MASTER

3D map creating using the structured light technique for obstacle avoidance

Rocque, M.J.

Award date:
2011
3D Map Creation using the Structured Light Technique for Obstacle Avoidance

Author:
Mukul Rocque

Supervisors:
Prof. Dr. Gerard de Haan (TU/e)
Dr. Frank van Heesch (Philips Research)
Abstract

Obstacle avoidance is an essential part of robotic navigation in which the robot, based on some global path planning strategy, has to reach a certain goal position. In robotic navigation the problem of obstacle avoidance refers to the determination of a local navigation strategy. Here, the navigation strategy is based on the detection of obstacles which is made depending on the local environment of the robot. This thesis presents the development and implementation of a novel obstacle avoidance system in which the detection of obstacles is based on the use of computer vision implemented using the principle of structured light based depth map creation. This allows the detection of an obstacle of at least 20mm x 18mm at the projection distance of 675mm. The obstacle avoidance system consists of a robotic platform called Create from iRobot with a vision system, made up of a camera-projector pair, all controlled by an embedded computing unit called BeagleBoard. The BeagleBoard, based on inputs from the vision system, computes a depth map of the observed environment relative to the robotic platform. This depth map is then used to determine the 3D coordinates of the observable environment which define the presence or absence of obstacles. An obstacle is defined taking into account the physical parameters of the entire system, which then governs the movements of the entire robotic platform. The system is tested on a number of real world obstacles and a variety of tests are carried to determine the system performance for varying distances, under varying ambient light conditions and different obstacle surfaces. The deviations obtained with respect to the real world depths are less than 10mm for the depth range between 600 to 800mm.

Keywords: Obstacle avoidance, structured light, depth map.
Dedicated to my parents and my brother
who have always been a source of inspiration and motivation
in every step of my life.
Acknowledgements

I would firstly like to express my deepest gratitude to my graduation supervisor Prof. Dr. Gerard de Haan for giving me the opportunity to work at Philips Research Eindhoven on this project. He has been instrumental in igniting a passion for Video and Image Processing in me and I hope to live up to his expectations in my professional career in the future.

My sincerest thanks to my company supervisor Dr. Frank van Heesch. I feel lucky to have him as my supervisor, who has never hesitated to find time to listen and help me solve my problems. He has always encouraged me to be innovative while tackling any problems I faced during my assignment and I feel a lasting gratitude for all his words of encouragement and support. I thank him for having the faith in my abilities to let me work on his project and I hope I have not disappointed him in any manner. He has not only been a great supervisor but also a superb friend.

My special thanks to all the researchers at Philips Research Eindhoven and my teachers at the Eindhoven University of Technology who have always helped me with their timely advice whenever needed.

I would also like to thank Mr. Sachin Bhardwaj for his review of my thesis and his continuous encouragement and advice during the duration of my graduation project. He has been a great role model and a tremendous source of motivation. I thank my friends Ketan Pol, Mayur Sarode, Mohit Kaushal, Pratik Sahindrakar, Shubhendu Sinha and Vidya Vijayasankaran for all the support, motivation and help they have provided. My special thanks to my best friend Hrishikesh who has always provided me with any necessary technical expertise I may have lacked. His words have kept me motivated and smiling and I will always regard him as one of the more important reasons for the successes I have achieved.
# Contents

1 Introduction .................................................. 2  
   1.1 Problem description ........................................ 2  
   1.2 Related work ............................................. 3  
   1.3 Report Organization ........................................ 4  

2 Depth estimation ............................................. 5  
   2.1 Depth estimation principle ................................ 5  
      2.1.1 Human vision .......................................... 5  
      2.1.2 Correspondence problem ................................ 6  
      2.1.3 Stereoscopic machine vision ............................ 7  
      2.1.4 Structured light based depth computation .......... 7  
         2.1.4.1 Simplified principle ............................. 8  
         2.1.4.2 Mathematical analysis ........................... 8  
   2.2 Structured light projection pattern ....................... 11  
      2.2.1 Classification ....................................... 11  
         2.2.1.1 Number of dimensions ........................... 11  
         2.2.1.2 Multiplexing strategy ........................... 12  
         2.2.1.3 Coding strategy ................................ 13  
         2.2.1.4 Pixel depth ...................................... 14  
      2.2.2 Finalized pattern ..................................... 14  
         2.2.2.1 Pattern encoding ................................. 15  
         2.2.2.2 Pattern decoding ................................. 17  

3 System Design .............................................. 18  
   3.1 System setup ............................................. 18  
      3.1.1 Embedded computing unit ............................. 19  
      3.1.2 Operating System .................................... 19  
      3.1.3 Robotic platform ..................................... 19  
      3.1.4 Projection system ................................... 20  
         3.1.4.1 Placement ....................................... 20  
         3.1.4.2 Orientation ..................................... 21  
         3.1.4.3 Projection distance ............................. 21  
      3.1.5 Camera system ....................................... 21  
         3.1.5.1 Placement ....................................... 22  
      3.1.6 Finalized hardware setup ............................. 23  
   3.2 Calibration ............................................... 24
3.2.1 System calibration based on Zhang's method ........................................... 24
3.2.2 Simplified calibration ............................................................................... 27
3.3 Depth estimation algorithm ........................................................................ 29
3.3.1 Pattern projection ...................................................................................... 29
3.3.2 Image capture ............................................................................................ 29
3.3.3 Solution to correspondence problem ......................................................... 29
   3.3.3.1 PreProcessing ..................................................................................... 30
   3.3.3.2 Grid primitive segmentation ............................................................... 31
   3.3.3.3 Grid primitive center location ............................................................ 31
   3.3.3.4 Grid primitive identification ............................................................... 31
   3.3.3.5 4-adjacent neighbors identification ................................................... 33
   3.3.3.6 Projector coordinate computation ...................................................... 34
   3.3.3.7 Validity check of computed projector coordinate ............................... 35
3.3.4 Depth map computation ............................................................................ 35
3.3.5 Obstacle detection ..................................................................................... 36
3.3.6 Robotic platform control .......................................................................... 36
4 Results ............................................................................................................. 37
4.1 Distance tests ............................................................................................... 37
   4.1.1 Planar depth ......................................................................................... 38
   4.1.2 Multiple planes ..................................................................................... 38
   4.1.3 Curved surface .................................................................................... 39
   4.1.4 Shiny surface ...................................................................................... 40
4.2 Ambient light intensity variation .................................................................. 40
4.3 Effect of depth on measurement accuracy ....................................................... 41
4.4 Obstacle detection ......................................................................................... 42
4.5 Complete system integration ........................................................................ 43
4.6 Execution time distribution .......................................................................... 44
5 Conclusion and Future Work ........................................................................ 45
List of Figures

2.1 Disparity in human vision ............................................. 6
2.2 Correspondence problem in stereovision ............................ 6
2.3 Depth computation ..................................................... 7
2.4 Hall’s depth computation for coplanar planes ......................... 8
2.5 Angles of rotation for non-coplanar planes ......................... 10
2.6 Coding strategy based on number of dimensions ..................... 11
2.7 Multiplexing strategies ................................................. 12
2.8 Coding strategies .......................................................... 13
2.9 Coding strategy based on pixel depth .................................. 14
2.10 Projected Pattern .......................................................... 16
2.11 Primitives and spacing .................................................. 16
2.12 Decoding window .......................................................... 17

3.1 Project overview .......................................................... 18
3.2 Create robotic platform .................................................. 19
3.3 Projector setup .............................................................. 21
3.4 Single frame acquisition steps in µEye cameras ....................... 22
3.5 Final System Setup ......................................................... 23
3.6 Calibration technique based on Zhang’s method ....................... 24
3.7 Calibration images .......................................................... 25
3.8 Projector - camera orientation ........................................... 26
3.9 Simplified calibration technique ......................................... 27
3.10 Calibration images placed at different depths ......................... 28
3.11 Calibration results ........................................................ 28
3.12 Solving the correspondence problem .................................... 29
3.13 Binarization ................................................................. 30
3.14 Filtered output .............................................................. 30
3.15 Segmented grid primitives ............................................... 31
3.16 Center location using bounding box ................................... 31
3.17 Grid primitive location on image plane ................................ 32
3.18 Grid primitive identification using template matching .............. 32
3.19 Detecting 4-adjacent neighbors .......................................... 33
3.20 Computing the location of grid primitive in structured light pattern ........................................................................ 34
3.21 Shift of Origin to compute 3D coordinates ........................... 36

4.1 Depth map for planar wall at perpendicular depth of 700mm from system ............................................. 38
4.2 Depth map for perpendicular plane surfaces from system .......... 38
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3 Depth map for curved surface from system</td>
<td>39</td>
</tr>
<tr>
<td>4.4 Depth map for metallic dustbin</td>
<td>40</td>
</tr>
<tr>
<td>4.5 Effect of light intensity variation on robustness</td>
<td>40</td>
</tr>
<tr>
<td>4.6 Error variation with distance</td>
<td>41</td>
</tr>
<tr>
<td>4.7 World coordinates for obstacle detection</td>
<td>42</td>
</tr>
<tr>
<td>4.8 Platform movement with obstacle</td>
<td>43</td>
</tr>
<tr>
<td>4.9 Relative time consumption for depth computation</td>
<td>44</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th></th>
<th>Table Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Intrinsic parameter</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>Projector extrinsics</td>
<td>26</td>
</tr>
</tbody>
</table>
The design of an autonomous mobile navigation system is a highly challenging problem in the fields of computer vision and robotic research. Such systems require mechanisms capable of making decisions in complex environments. One of the core problems in autonomous robotic navigation is obstacle avoidance.

Robotic navigation can be broadly categorized into global and local navigation. Global navigation refers to the ability of a robot to move to a desired goal position using some path planning strategy from a starting point to the goal. Local navigation, on the other hand, refers to the ability of the robot to determine its position relative to stationary or moving objects in the local environment and react accordingly. Obstacle avoidance refers to this local navigation problem where navigation decisions are made depending on the presence or absence of obstacles in the observable environment of the system.

