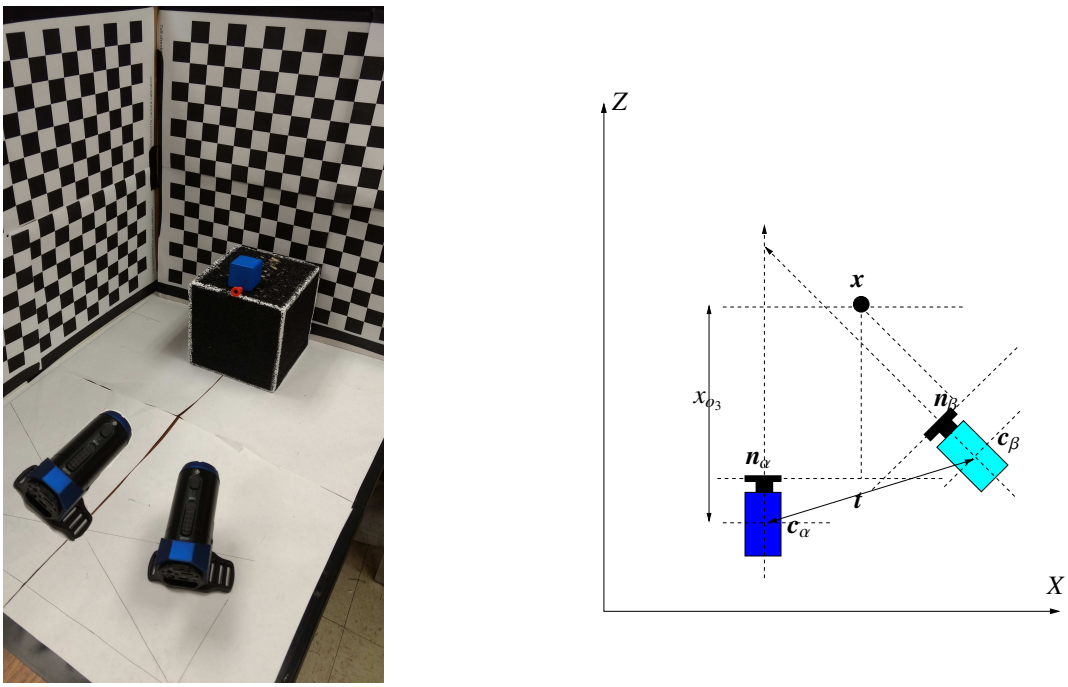

Figure 6.6. Representation of Experimental Setup for Example 1 of Unaligned Planes Method

$$\boldsymbol{x}_\alpha = \begin{bmatrix} 25 \\ 0 \\ \dfrac{\left\| P_\alpha \left[H_{\beta\to\alpha} \begin{bmatrix} 25 \\ 0 \\ 0 \end{bmatrix} \right] - \begin{bmatrix} -42 \\ 0 \\ 0 \end{bmatrix} \right\|}{\sin(45°)} \end{bmatrix}. \tag{6.24}$$

The third dimensional information of point $\boldsymbol{x}$ is thus obtained as 270 millimeters. Thus point $\boldsymbol{x}$ from plane α, can be represented as $\boldsymbol{x}_\alpha = \begin{bmatrix} 25 & 0 & 270 \end{bmatrix}^T$. The result has an error of less than 1 cm. While operating for larger visual regions with larger measurement units, this error is insignificant.

6.1.1. Scenario: Planes Separated by 90 **Degrees.** If the planes are at 90 degree angles to each other, then the third dimensional information of any given point is obtained by,

$$x_{\beta_3} = \|P_\beta[H_{\alpha\to\beta}P_\alpha[x]] - P_\beta[x]\|, \tag{6.25}$$

The experimental setup is shown in Figure 6.8. For the test cases shown in Figure 6.7, the α