The definition of an obstacle is specific to the obstacle avoidance system which includes the physical design parameters of the system and the specific environment the system is intended to be operated in. Obstacle avoidance relates to, firstly, the detection of individual objects in the environment of the system, followed by the classification of the detected objects as obstacles and finally, computing navigation decisions to avoid a detected obstacle. To ensure the above functionality the robotic system must be able to ‘see’ its surroundings to determine how far certain obstacles are and then navigate accordingly to avoid the obstacles. The detection of objects requires the creation of a depth map. A depth map represents the per-pixel depth of the observed environment relative from the obstacle detection system [1].

1.1 Problem description

This thesis focuses on a very specific application of autonomous robotic navigation in its use in household appliances. The project is aimed at an even more specific case of automated robotic vacuum cleaners which is of high interest for Philips Research, Eindhoven. For such systems the estimation of the positions of obstacles relative to the system plays a major role. An obstacle is defined as any element in the observable environment that the system is obstructed by during navigation. This includes elements such as walls, chair or table legs through which the system cannot pass or elements such as thick wires or carpets the system cannot pass over.

Hence, the goal is to design and implement an obstacle avoidance system for a robotic vacuum cleaner prototype using embedded components. It should be capable of the depth map creation of its immediate observable environment which can then be used in autonomous navigation in typical household environments.
1.2 Related work

The related works explored in the current project relate to existing techniques and implementations of depth measurement devices, navigation systems and vacuum cleaners. Depth estimation can be performed using various techniques depending on the underlying technology. Typical technologies use the time of flight principle in reflection based depth measurement of ultrasonic signals and light beams, and principles of triangulation in the case of vision based depth measurement. In techniques based on the time of flight principle using ultrasonic signals, an ultrasonic pulse is generated and transmitted, and the time required for the pulse to travel back after reflection gives an estimation of the depth [2]. LIDAR (Light Detection and Ranging) based systems also employ the time of flight principle but instead of using ultrasound pulses work with laser pulses [3].

Various autonomous navigation systems have been developed in the past using different sensors (eg:- ultrasonic sensors, infrared sensors, GPS, LIDAR, cameras etc. or combinations of these) [2][3][4][5]. However, more and more approaches are becoming vision based. Such systems have cameras as the main sensors and are favored due to the inherent low cost, weight and energy consumption. Furthermore, the image sensors provide much more information than the traditionally used sensors thereby enabling more flexibility.

A number of autonomous robotic vacuum cleaners exist in the market. The Samsung Navibot [6] uses infrared depth sensors but only for the detection of cliffs to avoid falling over, for example over stairs, and infrared cameras facing the ceiling used for mapping its position within a room. The Trilobite vacuum cleaner from Electrolux uses ultrasound sensors for depth determination apart from the cliff detecting infrared sensors. The iRobot Roomba series of vacuum cleaners also have the cliff depth sensors but contain obstacle detecting pressure sensor bumpers which on collision with an obstacle cause the robot to change direction [7][8]. The robot uses random movements to cover the entire area of the environment. Such mechanisms cause a lot of areas to be missed or unnecessarily navigated too often and increase the overall cleaning time for a given confined area.

A widely known application of structured light based depth map creation is in the motion sensing input device Kinect by Microsoft [9]. It consists of an IR projector-camera pair used for the depth determination from PrimeSense Ltd., an RGB camera for face recognition and an array of microphones for advanced speech recognition. The IR projector-camera pair allows for the use of the Kinect in ambient light conditions. The structured light pattern used in Kinect is a non-periodic speckle pattern produced by the mutual local interference of partially coherent beams as given by Z. Zalevsky et al. in [10]. This pattern allows for the determination of the range and 3D mapping of an object from a reference plane. The depth calculation technique employed in Kinect has been documented by K. Konolige et al. in [11]. The identification of every pixel in the IR image is performed using a correlation window. The depth is then calculated using triangulation by comparing the local pattern at a particular pixel with a pre-memorized pattern at that pixel. Mention of this method is also found in one of the patent applications by Prime Sense Ltd. in [12].

This thesis aims at using vision based approaches in the detection and subsequent avoidance of obstacles. This further enables the possibility of creating maps of household environments for every robotic system which can over the operation time be optimized to maximize the area coverage and minimize the time required for an effective coverage of the intended environment.
1.3 Report Organization

This thesis details the various aspects involved in the design and implementation of an automated obstacle avoidance system. Chapter 2 details the principle of depth computation using computer vision. Furthermore, it also contains an overview of the various structured light patterns available in literature and ends with the pattern finalized for the current project along with its encoding and decoding algorithms. Chapter 3 details the selection criteria for the various components of the system and the system setup using those components. This chapter also contains the calibration methodologies followed, the calibration results obtained and details the algorithm used for the obstacle avoidance. Chapter 4 gives the results obtained using the system for various object surfaces, depths, light conditions and gives the relative time consumption of the various parts of the algorithm to solve the correspondence problem. Chapter 5 contains the overall conclusions and future directions possible for this project.
Depth estimation techniques based on the time of flight principles of ultrasonic or light pulses suffer from a number of drawbacks. In systems employing such techniques reflections from objects are heavily affected by environmental parameters such as temperature, humidity etc. These affect the velocity of the ultrasonic pulse thereby introducing errors in depth measurement. Furthermore, absorptions and refractions of the ultrasound pulse depending on the objects surface introduce additional inaccuracies in depth estimation. Due to the use of the same underlying principle as used in ultrasound based systems, LIDAR also suffers from similar problems. Systems based on time of flight principle require either a number of pulse generator-receiver pairs or complex reflection systems to obtain higher spatial resolutions making such systems expensive.

Vision based depth measurement, on the other hand, is less affected by environmental parameters affecting techniques based on the time of flight computation. It refers to the use of cameras and image processing algorithms for depth measurement. The depth information is present due to the disparate views as observed from various camera viewpoints and the depth can be computed using triangulation techniques. The following sections give detailed descriptions of depth estimation principles using vision based technology. In subsequent sections a classification of special patterns called structured light patterns used for depth measurement is given. Finally, the encoding and decoding strategies of a pattern specific to the current project of obstacle avoidance is described in detail.

2.1 Depth estimation principle

The solution to the problem of vision based depth estimation can be obtained through different approaches all related to the triangulation of light rays. Triangulation refers to the computation of depth with the use of geometry and is better explained through examples of human vision.

2.1.1 Human vision

The principle of depth estimation for obstacle avoidance is comparable to the stereopsis principle employed in e.g. human vision. Stereopsis is the process, in visual perception, that leads to the perception of stereoscopic depth. This is the sensation of depth that emerges from the disparity between two different projections of the world on the human retinas.

Figure 2.1 shows an example of visual disparity as observed while viewing a single world point (Point 1 or Point 2) from two different viewpoints or in the case of humans - the eyes. Visual disparity occurs due to the projection of light reflected from a world point at different
locations in the human eyes’ retinas. This location disparity translates to angular variation and with a known fixed baseline between the two eyes, the brain is able to compute the depth of any point identified in both the retinal views. This disparity reduces as light reflected from far away objects reaches the retinae, as shown in Figure 2.1, and thus further an object from the eyes the lesser accurate human depth perception becomes. In complex world environments the problem of correspondence becomes critical to solve, a situation much different from the simple single point scenario shown in the figure above.

2.1.2 Correspondence problem

The correspondence problem is the problem of ensuring that the projections of the same world point on the two retinas are taken into account for disparity based depth estimation [13].

Figure 2.2 shows the correspondence problem as faced in depth estimation. The two cameras observe two different viewpoints of the object. The dot shown on the object represents a specific region in space which can be viewed at particular coordinates in the left camera image plane. The correspondence problem now relates to the problem of identifying this same region in the other image plane.
2.1.3 Stereoscopic machine vision

The principle of stereopsis can be applied to machine or computer vision with the use of two or more cameras [14][15][16]. The solution to the correspondence problem is performed using techniques such as feature extraction and feature matching between the two or more views. Such features are typically edges or corners that are searched for within the views. The main drawback of this method is that such a solution to the correspondence problem is often computationally expensive and due to the placements of the cameras or the observable environments, at times, no features might be available for depth map computation. Furthermore, another drawback of the stereoscopic based systems is the complication increase in low light conditions. The system, thus, inherently requires ambient light for the correct solution to the correspondence problem. Even though stereoscopic vision has its drawbacks, in applications with the desired light conditions and the presence of appropriate features, depth maps with high resolution and accuracy can be obtained.

2.1.4 Structured light based depth computation

The use of a structured light based depth computation system highly alleviates the complexity of the correspondence problem. A structured light based system is an active system, obtained by replacing one of the cameras in a stereoscopic vision system by a projector capable of projecting a desired pattern on the environment to be reconstructed.

In a structured light projection pattern the correspondence is made explicit by projecting defined patterns on the environment to be reconstructed. The pixel coordinates in such a pattern are easily decoded to find their position in the projected pattern. Figure 2.3 shows the principle of depth computation in a structured light based system. As can be seen, using a single point projection pattern, a projected point $P_1$ when projected on an object at distance $Z_1$ is imaged at the point $C_1$. Since there is only a single point projected the correspondence is trivially solved. Extending the concept to patterns with multiple points, with a mechanism to identify every pixel position in the projection plane, the correspondence is obtained by identifying that
pixel in the imaging plane. The complexity, robustness, accuracy and resolution of a structure light based depth computing system depend on the selected structured light pattern. This is analyzed in depth in Section 2.2.

2.1.4.1 Simplified principle

The simplified principle of depth estimation using structured light can be explained with reference to Figure 2.3. Here, a camera is assumed to be simplified to a pinhole model and a projector to be originating from a point source (an inverse pinhole-camera model). The projection and camera planes are at a distance focal length in front of their respective focus points. A structured light pattern point \( P_1 \) projected at a distance \( Z_1 \), is imaged at the ray-plane intersection point on the camera plane. For varying depths of a particular projection point \( (Z_1, Z_2, Z_3) \), the imaged point is always found to lie on a line \( L_1 \) in the camera plane. With the knowledge of the \( L_1 \), the location of an imaged point on the line directly corresponds to the depth. Here, depth is defined as the perpendicular distance of the world point from a fixed point in the depth computation system. To simultaneously compute the depth of multiple points in an observable scene a structured light pattern with multiple points can be used.

2.1.4.2 Mathematical analysis

A depth map creation consists of the determination of the distance of the observable environment relative to the detection system as explained earlier. This depth determination can be performed according to the analysis by Hall et al. in [17]. The formulation provided is for coplanar camera and projection planes and needs to be generalized to allow for a non-coplanar orientation of the projection plane with respect to the camera plane.

![Figure 2.4: Hall’s depth computation for coplanar planes](image)

The center of the camera plane is assumed to be the origin of the global coordinate system. The focal point of this imaging system is at \( F = (0, 0, f_1) \), with \( f_1 \) as the focal length. According to ray-plane intersection, any image point must lie on the ray between the focal point and the object point given by:

\[
P_1 = F + \alpha (O - F)
\] (2.1)
Here, \( P_1 \) represents the coordinates of the object point on the image plane with respect to the global origin - the center of the camera plane. Equation 2.1 is expanded as:

\[
\begin{bmatrix}
  x_i \\
  y_i \\
  0
\end{bmatrix}
= \begin{bmatrix}
  0 \\
  0 \\
  f_1
\end{bmatrix}
+ \alpha \begin{bmatrix}
  x_o \\
  y_o \\
  z_o - f_1
\end{bmatrix}
\]

(2.2)

where, \( O = (x_o, y_o, z_o)^t \) represents the object point and

\[
\alpha = \frac{f_1}{f_1 - z_o}
\]

(2.3)

From equations 2.2 and 2.1 the object coordinates can be computed as:

\[
x_o = \left( \frac{f_1 - z_o}{f_1} \right) x_i \quad \text{and} \quad y_o = \left( \frac{f_1 - z_o}{f_1} \right) y_i
\]

(2.4)

Equation 2.4 can be modified assuming a virtual image plane in front of the focal point. This allows the direct use of image plane coordinates to compute the height and width dimensions from the depth value with the knowledge of the intrinsic parameters of the camera. \( x_i \) and \( y_i \) are the image coordinates and to compute the world coordinates need to be converted taking into account its relative position to the principal point of the camera \((CC_x, CC_y)\), the pixel pitch and the direction of x and y axes. For the orientation as shown in Figure 2.4 the object coordinates with respect to the global origin as the camera center are given as:

\[
x_o (\text{mm}) = \left( \frac{z_o - 2 \times f_1}{f_1} \right) \times (CC_x - x_i) \times \text{PixelPitch}
\]

\[
y_o (\text{mm}) = \left( \frac{z_o - 2 \times f_1}{f_1} \right) \times (CC_y - y_i) \times \text{PixelPitch}
\]

(2.5)

Assuming \( O_2 = (x_2, y_2, z_2)^t \) represents the center of the projection plane with respect to the global origin, \( F_2 = (0, 0, f_2)^t \) represents the focal point of the projection plane with respect to the local origin or \( F_2 = (x_2, y_2, z_2 + f_2)^t \) with respect to the global origin, and the projected point corresponding to the imaged point \( P_1 \) is represented as \( \hat{P}_2 = (x_2 + x_{2i}, y_2 + y_{2i}, z_2)^t \) with respect to the global origin. The coordinate \((x_{2i}, y_{2i}, 0)^t\) represents the point corresponding to the imaged point \( P_1 \) with respect to the origin of the projection system.

The orientation of a non-coplanar projection plane can be represented using a rotational matrix according to Equation 2.6. Figure 2.5 shows the rotation representation for non-coplanar planes in terms of the Euler angles as given in [18]. The line N is the line of intersection between the two planes, the angle \( \phi \) is the angle between the x-axis of the reference system (the camera plane) and the line N, the angle \( \theta \) is the angle between the z-axes of the two planes and the angle \( \psi \) is the angle between the z-axis of the rotated projector plane.

\[
R = \begin{bmatrix}
  \cos \theta \cos \psi & -\cos \psi \sin \phi + \sin \psi \cos \phi \cos \theta & \sin \phi \sin \psi + \cos \phi \sin \psi \\
  \cos \theta \sin \psi & \cos \phi \cos \psi + \sin \psi \sin \phi \sin \theta & -\sin \phi \cos \psi + \cos \phi \sin \psi \\
  -\sin \theta & \sin \phi \cos \theta & \cos \phi \cos \theta
\end{bmatrix}
\]

(2.6)

With a rotation of \( R \) the projection point changes to \( \hat{P}_2 \) according the rotation matrix as:

\[
\hat{P}_2 = TP
\]

(2.7)

Applying the ray-plane intersection on the rotated plane equation gives:

\[
\hat{P}_2 = F_2 + \beta (O - F_2)
\]

(2.8)
Here, $\beta$ is the signed distance from the focal point of the projector to the object point as stated by Hall.

\[
\begin{bmatrix}
R_{11} (x_2 + x_{2i}) + R_{12} (y_2 + y_{2i}) + R_{13} (z_2) \\
R_{21} (x_2 + x_{2i}) + R_{22} (y_2 + y_{2i}) + R_{23} (z_2) \\
R_{31} (x_2 + x_{2i}) + R_{32} (y_2 + y_{2i}) + R_{33} (z_2)
\end{bmatrix}
= \begin{bmatrix}
x_2 \\
y_2 \\
z_2 + f_2
\end{bmatrix}
+ \beta \begin{bmatrix}
x_o - x_2 \\
y_o - y_2 \\
z_o - z_2 - f_2
\end{bmatrix} \tag{2.9}
\]

Solving Equations 2.4 and 2.9 gives an equation for the computation of $z_o$ as:

\[
z_o = f_1 \left\{ R_{11} \left( (z_2 + f_2) (x_2 + x_{2i}) \right) + R_{31} \left( (x_i - x_2) (x_2 + x_{2i}) \right) + z_2 (x_2 - x_i) \\
R_{32} \left( (x_i - x_2) (y_2 + y_{2i}) \right) + R_{33} \left( z_2 (x_1 - x_2) \right) \\
R_{12} \left[ y_2 (z_2 + f_2) + y_{2i} f_2 \right] + R_{13} \left[ z_2 (z_2 + f_2) \right] - f_2 x_1 \right\}
\]

\[
\left\{ (f_1 R_{11}) (x_2 + x_{2i}) + (f_1 R_{12}) (y_2 + y_{2i}) + f_1 (R_{13} z_2 - x_2) - x_i f_2 \\
(x_i R_{32}) (y_2 + y_{2i}) + (x_i R_{31}) (x_2 + x_{2i}) + (x_i z_2) (r_{33} - 1) \right\} \tag{2.10}
\]

The mathematical basis to computing the depth of an object point mentioned above, requires the estimation of the intrinsics of the camera and projector systems (such as the focal length) along with the rotational and translational extrinsics of the projector plane with respect to the camera plane (the rotational matrix $R$ and the center of the projection plane $O_2$). Such a calibration can be performed using tools provided, for example, by Fofi et al. in [19]. This calibration procedure is explained in more detail in Section 3.2.2.
2.2 Structured light projection pattern

In this section the various structured light patterns available in the literature are surveyed and finally a pattern suitable for the current application is designed.

2.2.1 Classification

The design of structured light patterns has evolved over the years after a large amount of research done in this field of computer vision. The projected pattern is designed with the intention of uniquely identifying each pixel in the projection pattern, done by either coding the pixel with its row or column position information or both. Structured light patterns are classified according to various pattern design parameters. Classifications are based upon the number of dimensions the pattern has, the multiplexing strategy used, the coding strategy used and the pixel depth of the patterns. Such classifications are explained in more detail in the sections below:

2.2.1.1 Number of dimensions

Structured light patterns can be classified according to the number of dimensions the pattern is made up of. These can be zero-dimensional points, a one-dimensional line or multi-dimensional patterns.

- **Zero-dimensional points**: A pattern consisting of a single point is the simplest form of a structured light pattern. The location of the projected point from the projection plane is known and the single point captured on the image plane can easily be identified. This ensures a simple and fast solution to the correspondence problem for a single point. However, such systems suffer from some major drawbacks. Firstly, for such point projections the depth of only a single point in space can be computed at a time instant, thus the single point pattern has to be extended in space and time by moving the point projection across the environment to be mapped. Such an extension increases the time required to scan any given area, reducing the efficiency of the entire system. Secondly, systems employing such projection strategies cannot be used in applications requiring relative motion between the system and the region to be mapped, since at one time instant only a single point can be mapped. Finally, a separate system needs to be implemented so as to move the single projection point across the scene to be mapped. This if done at a very high rate can completely remove the earlier drawbacks. Figure 2.6(a) shows an example of such a pattern with the red dot indicating the point projected at a given time instant and the arrows showing the area traversing direction.

- **One-dimensional lines**: Improving on the point based structured light systems, a single line can be projected [20] [21]. A major constraint here is that the projected line must be perpendicular to the displacement between the projector-camera pair. This allows the solution to the correspondence problem for an entire row or column depending on the
relative orientations of the projected line and the projector-camera pair. The solution to the correspondence problem is just as simple as for the point projection, allows decoding of multiple points at a time but also has the same drawbacks. Additionally, the positions of points within the line cannot be identified. Figure 2.6(b) shows an object on which a (red) line is projected. This line has to be traversed in time across space to map the entire object.

- **Multi-dimensional patterns**: Such patterns are coded in more than one-dimension and can be formed by extending the one-dimensional pattern spatially [22] [23] or temporally [24] or both [25]. The major drawbacks of the zero and one-dimensional patterns are removed using such patterns. However, the solution to the correspondence is more compute intensive since an additional step of decoding each pixel coordinate in the projected pattern has to be carried out which can lead to a loss in spatial resolution depending on the coding strategy used. Figure 2.6(c) shows one such two-dimensional pattern which projects a grid of points in space.

### 2.2.1.2 Multiplexing strategy

According to this classification, patterns are divided into temporally multiplexed or spatially multiplexed patterns.