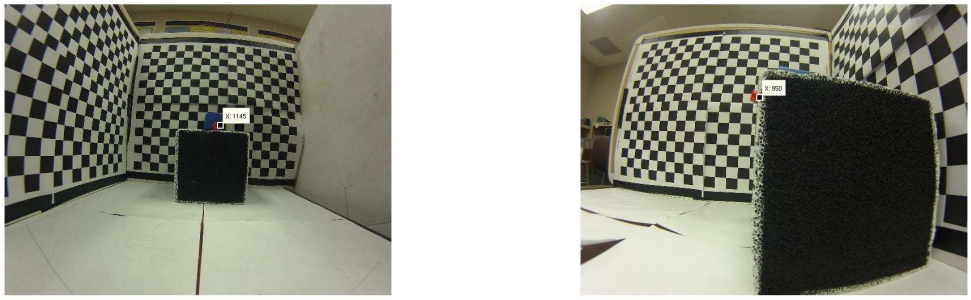

Figure 6.7. Projection of the Object from α and γ Planes

plane is assumed to be at origin $\boldsymbol{c_\alpha} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ and γ plane is at $\boldsymbol{c_\gamma} = \begin{bmatrix} 190 & 0 & 340 \end{bmatrix}^T$. The projections of point x on planes α and γ are obtained at $P_\alpha[\boldsymbol{x}] = \begin{bmatrix} 25 & 0 & 0 \end{bmatrix}^T$ and $P_\gamma[\boldsymbol{x}] = \begin{bmatrix} -20 & 0 & 0 \end{bmatrix}^T$ respectively. The angle ψ between the camera planes is 90 degrees.

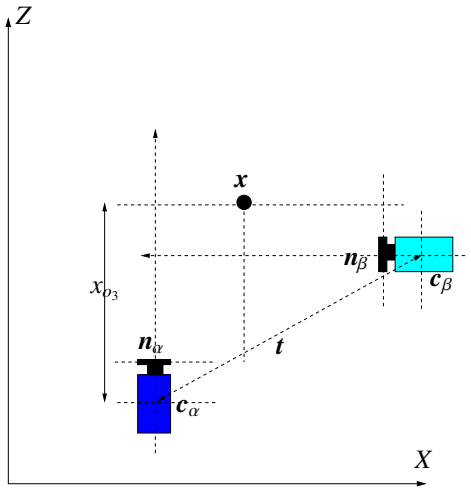

Figure 6.8. Representation of Experimental Setup for Example 2 of Unaligned Planes Method

Thus putting it in the Equation(6.22), we get,

$$\boldsymbol{x}_\alpha = \begin{bmatrix} 25 \\ 0 \\ \dfrac{\left\| P_\alpha \left[H_{\gamma\to\alpha} \begin{bmatrix} 25 \\ 0 \\ 0 \end{bmatrix} \right] - \begin{bmatrix} -20 \\ 0 \\ 0 \end{bmatrix} \right\|}{\sin(90°)} \end{bmatrix}. \tag{6.26}$$

Thus the three dimensional information about the point x from the camera plane α is $\boldsymbol{x}_\alpha = \begin{bmatrix} 25 & 0 & 280 \end{bmatrix}^T$.

Corollary 2. *When the camera planes are aligned to each other, the third dimensional information cannot be obtained.*

Proof For Equation(6.21), when the angle between two camera planes is zero, the denominator of equation also becomes zero thus,

$$x_{\alpha_3} = \frac{\|P_\beta[H_{\alpha\to\beta}P_\alpha[\boldsymbol{x}]] - P_\beta[\boldsymbol{x}]\|}{\sin(\measuredangle\nu)} = \frac{\|P_\beta[H_{\alpha\to\beta}P_\alpha[\boldsymbol{x}]] - P_\beta[\boldsymbol{x}]\|}{\sin(\measuredangle 0^\circ)} = \infty. \tag{6.27}$$

□

Hence, when camera planes are aligned, the third dimensional information cannot be determined using the Unaligned Plane Method. But the Binocular Disparity Method works well for aligned camera planes. Thus the Binocular Disparity Method can be used for cases of exception. In Section 6.2, we will see that even though the Unaligned Planes Method fails for 0 degree angular difference between camera planes, it does work for non-zero angles.

6.2. CREATING POINT-CLOUD FROM MULTIPLE POINTS

We used the test images from Middlebury stereo imaging website Scharstein et al. (2014) to test the Unaligned Plane Method. These images are considered as a benchmark for stereo imaging experiments. A set of such images are shown in Figure 6.9.

Figure 6.9. Test Images for Unaligned Planes Method

We obtained the stronger points in these images as shown in Figure 6.10 and then processed these points in each image using the point matching algorithm and retrieved the matched points from both images. 3000+ matching points were found through this process.

The setup in Scharstein et al. (2014) did not mention the distance between the cameras or the angle, so a small angle of 5 degrees and a small distance of 10 units was used to estimate the third dimensional information.

Figure 6.10. Matched Points from 2 Images

Using this method on multiple points, we generated a point-cloud. This point cloud was plotted on the three dimensional space. The three dimensional representation of Figure 6.9 can be seen in Figure 6.11 and Figure 6.12. A more elaborate set of results is also displayed in Figure 6.14.