- **Time-multiplexed codes**: In this strategy, the projected pattern consists of a set of patterns which illuminate the measurement surface temporally in succession as given in [24][26][27]. The codeword for every pixel is then formed by the combination of the illumination values for each pixel time-multiplexed across the various patterns. The various patterns developed are based on e.g.: binary, n-ary and gray codifications. Figure 2.7(a) shows time-multiplexed binary and grey coded patterns. Such patterns offer very high accuracy, resolution and can be employed in systems not involving relative motion between the system and the environment of which depth has to be computed. As proposed by Altschuler et al. [28] temporal strategies can be used for dynamic measurements, but this method requires the use of more complex optical systems.

- **Space-multiplexed codes**: In this strategy the codeword assigned to a point in the pattern is based on the neighborhood of that point. Here, all the coding information is put into a single projection pattern making this strategy highly suitable for the applications of autonomous navigation. The features used here are the intensities [23][29][30], the colors [31], the unique structure of the neighborhood [32] or the shape for the codification of a position. One way of encoding such patterns is by using well-defined sequences called De Bruijn sequences [33]. Such a sequence is of length $2^n$ with all possible strings of $n$ successive bits distinct as defined by Arazi in [34]. The sequences can be extended by using more alphabets (or colors) giving a length of $m^n$, where $m$ is the number of
alphabets used. Figure 2.7(b) shows examples of spatially multiplexed patterns. Such patterns necessitate the identification of the sub-blocks of pixels which have the special color or spatial property associated with them.

### 2.2.1.3 Coding strategy

The next classification parameter is the coding strategy. Structured light patterns are classified as absolute coded patterns, periodic coded patterns, patterns coded according to shape or according to the neighborhood.

![Coding strategies](image)

**Figure 2.8: Coding strategies**

- **Periodical coding**: Periodical coded patterns encode the positions of the pattern points periodically. The main advantage of using periodical coding is the reuse of coding elements such as color and spatial structures. This allows shorter codewords and can possibly improve the decoding robustness due to the use of local decoding windows. Figure 2.8(a) shows a pattern using periodical coding methodology [35]. A major drawback of this encoding strategy is the ambiguity resulting due to the periodic nature of the pattern.

- **Absolute coding**: In absolute coded patterns, each projected pattern point is encoded so as to have unique coordinates in the entire pattern. Such coding can be achieved by using a wide range of colors or by using pseudorandom sequences to uniquely code each point. Figure 2.8(b) shows an example of an absolute coded pattern [36]. As is obvious due to the unique codeword assignment the problem of ambiguous decoding is absent.

- **Shape coding**: Patterns encoded according to shape, allow for the decoding of every pattern point based on the shape of only that point. This ensures a local decodification improving robustness for abrupt changes in depth but has a few major drawbacks. Since every projected point must be uniquely identifiable, to enable high resolution a large number of shapes must be projected which can be accurately decoded. A large number of unique shapes requires large shapes to allow for large hamming distances for accurate decoding, this reduces the spatial resolution of such a system. No pattern employing only a shape coding strategy was found in the literature but Figure 2.8(c) shows a possible example of a shape coded pattern.

- **Neighborhood coding**: In this coding strategy the neighborhood of a pattern point defines the location of that pattern point. This requires the correct identification of each of the neighborhood points increasing the decoding window size. With such an increase, all the neighborhood points might not be detected during abrupt depth changes leading to a reduction in robustness of the decoding algorithm. However, with a reduction in the requirement of the uniqueness of each pattern point, the number of unique points reduces, thereby reducing the overall size of each point and thus points can be accurately identified even with relatively smaller neighborhood windows. Furthermore, the use of colors instead of spatially structured points can reduce the window sizes even more and
leads to patterns with higher spatial resolutions. Such neighborhoods can be formed in
time or in space. Figure 2.8(d) shows one such pattern as given in [37].

2.2.1.4 Pixel depth

The next classification parameter is the pixel depth of the structured light pattern. Structured
light patterns are classified as color, grey level or binary coded according to this classification.

![Image of pattern types]

(a) Color coded  (b) Grey level coded  (c) Binary coded

Figure 2.9: Coding strategy based on pixel depth

- **Color coded**: In color coded patterns a number of colors can be used in the codification of
the pattern. The use of colors allows for simple decoding by easily identifying the color [38]. However, for such patterns, an example shown in Figure 2.9(a), the colors perceived by a camera heavily depend on the intrinsic color of the surface. Hence, these patterns are mostly used on color neutral surfaces. In color non-neutral surfaces a solution to this drawback is the use of a time-multiplexed, completely white projection pattern with the color coded pattern, which captures the intrinsic color of the surface to be depth mapped. The knowledge of the inherent colors can then be used in correctly identifying the colors perceived by the camera. Due to the use of the multiple projection patterns imperative in systems employing color coded patterns, its use is not suitable in applications involving relative motion between the object to be mapped and the mapping systems.

- **Grey level coded**: In grey level coded patterns, the codification is based on the lumi-
nance of pixels. The grey coded patterns still suffer from the drawback of the luminance perception by the camera but with the exception that the perception is now only in one channel instead of multiple channels as in the case of colored pattern designs. Another drawback of such patterns is the reduced robustness of the depth estimation in environments where the texture resembles the code. Figure 2.9(b) shows a pattern with rows periodical coded by grey level intensities [39].

- **Binary coded**: In binary coded structured light patterns only two color intensities are
used to encode the pattern - white and black. Due to the use of only two colors the perception problem from the camera can be minimized using an appropriate thresholding mechanism. Figure 2.9(c) shows the binary pattern proposed by Griffin et. al in [37].

2.2.2 Finalized pattern

According to the simplified principle stated in Section 2.1.4.1 the structured light pattern selected must allow the identification of each pattern point in terms of its row and column location in the pattern. The application of automated obstacle avoidance requires relative motion between the automated avoidance system and the environment to be depth mapped, this necessitates the use of motion robust patterns. The application can involve depth mapping of
surfaces with a lot of discontinuities, which makes it imperative to use patterns with small decoding windows. Finally, the use-case scenarios of color non-neutral environments provides another constraint in the selection of the structured light pattern.

With the above constraints in mind the pattern chosen, suggested by Griffin et al. in [37], is two-dimensional and spatially multiplexed to be motion robust. It also has to be absolute-neighborhood coded with a small neighborhood to allow for unique identification of all pattern points and minimizing surface discontinuity errors. Finally, the pattern must be binary in pixel depth to work in color non-neutral environments. Additionally, as stated by Griffin, the pattern must consist of unique spatially designed grid primitives instead of color coded primitives to cope for the color non-neutral environments. The next sections detail the encoding and decoding methodology of the selected pattern.

2.2.2.1 Pattern encoding

The encoding methodology of the pattern requires the formation of a \((M, N)\) 2D De Bruijn Sequence (dBS), where \(M\) is the number of columns and \(N\) the rows. This requires the formation of a \((P, 3)\) and \((P, 2)\) dBS and then using those two sequences to form the 2D dBS. A \((i, j)\) dBS is a sequence of length \(i^j\), with \(\text{characters} \in [1, P]\), where every possible subsequence of length \(j\) appears as a continuous sequence of characters exactly once. The pattern is generated according to the following steps:

1. **Selection of \(P\)**: The span of the dBS \(P\) is selected depending on the spatial resolution requirements of the application. For the current application the entire structured light pattern should be at least equal to the working area of the robotic platform. Selecting the pattern dimension exactly equal to the working area at a given projection distance gives the most efficient use of the available camera resolution and allows for the detection of obstacles in the platform’s path. The computation of the working area is given in detail in Chapter 3. A span of \(P = 4\) gives a spatial resolution adequate for the current application.

2. **Formation of \((i, j)\) De Bruijn Sequence**: The generation of a \((i, j)\) dBS is as given by Hsieh in [40]. Following the procedure results in an efficient decoding methodology and yields the \((P, 3)\) and \((P, 2)\) dBS respectively as:

\[
G(P,3) = 444344244143343243142342242141341241133323313223213123112212111 \\
G(P,2) = 4434241332312211
\]

3. **Formation of \((M, N)\) 2D De Bruijn Sequence**: Using the dBS sequence pairs given above a \((M, N)\) dBS can be formed as stated by Griffin in [37]. The algorithm given by Griffin contained a minor flaw due to which it did not generate the pattern as claimed. The generation of the first row of the 2D dBS is as mentioned by Griffin with the first row corresponding to the dBS \(G(P, 3)\). The second row is created by adding the first element of \(G(P, 3)\) to each element of \(\{G(P, 3) \text{ modulo } P\}\). Griffin fails to mention here that if the previous operation results in a value > \(P\) then another modulo operation with \(P\) has to be carried out. The modulo operation mentioned throughout this part of the algorithm differs from the normal mathematical definition since it is only on the set \([1, P]\) and does not contain the element 0. Figure 2.10 shows the final structured light pattern.
The dBS constructed from the above steps contains a \((M,N)\) 2D grid of values in \([1,P]\). This is converted into the projection pattern using the assignment of structured elements called grid primitives to each value as shown in the figure. Each primitive is of size 9 x 5 pixels with a vertical spacing of 6 pixels and a horizontal spacing of 2 pixels between adjacent primitives as shown in Figure 2.11.
2.2.2.2 Pattern decoding

The decoding process of a structured light pattern effectively computes the coordinates of every pattern point in the projected plane. An algorithm capable of efficiently decoding the Griffin pattern is given by Hsieh in [41] but requires the encoding process to be followed according to Hsieh as stated in Section 2.2.2.1. This algorithm requires correct detection of a grid primitive and its four adjacent neighbors, a 5-tuple word, as shown in Figure 2.12. The row position of a particular primitive is first determined using the adjacent neighbors above and below the primitive. The column position is then determined using the \( G(P,3) \) \( dBS \), the adjacent neighbors on the left and right of the primitive and the row position determined previously. The algorithm is too detailed to be explained in the current context and the interested reader can refer to [41] for details.

![Figure 2.12: Decoding window](image)

A minor flaw in the algorithm described by Hsieh is in the computation of step 3 of the given algorithm. Here, instead of computing \( w_{x,1} \) the value \( w_{x+1,1} \) needs to be computed for the correct decoding of the column position of a given grid value. As stated by Hsieh, this algorithm is efficient only in terms of space complexity as it requires the storage of only \( P^2 \) elements of the \( G(P,2) \) \( dBS \). The algorithm is computationally more expensive in time since it contains a lot of mathematical operations necessary for decoding the location of a primitive. Another decoding solution exists in which a look-up table with the row and column positions of each of the 5-tuple word combinations existing in the pattern is stored and the decoding is performed by brute-force searching. This leads to a space requirement of \( 5 \times P^2 \times P^3 \) memory units. As can be seen for a smaller number of grid primitives, as in the case of the current pattern, the brute force search is still feasible.
This section gives a detailed explanation of the system design. Figure 3.1 gives an overview of the various elements involved in the execution of the obstacle avoidance for mobile robotic platforms.

![Figure 3.1: Project overview](image)

The offline system setup refers to all the aspects of constructing the obstacle avoidance system. This includes the selection of the hardware and software components and then the assembly of hardware components along with the supporting software modules. This is then followed by the calibration procedure which takes into account the relative placement of the camera and the projector. Both the system setup and calibration are one-time offline procedures and are followed by the depth estimation algorithm which is necessary to map the observable environment of the obstacle avoidance system. The following sections explain these concepts and their sub-blocks in more detail.

### 3.1 System setup

As a first step towards an implementation, a high level component selection and assembly is carried out. For the current application of obstacle avoidance, the components considered are a robotic platform, a camera, a projector, and an embedded computing system capable of performing the obstacle detection and robot control for obstacle avoidance. The motivation for the selection and the assembly of each of the components is given in further detail in the following sections.
3.1.1 Embedded computing unit

The embedded computing unit is the heart of the navigation system controlling the entire system. It needs to be able to communicate with and control the camera, the projector and the robotic platform to efficiently achieve the goal of automated navigation. One system that satisfies all these constraints is the BeagleBoard by Texas Instruments [42]. It contains an ARM core as the CPU, with additional peripheral connections for USB, RS-232 communication and DVI-D.

3.1.2 Operating System

The selection of an appropriate operating system (OS) is one of the main aspects which affects the development of the obstacle avoidance system employing the BeagleBoard. The main factors in the selection of the OS were economic aspects such as cost and licensing charges and the availability of drives necessary for the interface with the various elements of the system. The open source Linux based Angstrom distribution [43] was found to be the ideal OS for the BeagleBoard for the current application, since it was the only freely available OS having drivers for the interface with the selected camera system. This requirement is mentioned in depth in Section 3.1.5. For an easier development with the BeagleBoard system supplementary software packages need to be used. Packages for the I²C communication over DVI-D, compilers and libraries for the development of software etc. not supplied with the operating system had to be additionally installed.

3.1.3 Robotic platform

The robotic platform refers to a robot capable of following appropriate commands to achieve the desired navigation. The main criterion in the selection of a robotic platform is its ability to allow controllable navigation. One such platform was found to be the Create by iRobot [44] shown in the Figure 3.2.

![Create robotic platform](image)

Figure 3.2: Create robotic platform

This platform is explicitly designed for robotics development by iRobot which is a company specializing in the design and manufacture of vacuum cleaning robots. Hence, Create is also designed with physical dimensions similar to iRobot’s cleaners. For the robot control, the Create is provided with a special interface called Open Interface (OI). This consists of an electronic and software interface. The electronic interface is through either a 7 pin mini-DIN connector or a DB-25 connector. The software interface is a set of commands which need serial communication through the electronic interface for control information transfer to Create. For the electronic interface the Beagleboard is connected to Create using a special DB-9 to mini DIN serial cable supplied by iRobot. A direct connection from the Beagleboard’s serial port is not possible since the Beagleboard gives RS232 signals at 3.3V but Create requires TTL signals at 5V. A special cable is provided by iRobot containing an internal voltage converter to achieve this function.
The commands for the control of Create to be used in the current project are mentioned below:

- Before sending any command to Create a Start command has to be sent which starts the OI.

  Command: \([128]\).

- The drive command controls Create’s drive wheels. This requires two other parameters to be given - the velocity and the radius. Velocity is a value in the range -500 to 500 mm/s and radius between -2000 to 2000mm. The radius is measured from the center of the turning circle to the center of Create. A drive command with a positive velocity and positive radius makes Create drive forward while turning left, while a negative radius makes it turn right. Both these parameters are represented in 16-bit 2’s compliment.

  Command: \([137]/[\text{Velocity high byte}]/[\text{Velocity low byte}]/[\text{Radius high byte}]/[\text{Radius low byte}]\)

- Special cases of drive commands are:
  - Straight = \([8000]\) or \([7FF]\) in hex
  - Turn in place clockwise = \([FFFF]\) in hex
  - Turn in place counter-clockwise = \([0001]\) in hex

- Create contains an internal mechanism which determines how much distance it has traveled which can be used in defining the range of movement of the platform. This distance is defined in \(mm\) and also needs to be represented in 16-bit 2’s compliment.

  Command: \([156]/[\text{Distance high byte}]/[\text{Distance low byte}]\)

3.1.4 Projection system

The projection system must be capable of projecting the selected projection pattern which is the first step in the solution of the correspondence problem. The Pico projector system developed by Texas Instruments \([45]\) allows for an easy construction of a desired projection pattern and has a very small form factor motivating its selection as the projection system. This projector supports a suitable resolution of 320 x 480 and connects to the BeagleBoard using the HDMI connection with the control being handled by the Angstrom OS through \(I^2C\) communication. The placement of the projector, with respect to its placement on the platform and its orientation, forms a very important part of the system setup as it determines the depth computation area of the entire system.

3.1.4.1 Placement

The projector has to be placed on the top of the robotic platform due to the design of the platform. To ensure minimum angular distortion on the projected plane the projector should be as close to the central line of the robotic platform as possible. A wide gap between the projector and this central line results in a deformed projection as shown in Figure 3.3(c). Due to the structure of the projector heat sink the minimum off-center displacement for the projector’s center is 30mm.
3.1.4.2 Orientation

The DLP Pico projector used in the current system has an asymmetric display with an angular displacement in the vertical direction. This is shown in Figure 3.3(b). For the projection pattern to cover the entire height of the platform the projection will have to be focussed at a very large distance from the platform. To avoid this, the projector is positioned upside down. However, this causes the projection pattern to be displayed inverted in space. The Pico projector has an I²C interface which can be used to configure its display. Using special commands the display is flipped about both the axes (I²C : Sub-address 0x08, Data 0x00000000 ; I²C : Sub-address 0x09, Data 0x00000000) to correctly display the pattern [46].

Furthermore, the projector in default mode works in the ‘scaling’ mode where any input image is scaled to the projector resolution. This introduces scaling artifacts in the projected pattern leading to errors in the decoding process. A solution to reduce these errors is the use of the ‘cropping’ display mode (I²C : Sub-address 0x05, Data 0x00000007) where for every image only the top-left part is displayed as shown in the Figure 3.3(a). The I²C commands used for this process are executed in a script file during system boot-up so as to ensure the correct orientation of the projected pattern after system start-up.

3.1.4.3 Projection distance

The projection distance corresponds to the distance from the robotic platform at which the projection pattern is at least the size of the working area of the system. This working area of the system can be defined taking into account the positioning of all the components of the system and is given in Section 3.1.6. The width of the system is a more important factor in defining the projection distance since the height of the system can be reduced by adjusting the placements of the components, however, the minimum width of the system is constrained by the diameter of the circular robotic platform which is fixed. Thus, the main factor defining the projection distance is the horizontal angle of projection of the projector. Straightforward geometry leads to the computation of this distance at 675mm.

3.1.5 Camera system

The selection criteria for the camera system is based on the quality aspects of the camera, its flexibility with lens choices and the availability of control software. With the selection of a binary structured light pattern, a monochrome camera proves sufficient for the current system. Greyscale images further reduce the overall memory footprint and the computation intensity in processing the input thereby allowing for an efficient use of the limited available resources of the BeagleBoard. Finally, a camera with a simple interface allowing for a quick and easy
development is preferred. The $\mu$EyeU1122xLE - M USB camera by iDS, with the additional advantage of allowing varifocal S-mount lenses, was chosen since it satisfied all the previous constraints. It also allowed for more flexibility with control over a number of internal features such as resolution, frame rate, region of interest etc.

![Single Frame Acquisition Diagram](image)

**Figure 3.4:** Single frame acquisition steps in $\mu$Eye cameras

The camera was a major selection criterion in the finalization of the OS for the BeagleBoard since its drivers were not developed for use on every OS and processor architecture combination. The drivers provided by iDS allowed for extremely slow single frame acquisitions at approx. 1 fps, which would not prove adequate for a robot navigation system and were only available as binaries preventing modifications to allow for more control. Using the APIs provided, a new control software was developed, according to the steps in Figure 3.4. As a first step the camera needs to be initialized with parameters for exposure time and the pixel depth. In the next step, memory is allocated and set for the frame to be captured. The frame capture is carried out next using an appropriate software trigger and the captured frame is saved to memory. This frame is then used in the computation of the 3D coordinates as explained in Section 3.3. It was found that the bottleneck in achieving the desired frame rate was the initialization step of the frame acquisition. By using a single initialization followed by sequential frame acquisitions the frame rate could be improved to 10 fps. This, however, made it mandatory for the depth estimation to be performed sequentially after the frame acquisition. For higher frame rates the video mode feature of the cameras could be used.

### 3.1.5.1 Placement

Similar to the placement of the projection system the camera also has to be placed on top of the robotic platform. To allow for focussing at the projection distance and to cover the working area by the image plane, a S-mount varifocal lens is used. The camera is placed as close to the central line of the robotic platform as possible so as to reduce distortions caused due to the angular imaging. However, this allows for smaller displacements in the imaged projected pattern for varying depths. Increasing the distance between the camera-projector pair increases the disparity leading to higher accuracy in the depth measurement but increases the occlusions caused due to an increased angular separation between the optical axes. An increase in occlusions reduces the spatial area which can be depth mapped, lowering the resolution of the entire system.
3.1.6 Finalized hardware setup

Figure 3.5 shows the setup as used for the current project of obstacle avoidance.