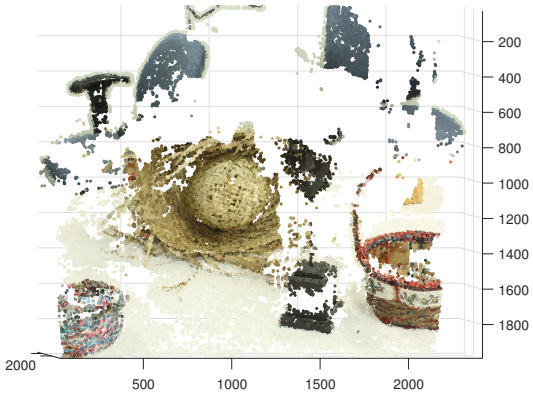

Figure 6.11. Front View of Point Cloud

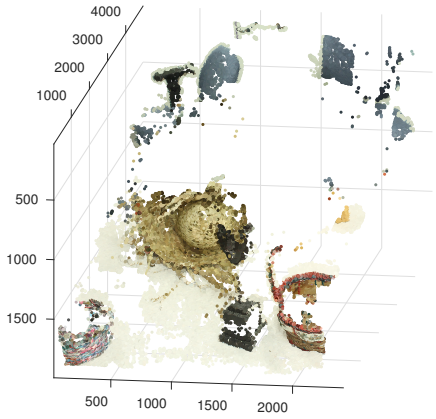

Figure 6.12. Auxiliary View of Point Cloud

We performed a second test on another data-set obtained from the same source and the original data-set which was tested with the first method. The results are provided in Figure 6.13.

As seen from Figure 6.9, both images are almost aligned with each other with a very small angle between them. Corollary 2 states that the Unaligned Planes Method does not work when the angle difference between the planes is zero. But the method works well for non-zero angles, in the case of our example, five degrees. Thus the case of aligned planes can be remedied by introducing a non-zero angle to the calculations.

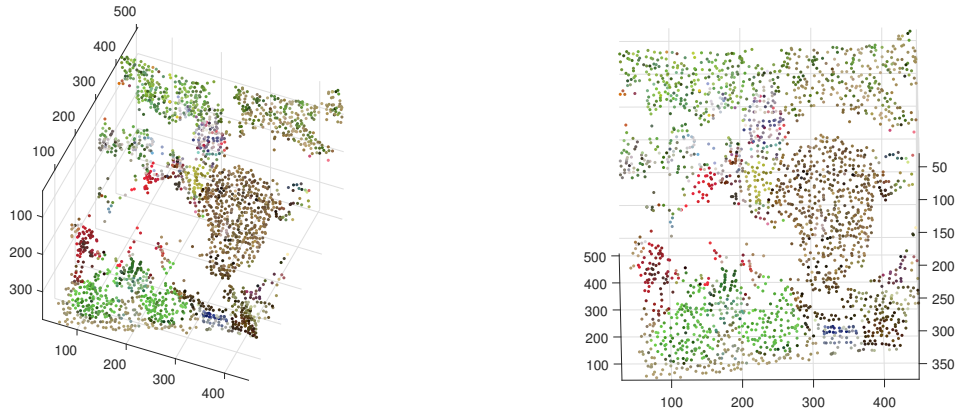

Figure 6.13. Results from Method 2, Point Cloud from (a) Front View and (b) Auxiliary View

The method was implemented using MATLAB and its timing complexity was measured. It produces the results in 0.12 seconds for Binocular Disparity Method and 0.14 seconds for the Unaligned Planes Method. Thus for a small compromise of timing complexity, more cases can be covered using Unaligned Planes Method. The performance of Unaligned Planes Method is faster than the performance observed for other third dimensional information estimation techniques such as depth from sliding projection by Chris Hermans in Hermans et al. (2009) or the multi-view stereo method introduced by Goesele in Goesele et al. (2006).