Figure 3.5: Final System Setup

As can be seen the working volume of the entire system is $B \times H \times W$. The working volume refers to the dimensions the system needs to take into account while driving in the environment. At any time instant the system should ensure an area of $W \times H$ in front of it is clear of any obstacles to drive in the forward direction and the dimension $B$ determines the distance to be covered before the entire robotic system is cleared through an obstacle free opening.
3.2 Calibration

System calibration is a necessary step in the computation of 3D coordinates. In this thesis, two calibration procedures were implemented. The first procedure is based on Zhang’s calibration method [47] and aims to be used to determine the intrinsic and extrinsic parameters necessary for the depth estimation according to the mathematical analysis, while the second one is a simplified procedure developed to efficiently compute the depth. Both the calibration procedures are explained in further detail in the following sections.

3.2.1 System calibration based on Zhang’s method

In this calibration procedure intrinsic and extrinsic parameters for the camera, the projector and the visual system as a whole are computed. This is in accordance to the mathematical analysis presented in Section 2.1.4.2, which states that a calibration procedure must precede depth estimation.

One procedure to achieve this calibration is by first calibrating the camera and then using the calibrated camera in the calibration of the projector. The camera calibration is performed using Zhang’s calibration method [47] and implemented in Matlab by Bouguet in [48]. This method assumes a pinhole model of the camera and using a checkerboard pattern imaged at various orientations with respect to the camera gives its intrinsic parameters such as the focal length and the distortion coefficients and its extrinsics giving the rotational and translational matrices. Using this calibrated camera the projector needs to be calibrated next. Figure 3.6 shows the setup needed to perform such a system calibration.

![Figure 3.6: Calibration technique based on Zhang's method](image)

The projector calibration can be achieved using the Matlab toolbox developed by Falcao et al. in [19]. This toolbox assumes the projector to be a point source thereby implying an inverse pinhole camera model and inherently uses the Bouguet toolbox. The calibration setup shows a calibration plane containing a printed checkerboard calibration pattern which must be used to calibrate the camera and a similar pattern projected next to it. The camera needs to capture both the patterns from various orientations in order to calibrate the system as a whole. The projector calibration gives the focal length of the projector and the rotational, and translational matrices with respect to the camera. Thus, the calibration procedure all gives the parameters such as focal lengths of the camera and the projector, and the principal point of the camera along with the orientation of the projector with respect to the camera plane necessary to compute the depth as described in Section 2.1.4.2.
Calibration results

Figure 3.7 shows a few sample images used for the system calibration. The pattern on the left is the printed pattern and the one on the right is projected onto the calibration plane.

A total of 15 images was used for the calibration. A large dataset of calibration images implies a higher number of points for calibration which gives more accurate results.

1. Intrinsic parameter

The intrinsic parameters necessary for the depth map computation, according to the analysis in section 2.1.4.2, is the focal length and the principal point of the camera, and the focal length of the projector.

<table>
<thead>
<tr>
<th>Component</th>
<th>Focal Length ((mm))</th>
<th>Principal Point ((pixels))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera</td>
<td>(8.952 ; 8.951)</td>
<td>(330.4 , 220)</td>
</tr>
<tr>
<td>Projector</td>
<td>(5.722 ; 5.937)</td>
<td>N.A.</td>
</tr>
</tbody>
</table>

Table 3.1: Intrinsic parameter

Table 3.1 gives the focal lengths of the camera and the projector obtained, and the principal point of the camera using the calibration procedure. The calibration tool gives focal length results in pixel units which are converted into distance units by multiplication with the pixel pitch of the camera. The presence of two unequal focal lengths indicates an aspect ratio different from 1. This occurs if a pixel in the CMOS sensor array is not exactly square in shape.
2. Extrinsic parameters

<table>
<thead>
<tr>
<th>Rotational</th>
<th>Transition (mm)</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
0.993 & 0.006 & 0.109 \\
-0.027 & 0.982 & 0.190 \\
-0.106 & -0.192 & 0.975 \\
\end{bmatrix}
| \[
\begin{bmatrix}
55.24 \\
-3.98 \\
11.56 \\
\end{bmatrix}
|

Table 3.2: Projector extrinsics

Figure 3.8: Projector - camera orientation

Figure 3.8 shows the orientation of the projector and camera with respect to world coordinates. Here, the image center is the origin of the world coordinate system as explained in Section 2.1.4.2. Table 3.2 gives the rotational and translational matrices which are essential for the computation of depth according to Equation 2.10.
3.2.2 Simplified calibration

Since the above mentioned calibration methods provide extremely accurate models for camera-projector pair systems not necessary for the current application of obstacle detection, a simpler calibration method is implemented which exploits the basic principle of depth estimation as given in Section 2.1.4.1. This calibration allows for a simple and, hence, a fast depth computation once the correspondence problem has been solved.

Figure 3.9: Simplified calibration technique

Figure 3.9 shows the calibration procedure employed in the current project. The obstacle avoidance system with its fixed setup projects the selected structured light pattern on the calibration plane. The plane is moved normally to the central line of the system with known distances, such that all the points on the plane are equidistant from the system to be calibrated. For every distance the corresponding image coordinates for each primitive of the structured light pattern are recorded. Due to the variation in distance, the primitives are imaged at different coordinates on the camera plane but this variation is linear with the varying depth and the depth variation is projected across a unique line on the camera plane for each grid primitive. The calibration procedure aims at determining this line for each grid primitive. A higher horizontal than vertical displacement of the projector from the camera causes the variation of the column coordinate for each primitive on the camera plane to be much higher than the variation of the row coordinate. Once the calibration has been performed precautions must be taken to avoid disturbing the hardware setup to prevent the need of re-calibrating the entire system.
Calibration results

For the simplified calibration a set of 3 calibration images were used. The image set used in this calibration is shown in Figure 3.10

![Calibration images](image)

(a) Depth = 489mm  (b) Depth = 572mm  (c) Depth = 692mm

**Figure 3.10:** Calibration images placed at different depths

As can be seen from the calibration images as the depth increases the pattern moves to the left in the image plane. This is due to the angular setup of the projector. For each depth of the calibration plane, the column coordinate of each grid primitive (identified by its corresponding row and column position in the projection pattern) is noted.

![Calibration results](image)

(a) Depth line for all primitives  (b) Least square fit line for a primitive

**Figure 3.11:** Calibration results

Figure 3.11(b) shows the calibration points imaged for a particular grid primitive at varying depth for the corresponding column coordinate on the image plane. Using linear regression a least square fit line $L$ is then computed represented as Equation 3.1.

$$L \equiv \text{Slope} \times \text{Column No.} + \text{Intercept} \quad (3.1)$$

If the column coordinate in the image plane for a particular grid primitive is known, the depth can be linearly computed using the knowledge of the line $L$. This calibration data is stored as a look-up table for each primitive with the line defining parameters of slope and $y$-intercept. Figure 3.11(a) shows the depth lines for all the grid primitives obtained with the current setup and Figure 3.11(b) shows the depth line for one primitive with the calibration points used. With an increase in the number of calibration planes, with a smaller depth step, the number of points obtained for the least square fit of the depth axis increase giving more accurate calibration results. This number of points is limited by the depth range of the calibration planes, the camera resolution and orientation since for every depth all the grid primitives must be imaged at the camera plane. For the current application of obstacle avoidance with relaxed accuracy requirements, three calibration points are shown to give accurate results as shown in Chapter 4.
3.3 Depth estimation algorithm

With the system setup and the calibration performed, the system can be deployed in desired environments to estimate depths. In this section a detailed description of the algorithm used to achieve this goal is given. The algorithm follows the sequence of operations as given in Figure 3.1. Each of the steps of the algorithm are explained in detail in the sections below.

3.3.1 Pattern projection

The first step in the depth determination algorithm is the projection of the structured light pattern. Since a single frame pattern is selected for the current application of obstacle avoidance, a simple image viewer (e.g. GQView) can be used to achieve projection using the selected Pico projector.

3.3.2 Image capture

The projected pattern deforms due to the depth variations in the observable environment to be mapped. This is captured using the µEye camera selected.

3.3.3 Solution to correspondence problem

The solution to the correspondence problem is the most critical step in the creation of the depth map. An incorrect correspondence determination for a grid primitive can lead to high inaccuracies in depth estimation for the particular grid primitive. The following flowchart details the decodification process of the projected pattern which yields the solution to the correspondence problem.

![Flowchart of correspondence problem solution](image)

**Figure 3.12:** Solving the correspondence problem

As an initial step the solution was implemented using Matlab to check the functionality of the algorithm and then ported to the BeagleBoard using OpenCV and C. The following steps detail the implementation details of the algorithm:

---

29
3.3.3.1 PreProcessing

The main goal of solving the correspondence problem can be achieved by identifying each grid primitive. This requires the segmentation of the foreground primitive object (white pixels) from the background (black pixels). Since the projected pattern is binary in depth, all processing to decode the primitives can be done in binary which further leads to a more efficient algorithm. To improve the binarization of the greyscale input, contrast enhancement using histogram equalization is applied. This is then converted into a binary image using local thresholding. Figure 3.13 shows the improvement achieved using contrast enhancement.

![Binarization](image)

Figure 3.13: Binarization

As can be seen from the above figure, contrast enhancement introduces a lot of speckle noise. Furthermore, two horizontal white bands are present at the upper and lower boundaries of the captured pattern. All these do not form a grid primitive and only add to the detection of false positives and must be filtered out. A very easy filtration technique to remove the speckle noise is using a median filter. However, a median filter affects the grid primitives distorting its edges. A more robust filter can be produced using morphological techniques. As can be seen the objects to be filtered have specific differentiating features from the primitives. The speckle noise is disconnected from its neighbors and hence constitutes very small objects. The horizontal boundaries have a very high ratio of the length to its width which can be used as a filtration criteria. Figure 3.14 shows the output obtained using the above filtration.

![Filtered output](image)

Figure 3.14: Filtered output
3.3.3.2 Grid primitive segmentation

Grid primitive segmentation refers to the isolation of the grid primitive objects from the background. Similar to the preprocessing filter, a morphological segmentation technique can be used. Here, for every foreground (white) pixel all its 8-connected neighbors are found. All such connected pixels represent one object segmented as a grid primitive. Figure 3.15 shows the output obtained, with each enclosed red boundary defining the segmented primitive.

![Segmented grid primitive edges shown in red](image)

Figure 3.15: Segmented grid primitive edges shown in red

3.3.3.3 Grid primitive center location

The next step in the algorithm is the identification of the location of each grid primitive in the image plane, thereby solving half of the correspondence problem. This location is recognized as the pixel position of the center of the primitive.

![Center location using bounding box](image)

Figure 3.16: Center location using bounding box

The center of each grid primitive can be easily computed as the center of the rectangular bounding box of each segmented grid primitive as shown in Figure 3.16. The column coordinate of this center is directly used in the computation of the depth at which the primitive is projected. Figure 3.17 highlights the center obtained for each of the segmented primitives as red points.

3.3.3.4 Grid primitive identification

Once a grid primitive has been segmented, the algorithm needs to determine its value as encoded in the projection pattern. Since the projected pattern encodes each primitive according to its
shape, a template matching algorithm is used to decode the primitive. The reference primitives are the perfectly shaped primitives as shown in the Figure 2.10.

The template matching algorithm performs cross correlation between the reference primitives and every segmented primitive according to the Equation 3.2 as given in [49]:

\[
R_{corr}(x,y) = \sum_{x',y'} [T(x',y') \cdot I(x+x',y+y')]^2
\]  

(3.2)

Here, T represents the reference template the primitive is matched against, I represents the segmented grid primitive and R represents the resulting 2D cross correlation of size.

\[(\text{Image Height} - \text{Template Height} + 1) \times (\text{Image Width} - \text{Template Width} + 1)\]

The segmented grid primitive is resized to the reference primitive size, using bicubic interpolation, so as to have a resulting correlated image of size 1x1 pixel. The most correlated of the reference primitives gives the decoded value of the grid primitive. Figure 3.18 shows the decoded primitives color coded according to the primitive value encoded in the pattern.

As can be seen from the figure, all the grid primitives segmented correctly are correctly decoded. This enables a robust solution to the correspondence problem.
### 4-adjacent neighbors identification

As stated in [37] the row and column position of every grid primitive in the projected pattern can be simply decoded by finding its 4-adjacent neighbors. Since such a 5 value word occurs only once in the projected pattern a look up table can give the corresponding row and column numbers in the projected pattern. This look-up table can be generated offline and needs to be statically computed only once.

The 4-adjacent neighbors to each detected and decoded primitive can be found in the 4 directions using appropriate search windows. To optimize the search window for the neighbor detection small windows around the center of a primitive is searched. The regions are shown in Figure 3.19:

![Figure 3.19: Detecting 4-adjacent neighbors](image)

The $MaxPrimitiveWidth$ and $MaxPrimitiveHeight$ parameters represent the widest and tallest primitive detected respectively in the previous step. These parameters are necessary to avoid missing any neighbors. The $HSpace$ and $VSpace$ parameters represent the horizontal and vertical spacing between two grid primitive edges respectively.

Due to the size of the window, and the incorrectly segmented grid primitives it is possible that more than one grid center is found indicating the presence of a number of grid primitives closely located near the primitive (its center represented by $GridX, GridY$). Here, the selected neighbor is the one closest to the central grid primitive and is found by the following equation:

$$ (X, Y) = \min \left( \forall i, \sqrt{(x_i - GridX)^2 + (y_i - GridY)^2} \right) $$

where,

- $(X, Y)$ gives the position of the center of the closest neighbor in the appropriate neighbor window.
- $(x_i, y_i)$ gives the position of the $i^{th}$ detected center within the window.

Once all the 4-adjacent neighbors for each grid primitive have been decoded the location of the grid primitive in the structured light pattern can be computed.
3.3.3.6 Projector coordinate computation

The location of each grid primitive is found by matching the grid primitive word decoded upto the previous step. This word can be constructed depending on the value of the current primitive and that of its neighbors according to the sequence:

\[
\{ \text{CurrentValue}, \text{NorthValue}, \text{WestValue}, \text{SouthValue}, \text{EastValue} \}
\]  

(3.4)

Using an efficient search algorithm, such as binary search, this matching can be optimized for both time and space complexity. As a pre-requisite for the binary search algorithm, the look-up table is sorted with the word sequence converted into a single 5 digit decimal number and stored accordingly. The binary search algorithm can then find a matching sequence in \(O(\log n)\) time complexity and using a look-up table with the number of elements the same as the number of primitives in the pattern (1024 in the current case). The time complexity of the search algorithm can be further reduced to \(O(1)\) using the same look-up table size but this requires the formulation of an encoding scheme which matches each sequence to an appropriate index in the look-up table. With the current pattern, containing 1024 sequence combinations, this would not yield a significant performance gain in the overall solution to the correspondence problem and hence is not attempted. The search result obtained for every grid primitive word pair gives its corresponding row and column location in the structured light pattern. Figure 3.20 shows examples of this decoding strategy.

Figure 3.20: Computing the location of grid primitive in structured light pattern
3.3.3.7 Validity check of computed projector coordinate

The structured light pattern location of every grid primitive depends on the correct decoding of that grid primitive and its 4 adjacent neighbors as stated in the steps above. This constraint can lead to erroneous results if any one of the neighbors or the grid primitive itself is wrongly decoded. One way to do this, to a certain degree of robustness, is to check if the neighbors are decoded to have neighbor locations in the structured light pattern. The more the neighbor locations taken into consideration for this validity check the better. This, however, reduces the spatial resolution of the depth map. The probability of having three adjacent neighbors wrongly detect is found to be low, thereby removing a significant amount of errors and increasing the spatial resolution slightly. Equation 3.5 gives the steps used in more detail:

\[
\text{Location Valid} = \begin{cases} 
N \text{ Neighbor Row} &= \text{Primitive Row} - 1 \\
\text{and} \\
N \text{ Neighbor Column} &= \text{Primitive Column} \\
\text{OR} \\
W \text{ Neighbor Row} &= \text{Primitive Row} \\
\text{and} \\
W \text{ Neighbor Column} &= \text{Primitive Column} - 1 \\
\text{OR} \\
S \text{ Neighbor Row} &= \text{Primitive Row} + 1 \\
\text{and} \\
S \text{ Neighbor Column} &= \text{Primitive Column} \\
\text{OR} \\
E \text{ Neighbor Row} &= \text{Primitive Row} \\
\text{and} \\
E \text{ Neighbor Column} &= \text{Primitive Column} + 1 
\end{cases}
\]

(3.5)

3.3.4 Depth map computation

With the correspondence solved for the grid primitives, the depth map can be computed according to the principle mentioned in Section 2.1.4.1. The correct depth axis parameters of slope and y-intercept can be extracted from the stored look-up table by using the knowledge of primitive location in the structured light pattern. For every valid grid primitive (at the \(i\)th row and the \(j\)th column in the projected pattern) the depth can be computed according to the line Equation 3.6:

\[
\text{Depth (mm)} = \text{Slope}_{i,j} \times \text{Primitive column location in image plane} + \text{Intercept}_{i,j}
\]

(3.6)
3.3.5 Obstacle detection

With the depth map computed, an obstacle can be detected taking into account the physical dimensions of the system. As given in Equation 2.5, the 3D coordinates of any point can be computed by using the depth computed up to the previous steps in the algorithm. The coordinates computed are with the camera center \((O_1)\) as the origin. This origin needs to be shifted to the edge of the system \((O_2)\) to more easily identify obstacles as shown in Figure 3.21.

![Figure 3.21: Shift of Origin to compute 3D coordinates](image)

Due to time constraints a simple obstacle detection technique is used. Here, an obstacle is detected as any point present at a given depth if its coordinates are within the physical dimension of the system with respect to the origin \((O_2)\). No surface relationships between points in the depth map are taken into consideration to define an obstacle.

3.3.6 Robotic platform control

With the knowledge of an obstacle the robotic platform can be given control signals dictating its movement. The closest obstacle depth from the frontmost point of the platform defines the distance the platform can move in the forward direction. This distance is converted into the appropriate drive commands for the platform as given in Section 3.1.3. The platform should then travel the determined distance and stop.
In this chapter results obtained using the obstacle avoidance system are presented. Sample results for the depth computation are shown first for different surfaces. From these depth maps the presence or absence of an obstacle, defined by the system setup parameters, and the 3D coordinates of the obstacle are computed. Furthermore, a sample case of the system detecting an obstacle is shown. Finally, the performance of the system under varying light conditions is evaluated.

4.1 Distance tests

For obstacle detection and the subsequent obstacle avoidance, it is necessary to firstly determine the depth of objects in the observable environment. This section gives a few sample results showing the depth maps obtained for various observed environments. The results show the depths obtained for the image points as seen by the camera.
4.1.1 Planar depth

The simplest scenario is the presence of a planar surface perpendicular to the central line of the system.

![Figure 4.1: Depth map for planar wall at perpendicular depth of 700mm from system](image)

Figure 4.1 shows the depth map of a wall placed at a distance of approximately 700mm perpendicular from the system. The points in red show the center position of the grid primitives and the linearly interpolated depth between neighboring primitives is shown with blue lines. A projection of the depth map on the depth axis can be seen in green. The average error obtained for the valid primitives, as defined in Section 3.3.3.7, is found to be 1.56mm.

4.1.2 Multiple planes

A typical scenario in the indoor household environments for which the current system is designed is the presence of a number of parallel planes such as walls, furniture etc.