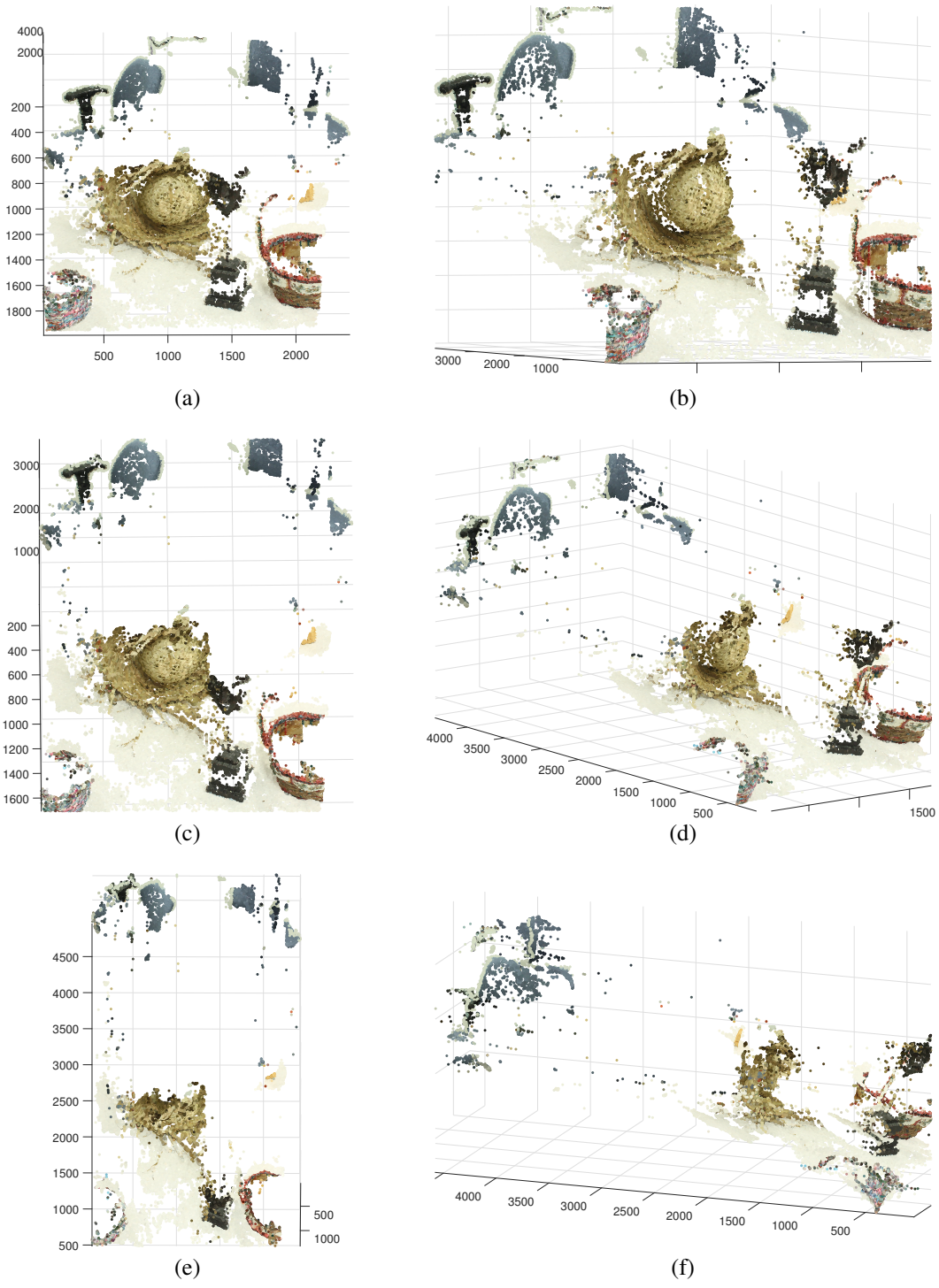

Figure 6.14. Representation of Resulted Points from Unaligned Planes Method in three Dimensional Space from, (a),(c),(e) Front to Top and (b),(d),(f) Front to Side

7. LIBRARIES FOR IMAGE PROCESSING AND STORAGE OF THE POINTS

For Localization and Map Building, as shown in Figure 1.1, we need to provide the information about the environment in terms of points/coordinates. The points then can further be processed to form lines, polygons and other complex objects. The points, for the ease of accessibility, are stored in a certain hierarchy of time-stamps and point-label combination. In this section, we will discuss variables and library functions for processing and storage of the points. The implementation is done in OpenCV using C++ and the initial prototyping was done on MATLAB.

7.1. VARIABLES AND DATA STRUCTURES TO STORE DATA

This section describes the storage related libraries and global variables that are used for this application.

7.1.1. Structure for Storing Images. OpenCV uses a class Mat as a storage container for n-dimensional dense arrays. It is used to store real or complex-valued vectors, matrices, gray-scale or color images. A custom structure for storing images used here has an object of class Mat. The structure also keeps a track of number of points matched from that image.

```
struct ImageMat{
Mat img;
int number_points_detected;
ImageMat(){
number_points_detected=0;
}
};
```

7.1.2. Structure for Storing Points. OpenCV has a point storing classes Point2f, Point3f etc. These classes are not enough to store the points and we need time-stamps and the colors too, to track the motion as a future work. The points has to be stored in the following hierarchy.