![Figure 4.2: Depth map for perpendicular plane surfaces from system](image)

Figure 4.2 shows the presence of three such planar objects with notations similar to the previous result. The left most plane is at a depth of 700mm, the central plane at 745mm and...
the rightmost at a depth of 650mm. The average error obtained for the current case is 4.11mm. As can be seen from the figure there are regions where the depth has not been estimated. These are due to the sudden change in depth which can cause an incorrect detection of the neighbors, leading to incorrect correspondences and hence incorrect depths. These have been filtered out using the validity check in the algorithm. Another reason for missing depth values is the problem of occlusion due to which entire primitives can be invisible for the camera.

4.1.3 Curved surface

The presence of curved surfaces is possible in the use case scenarios for the application of obstacle avoidance. Figure 4.3 shows the depth map estimated for one such scenario.

![Figure 4.3: Depth map for curved surface from system](image)

The environment consists of a planar surface at a depth of 650mm next to the curved surface. For the curved surface three sample points were measured for depth along the curve. The relative errors obtained for the markers from left to right is 3mm, 8mm and 8mm respectively. A more accurate measure of accuracy for the curved surface has to be performed by either accurately measuring the depth of each projected pattern point on the surface or by modeling the curved surface mathematically.
4.1.4 Shiny surface

Surfaces with a high amount of reflectivity can lead to surface regions missing depth values. Such surfaces can be present in abundance in typical environments due to the presence of metallic or well polished furniture or obstacles such as mirrors.

![Figure 4.4: Depth map for metallic dustbin](image)

Figure 4.4 shows the depth map obtained for one such surface at a depth of 586mm. As can be seen the depth map is not computed in the central region. Due to reflectance the acquired image saturates in this region, reducing the contrast between the grid primitives and the background causing errors. The location and size of this region depends on the reflectivity of the surface, the depth of the surface from the system and the angle of the surface with respect to the projector and camera.

4.2 Ambient light intensity variation

Ambient light present in the environment can cause serious errors in the detection and decoding of the grid primitives. Figure 4.5 shows the trend observed while varying ambient light intensity on robustness.

![Figure 4.5: Effect of light intensity variation on robustness](image)
Here, robustness refers to the percentage of correctly decoded grid primitives, in terms of its position in the structured light pattern. The light intensity is measured at ground level since this is the use case scenario for the current project. As can be seen with an increase in light intensity, robustness decreases linearly. It is expected that the robustness value drops abruptly for much higher intensities due to reduced contrast between the grid primitives and the background. However, such conditions could not be simulated at the ground level. All measurements were taken against a white background giving a high contrast and thereby the best case results possible.

4.3 Effect of depth on measurement accuracy

In this section, the errors obtained for varying depths is determined. This is necessary to determine the optimum depth range of the system in terms of depth accuracy.

Figure 4.6: Error variation with distance

Figure 4.6 shows the average error obtained for the varying distance of a planar surface in front of the system. The plot shows the errors are less within a specific range of depth and increase outside this range. This range is defined by two main parameters, the depth of field and the distortion parameters of the visual system.

The visual system consisting of the projector and the camera is focussed at a depth of 675mm from the system and hence the errors obtained at this depth lie at the minima of the plot. As the depth is increased or decreased the projected primitives are less focussed on both the object at that depth and also on the camera plane. This causes errors in the accurate segmentation of each primitive leading to inaccurate determinations of their column locations in the camera plane and thus the corresponding errors in depth measurements. Furthermore, errors increase due to the distortion factors which are more prominent in regions away from the principal point of the image plane. For depths with higher errors, as shown in Figure 4.6, a large number of primitives lie in the distorted regions of the camera plane further adding to the errors obtained.
4.4 Obstacle detection

For obstacle detection it is necessary to not only determine the depth but also the physical dimensions of the mapped surface. Once the depth of the environment is mapped, the remaining two dimensions need to be computed.

Figure 4.7: World coordinates for obstacle detection

Figure 4.7 shows the reconstruction of an obstacle in terms of the world coordinates. The obstacle here is the leg of a table at a depth of 675mm from the system and a planar surface at a depth of 700mm to the right of the leg. The 3D coordinates marked in red shows an obstacle since it falls in within the working dimension of 173mm x 331mm of the system. The coordinates marked in green represent object points which are not obstacles to the system.
4.5 Complete system integration

The integration of all the modules together yields the obstacle avoidance system. Once the depth map is computed, the observable environment is translated into 3D coordinates with respect to the origin fixed at the edge of the robotic platform. This is then used in identifying the presence of an obstacle. Finally, the minimum distance to be traveled to the obstacle is computed and drive commands are sent to the robotic platform.

Figure 4.8: Platform movement with obstacle

Figure 4.8 shows the result obtained when the system encounters a wall in its environment. The obstacle (wall) is placed at a distance of 685mm from the global origin positioned as explained in Section 3.3.5. The distance the platform has to travel reduces to 570mm due the position of the origin with respect to the front edge of the platform. The path traversed by the platform is shown in green with the starting and ending points represented as green and red points.

The distance to the obstacle computed by the system was 560mm, found to be with an error of 10mm. The platform, however, travels a distance of 565mm and stops 5mm from the obstacle. It is also observed that on increasing the velocity of the platform the error increases. This is a shortcoming of the drive system of the platform and the error can be accounted to mechanical factors such as friction and slippage between the wheels of the platform and the surface. Furthermore, as can be seen from the figure, the path traversed is not a straight line even though the drive commands were specified for a straight line motion, which is another drawback of the platform. These shortcomings need to be investigated to reduce the errors obtained with the system.
4.6 Execution time distribution

The algorithmic efficiency in depth computation defines the efficiency of the entire system. With an aim of deployment of the obstacle avoidance system in real time environments it is necessary to identify the compute intensive parts of the algorithm and optimizing those to achieve high performance.

Figure 4.9 shows the time consumption of the individual blocks of the algorithm. This plot shows the distribution without any applied optimizations. As can be seen the most compute intensive part is the correlation based grid segmentation followed by the neighborhood detection. Grid segmentation involving cross-correlation, as can be seen from Equation 3.2, contains Multiply And Accumulate (MAC) based operations and thus DSPs having architectures optimized for MAC operations are ideal for the segmentation part of the algorithm. The other algorithm subparts such as neighborhood detection and validity check require small windows for their operation and thus can be performed in parallel independently using the NEON SIMD vector extension on the BeagleBoard. Using such mappings to the various cores, the workload can be more evenly distributed leading to faster implementations and a higher throughput.
CHAPTER 5

Conclusion and Future Work

This thesis details the design and implementation of an obstacle avoidance system for a robotic vacuum cleaner prototype using embedded components. It is based on the principle of depth estimation using the active vision technique of structured light. This structured light based system shows accurate depth estimation over a range of surface objects under varying ambient light conditions which shows its advantages over the conventional stereoscopic vision based depth estimation. It is able to estimate the depth of an obstacle with up to 10mm accuracy for obstacle depths ranging from 600 to 800mm and is able to determine the presence of an obstacle of size 20mm x 18mm at a projection distance of 675mm. This is found to be adequate for a number of real-world obstacles, expected to be encountered by a vacuum cleaning system. The results obtained using the current system also demonstrate that by using a simplified calibration procedure and principle, depth can be accurately measured and obstacles can be detected.

For the design of a mobile obstacle avoidance system, the components selected need to be embedded in nature. The entire system consists of a robotic platform, a visual system to provide the depth information input, and an embedded computing unit to analyze the input and give drive commands to the platform. The robotic platform selected was the Create from iRobot which allows navigation based on the reception of drive commands through a serial interface. It is found to be accurate to distance commands at low drive velocity but this accuracy decreases with an increase in drive velocity. The visual system consists of a Pico projector from Texas Instruments that allows an easy construction of any desired projection pattern and a µEye USB camera from iDS that gives more control over its internal parameters. The embedded computing unit selected for the current project is the BeagleBoard by Texas Instruments which contains peripheral connections to interface with all the other components of the system and hardware to satisfy the performance needs. The OS was selected based on the availability of compatible drivers for the camera. The placement of the visual system on the robot platform was another critical aspect of the system design with the finalized system setup as shown in Figure 3.5.

The selection of the structured light pattern marks another critical aspect of the system design with the pattern defining the robustness and the accuracy with which an obstacle is detected. The pattern finally selected is shown in Figure 2.10. It is a two-dimensional and spatially multiplexed pattern to enable motion robustness, and binary in pixel depth to work in color non-neutral environments. The pattern is absolute-neighborhood coded with spatial primitives and a small neighborhood to allow for a unique identification of all pattern points which minimizes surface discontinuity errors.
Future works

The main drawback of the current system is the repeated need for calibration. This happens due to the unstable nature of the projector and camera mountings which change their orientation due to any applied forces. A simple solution to this problem is the use of fixed mountings instead of the variable ones used in the current development setup.

To improve the performance of the entire system the algorithm can be split and distributed on multiples cores. The correlation based grid segmentation can be performed on the DSP core optimized for such operations. The neighborhood detection and the validity check operations are based on individual windows and thus can be performed in parallel on the SIMD vector extension for ARM using the provided NEON instruction set available on the BeagleBoard. Using an efficient algorithm mapping on such a heterogenous architecture can provide orders of magnitude improvement in the algorithm implementation and such mappings need to be investigated for an increased throughput for the entire system.

Results show a reduction in robustness of the current system with an increase in the ambient light in the observable environment. A possible solution to improving the robustness is by using an Infrared (IR) visual system. However, IR wavelengths are also present in ambient light and can be emitted by other sources present in the environments the current system is aimed for. Hence, the use of imaging at different light spectrum needs to be investigated with an aim to improve the robustness of the entire system.

Another aspect which can be investigated is the use of a higher resolution visual system. A higher resolution projector will be capable of projecting finer patterns reducing the individual size of the primitives, thereby reducing the decoding window size and thus the shape discontinuity errors giving a more accurate object mapping along its edges. A higher resolution visual system also promises to increase the spatial resolution of the system allowing the depth mapping of smaller obstacles. However, the need of increasing the spatial resolution depends on the definition of an obstacle which in turn depends on the application.