Label $\rightarrow$ Time-stamp $\rightarrow$ <Point x, y, z >

Each point at different time-stamp is matched with itself and it is stored under the same label as the same point. The overall structure looks like this

Point1 $\rightarrow t0 - t5 \rightarrow$ <Point1 x, y, z >

Point1 $\rightarrow t5 - t7 \rightarrow$ <Point1 x, y, z >

Point2 $\rightarrow t0 - t5 \rightarrow$ <Point2 x, y, z >

Point3 $\rightarrow t0 - t5 \rightarrow$ <Point3 x, y, z >

For example, in the table above, it can be seen that Point1 is stored twice but it is stored under the same label. The data is stored under different time stamps. First time-stamp goes from $t0$ to $t5$ and second time-stamp goes from $t5$ to $t7$. The point is not updated till the object actually changes its position. The data stored under points are stored in 1st, 2nd and 3rd dimensions after converting them into the global coordinate system as showed in Section 6. This structure can be used to save more data such as color of the point.

```
struct point{
string label;
time_t t;
Point3f P;
Vec3b C;
};
```

The variable C holds the color of the point and Point3f holds the 1st, 2nd and 3rd dimension of the point.

The vec3b and Point3f are the inbuilt classes provided by OpenCV.

```
typedef Vec<uchar, 3> Vec3b;
typedef Point3_<float> Point3f;
```

7.1.3. Dictionary for Storing Points. To store the points as a dictionary or a key-value pair, as in Section 7.1.2, we need to use a Hashmap from C++. The label and the time-stamp are the keys for storing the point.

```
struct P_List{
map < pair<string, time_t>, point > Points_List;
vector<string> pointlabel;
vector<time_t> timestamp;
};
```

7.2. FUNCTIONS FOR OBTAINING THIRD DIMENSION

This section describes the functions that are used to obtain the 3rd dimension.

7.2.1. Function for Matching Points from Two Camera Planes. Before we go into obtaining third dimension, first we need to identify the points for which we are going to do the processing. Not all points can be used to obtain the depth. So first, the strongest points are obtained from an image and then the points are matched with it's pair. A FLANN based matcher is used to match the points from two images. FLANN uses the Hierarchical K-means Tree for generic feature matching as in Muja and Lowe (2009). The set of points are returned in the form of Keypoint Vector.

```
int minHessian = 5;
Ptr<FeatureDetector> detector = ...
FastFeatureDetector::create(minHessian);
 std::vector<KeyPoint> keypoints_1, keypoints_2;
```

```
detector->detect(img_1, keypoints_1);
detector->detect(img_2, keypoints_2);
Mat descriptors_1, descriptors_2;
Ptr<SURF> extractor = SURF::create();
extractor->compute(img_1, keypoints_1, descriptors_1);
extractor->compute(img_2, keypoints_2, descriptors_2);
  FlannBasedMatcher matcher;
  std::vector< DMatch > matches;
  matcher.match( descriptors_1, descriptors_2, matches );
  double max_dist = 120; double min_dist = 5;
  for( int i = 0; i < descriptors_1.rows; i++ )
  { double dist = matches[i].distance;
    if( dist < min_dist ) min_dist = dist;
    if( dist > max_dist ) max_dist = dist;
  }
  printf("-- Max dist : %f \n", max_dist );
  printf("-- Min dist : %f \n", min_dist );
  std::vector< DMatch > good_matches;
  for( int i = 0; i < descriptors_1.rows; i++ )
  { if( matches[i].distance <= max(2*min_dist, 0.2) )
    { good_matches.push_back( matches[i]); }
  }
```

7.2.2. Functions for Obtaining Rotation Matrix and Projection. To obtain the third dimension, as in Section 6, we need to define a few methods or the functions to obtain Rotation Matrix and the to obtain the Projections.

```
Mat get_rotation(int phi,int psi,int theta){
Mat product1;
Mat RX = (Mat_<double>(4, 4) <<
      1,          0,           0, 0,
      0, cos(phi), -sin(phi), 0,
      0, sin(phi),  cos(phi), 0,
      0,          0,           0, 1);
    Mat RY = (Mat_<double>(4, 4) <<
      cos(psi), 0, -sin(psi), 0,
      0, 1,          0, 0,
      sin(psi), 0,  cos(psi), 0,
      0, 0,          0, 1);
    Mat RZ = (Mat_<double>(4, 4) <<
      cos(theta), -sin(theta), 0, 0,
      sin(theta),  cos(theta), 0, 0,
      0,          0,           1, 0,
      0,          0,           0, 1);
product1=RX*RY*RZ;
return product1;
}

Mat get_proj(Mat point){
Mat proj = (Mat_<double>(4, 4) <<
      0, 0, 0, 0,
      0, 0, 0, 0,
      0, 0, 1, 0,
```

```
      0, 0, 0, 1);
Mat eye4 = (Mat_<double>(4, 4) <<
      1, 0, 0, 0,
      0, 1, 0, 0,
      0, 0, 1, 0,
      0, 0, 0, 1);
return (eye4-proj)*point;
}
```

The angles ϕ,θ,ψ and angle ν are defined globally as integer variables.

7.2.3. Function for Obtaining Depth. Using the above mentioned functions, the depth is obtained by the use of function getdepth. It takes input of Matched Keypoints, two sets of keypoints from the pair of images, the image from plane α and the rotation matrices.

```
vector<Point3f> get_depth(std::vector< DMatch > v,...
...std::vector<KeyPoint> keypoints_1,std::vector<KeyPoint> ...
... keypoints_2,Mat img,Mat R1,Mat R2){
Point3f T1=c1_act-c2_act;
R1.at<double>(0,3)=T1.x;
R1.at<double>(1,3)=T1.y;
R1.at<double>(2,3)=T1.z;
Mat c1_actm = (Mat_<double>(4, 1) <<
c1_act.x,
c1_act.y,
c1_act.z,
1);
Mat c2_actm = (Mat_<double>(4, 1) <<
c2_act.x,
```

```
c2_act.y,
c2_act.z,
1);
Mat c1_act_2=R2*c1_actm;
Mat c2_act_2=R2*c2_actm;
Mat T2=c1_act_2-c2_act_2;
R2.at<double>(0,3)=T2.at<double>(0,0);
R2.at<double>(1,3)=T2.at<double>(1,0);
R2.at<double>(2,3)=T2.at<double>(2,0);
for(int i=0;i<v.size()-1;i++)
{
int pt1_x=keypoints_1[ v[i].trainIdx ].pt.x-1024;
int pt2_x=keypoints_2[ v[i].queryIdx ].pt.x-1024;
Mat temp1 = (Mat_<double>(4, 1) <<
pt1_x,
0,
0,
1);
Mat temp2 = (Mat_<double>(4, 1) <<
pt2_x,
0,
0,
1);
Mat p21=R1*temp1;
Mat p2_1=get_proj(p21);
Mat temp3 = (Mat_<double>(4, 1) <<
pt1_x,
```

```
0,
norm(p2_1,temp2,NORM_L2)/sin(nu),
1);
Mat p_act=R1*temp3;
p_act.at<double>(0,0)=keypoints_1[ v[i].trainIdx ].pt.x;
p_act.at<double>(1,0)=keypoints_1[ v[i].trainIdx ].pt.y;
}
}
```

7.3. OTHER FUNCTIONS

This section describes the functions that are used as an extra support for our application.

7.3.1. To Retrieve Point from Storage. The function takes the list of points, the point-label and the time-stamp and then returns the point information of that point from permanent storage.

```
point retreivepoints(P_List &P, time_t t, string key);
```

7.3.2. To Erase Point from Storage. This function takes the list of points, the point-label and the time-stamp and then erases the point from permanent storage.

```
void erasepoints(P_List &P, time_t t, string key);
```

7.3.3. Finding Color of the Point. This vector takes the set of keypoints, the id of the point, original image from camera plane α and returns the color vector.

```
Vec3b get_color(vector<KeyPoint> keypoints_1,int i,Mat img)
```

This function returns set of points in 3-dimensional space that were matched earlier using point-matching algorithm. These points can further be stored in the dictionary as a key-value pair as mentioned in Section 7.1.3

8. CONCLUSION AND FUTURE WORK

The work presented in this thesis presents a depth determination approach for in-motion control systems that is in possession of single camera. The camera perceives surrounding data by capturing the projection on camera planes from multiple positions. The change in position and difference in projections gives us enough information about the object that helps us determine the exact depth of the object from any camera plane. The method can be repeated to all the points in the image. The depth of these points in three dimensional space gives the control system enough knowledge about the depth of its surrounding.

The Unaligned Planes Method was compared with the already existing Binocular Disparity Method. The pre-existing method was limited, as it needed planes to be completely aligned. The newly discovered method has a higher computational requirement than the previous method by a small factor. The Unaligned Planes Method gives us a wider scope of analysis and mapping than the pre-existing methods such as Binocular Disparity Method. Thus the increase in computationally expensiveness is also justified.

At each step, the algorithm produces more than 1000 matched points and these points are stored in the storage board. These points accessed constantly by the algorithm to see if the point that is matched is same or not. The hierarchical nature of stored data avoids the iterative search for the matched point in the storage. This saves the system some processing time.

The developed system was able to be controlled from remote locations as it is connected to the internet through a secure gateway. If further development requires more computational power, the control panel can serve as an API. Using this API, the data can be accessed from any cloud service. By processing this data, any feedback signal can be provided back from such cloud based service.

In this project, only the points and their depth is obtained using the set of images. More number of points gives us more idea about the surrounding. But the actual structural data about the surrounding is still not obtained. Thus, in further work, using these point data, complex objects including lines, planes, three dimensional structures can be constructed.

The stored points also stores the time at which the point was recorded. Using this time information, it is possible to track a moving object and its trajectory.

APPENDIX A

MAIN SIMULATION FILE : MAIN.M

0.1. MATLAB File to Test the Unaligned Planes Method.

```
% I1=imread('left.png');
% I2=imread('right.png');
% I1=imread('view1.png');
% I2=imread('view2.png');
I2=imread('im01.png');
I1=imread('im12.png');

figure
imshowpair(I1, I2, 'montage');
title('Original Images');
 load upToScaleReconstructionCameraParameters.mat
% I1 = undistortImage(I1, cameraParams);
% I2 = undistortImage(I2, cameraParams);
figure
imshowpair(I1, I2, 'montage');
title('Undistorted Images');

%%
roi = [30, 30, size(I1, 2) - 30, size(I1, 1) - 30];
% Detect feature points
imagePoints1 = detectMinEigenFeatures(rgb2gray(I1), 'ROI',...
 roi, 'MinQuality', 0.00000001);

% Visualize detected points
```

```
figure
imshow(I1, 'InitialMagnification', 50);
title('150 Strongest Corners from the First Image');
hold on
plot(selectStrongest(imagePoints1, 3500));

% Create the point tracker
tracker = vision.PointTracker('MaxBidirectionalError', 4,...
 'NumPyramidLevels', 15);

% Initialize the point tracker
imagePoints1 = imagePoints1.Location;
initialize(tracker, imagePoints1, I1);

% Track the points
[imagePoints2, validIdx] = step(tracker, I2);
matchedPoints1 = imagePoints1(validIdx, :);
matchedPoints2 = imagePoints2(validIdx, :);

% Visualize correspondences
figure
showMatchedFeatures(I1, I2, matchedPoints1, matchedPoints2);
title('Tracked Features');

figure;
points3d=depth_estimate(matchedPoints1,matchedPoints2,...
[0,0,0,1],[10,0,0,1],0,5,0);
```

```
points3d=points3d/50;
numPixels = size(I1, 1) * size(I1, 2);
allColors = reshape(I1, [numPixels, 3]);
colorIdx = sub2ind([size(I1, 1), size(I1, 2)],...
 round(matchedPoints1(:,2)),round(matchedPoints1(:, 1)));
color = allColors(colorIdx, :);
ptCloud=pointCloud(points3d,'Color', color);
%ptCloud=pointCloud(points3d);
pcshow(ptCloud, 'VerticalAxis', 'y', 'VerticalAxisDir', 'down',...
'MarkerSize', 45);
```

APPENDIX B

SUPPORTING FUNCTIONS

0.1. MATLAB Function to get Projection.

```
function proj=get_proj(point)
    proj=(eye(4)-[0,0,1,0]'*[0,0,1,0])*point;
end
```

0.2. MATLAB Function to get Rotation Matrix.

```
function R=get_rotation(phi,psi,theta)
    Rx=[1,0,0;0,cosd(phi),-sind(phi);0,sind(phi),cosd(phi)];
    Ry=[cosd(psi),0,sind(psi);0,1,0;-sind(psi),0,cosd(psi)];
    Rz=[cosd(theta),-sind(theta),0;sind(theta),cosd(theta),0;0,0,1];
R=Rx*Ry*Rz;
end
```

0.3. MATLAB Function to Estimate Depth.

```
function [ points3d ] = depth_estimate( point_set_1,point_set_2,...
c1_act,c2_act,alpha,beta,theta )
pending=point_set_1;
point_set_1(:,1)=point_set_1(:,1)-1024;
point_set_2(:,1)=point_set_2(:,1)-1024;
point_set_1(:,2)=point_set_1(:,1)-768;
point_set_2(:,2)=point_set_2(:,1)-768;
R1=get_rotation(alpha,beta,theta);
T1=c2_act-c1_act;
H1=[R1,T1(1:3)';0,0,0,1];

R2=get_rotation(alpha,-1*beta,theta);
```

```
c1_act_2=R2*c1_act(1:3)';
c2_act_2=R2*c2_act(1:3)';
T2=c1_act_2-c2_act_2;
H2=[R2,T2;0,0,0,1];
points3d=[pending zeros(length(point_set_1),1)];
%points3d=[pending 0];
for i=1:length(point_set_1)
    temp1=[point_set_1(i,1),0,0,1];
    temp2=[point_set_2(i,1),0,0,1];
   p21=H1*temp1';
   p2_1=get_proj(p21);
   dist2=abs(sqrt(sum((p2_1-temp2').^2))/sind(beta));
   temp3=temp1;
   temp3(3)=dist2;
   p_act=H1*temp3';

   points3d(i,3)=p_act(3);
end

end
```

BIBLIOGRAPHY

Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

Edward H Adelson and John YA Wang. Single lens stereo with a plenoptic camera. *IEEE transactions on pattern analysis and machine intelligence*, 14(2):99–106, 1992.

Chris Hermans, Yannick Francken, Tom Cuypers, and Philippe Bekaert. Depth from sliding projections. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1865–1872. IEEE, 2009.

Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J Gortler, and Leonard McMillan. Image-based visual hulls. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 369–374. ACM Press/Addison-Wesley Publishing Co., 2000.

Greg Slabaugh, Ron Schafer, and Mat Hans. Image-based photo hulls. In *3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on*, pages 704–862. IEEE, 2002.

Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in Neural Information Processing Systems*, pages 1161–1168, 2005.

Carlos Hernández Esteban and Francis Schmitt. Silhouette and stereo fusion for 3d object modeling. *Computer Vision and Image Understanding*, 96(3):367–392, 2004.

Adam Shadbolt. From 2d photographs to 3d caricatures. *Undergraduate project Dissertation, Department of Computer Science, University of Sheffield (June 2003)*, 2003.

Ramesh Jain, Rangachar Kasturi, and Brian G Schunck. *Machine vision*, volume 5. McGraw-Hill New York, 1995.

Ludovic Apvrille, Tullio Tanzi, and Jean-Luc Dugelay. Autonomous drones for assisting rescue services within the context of natural disasters. In *General Assembly and Scientific Symposium (URSI GASS), 2014 XXXIth URSI*, pages 1–4. IEEE, 2014.

Pravin Dhake. A real time operating system based test-bed for autonomous vehicle navigation. 2007.

Robert S Woodley and Levent Acar. A testbed system for nonlinear or intelligent control. In *American Control Conference, 1999. Proceedings of the 1999*, volume 5, pages 3441–3445. IEEE, 1999.

PFM Aaeon's. 945c cpu boards: Pc/104 cpu module with onboard intel atom n270 processor.

Krishnan Raghavan. Computer vision libraries for trailer truck testbed using open source computer vision libraries. 2014.

Sensoray. Sensoray model 911 user manual. 2005.

Bruce Timberlake. Building a lamp server, 2010.

Paolo Mantegazza, EL Dozio, and S Papacharalambous. Rtai: Real time application interface. *Linux Journal*, 2000(72es):10, 2000.

Paul J Besl. Active, optical range imaging sensors. *Machine vision and applications*, 1(2): 127–152, 1988.

Gines Benet, Francisco Blanes, José E Simó, and Pascual Pérez. Using infrared sensors for distance measurement in mobile robots. *Robotics and autonomous systems*, 40(4): 255–266, 2002.

Emanuele Trucco and Alessandro Verri. *Introductory techniques for 3-D computer vision*, volume 201. Prentice Hall Englewood Cliffs, 1998.

Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition*, pages 31–42. Springer, 2014.

Mikko Kytö, Mikko Nuutinen, and Pirkko Oittinen. Method for measuring stereo camera depth accuracy based on stereoscopic vision. In *IS&T/SPIE Electronic Imaging*, pages 78640I–78640I. International Society for Optics and Photonics, 2011.

Michael Goesele, Brian Curless, and Steven M Seitz. Multi-view stereo revisited. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2402–2409. IEEE, 2006.

Marius Muja and David G Lowe. Flann, fast library for approximate nearest neighbors. In *International Conference on Computer Vision Theory and Applications (VISAPP09)*, volume 3. INSTICC Press, 2009.

VITA

Aditya Thakur was born in Mumbai, India. After completing his schoolwork at C. K. Thakur High School in India in 2010, Aditya entered S.K.N. College of Engineering affiliated to University of Pune in Pune. He received a Bachelor of Engineering with a major in Electronics and Telecommunication Engineering from University of Pune in May 2014. In July 2017, he Completed his MS degree in Electrical Engineering from the Missouri University of Science and Technology at Rolla, MO, USA.