Vision-Based Localization using Reliable Fiducial Markers

By:

Alexandros Stathakis

A thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements for the degree of

Master of Applied Science

in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

School of Electrical Engineering and Computer Science

University of Ottawa

December 2011

Abstract

Vision-based positioning systems are founded primarily on a simple image processing technique of identifying various visually significant key-points in an image and relating them to a known coordinate system in a scene. Fiducial markers are used as a means of providing the scene with a number of specific key-points, or features, such that computer vision algorithms can quickly identify them within a captured image. This thesis proposes a reliable vision-based positioning system which utilizes a unique pseudo-random fiducial marker. The marker itself offers 49 distinct feature points to be used in position estimation. Detection of the designed marker occurs after an integrated process of adaptive thresholding, k-means clustering, color classification, and data verification. The ultimate goal behind such a system would be for indoor localization implementation in low cost autonomous mobile platforms.

Acknowledgements

I would like to thank my supervisor, Dr. Emil M. Petriu for his guidance and support during my research term. Without him this thesis would not have been possible. Secondly, I would like to thank my colleagues at SMRLab, such as Phillip Curtis, Jamieson McCausland, Valentin Borsu, and Fouad Khalil for their productive and useful discussions. Lastly, I would like thank all my family and friends, who were always there for support and encouragement.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AR	Augmented Reality
CCD	Charge-Coupled Device
CRC	Cyclic Redundancy Check
GF	Galois Field
GPS	Global Positioning System
HD	Hamming Distance
HOM	Hoffman Marker
HSV	Hue-Saturation-Value
IGD	Institute Graphische Datenuerarbeitung
PRA	Pseudo-Random Array
PRS	Pseudo-Random Sequence
QR	Quick Response
RGB	Red-Green-Blue
RS	Reed Solomon
SCR	Siemens Corporate Research
SLAM	Simultaneous Localization And Mapping
SVD	Singular Value Decomposition

Chapter 1. Introduction

This chapter provides motivation for the work documented in this thesis. It also details the main objectives of the work completed, as well as the organization and contribution of the thesis.

1.1. Motivation

New advances in modern technologies have brought about the growth of a large research field identified as Machine Vision. Machine Vision, otherwise known as Computer Vision, is concerned with the analysis and computation performed on digital images. Among the main aspects/topics associated with Machine Vision is marker registration. The marker registration process consists of calculating the transform between a known object in the environment and the camera that has captured it. This becomes extremely useful for many applications, such as: augmented reality, robotic navigation and scene recognition, medical imaging systems, and video compression. This thesis aims to focus in on using marker registration as a means of robotic navigation.

The world of marker registration is vast, encompassing various types of markers. Among the simplest is the planar marker, more specifically fiducial markers. Fiducial markers are generally objects placed within a scene to provide visual cues or reference points such that a computer vision system can easily isolate the object's location. Currently fiducials are used to supply reliable position estimates for robotic and imaging registration applications [**1**], as well as

augmented reality [**2**] [**3**]. Use of machine vision technology for indoor localization aims to provide a few advantages over conventional localization techniques.

Popular outdoor localization techniques such as GPS (Global Positioning System) cannot be used for indoor localization due to the attenuation of satellite signals from physical obstacles such as walls and ceilings [**4**]. When looking to the research field for indoor localization methods, one can find the following:

- Radio frequency propagation [**5**], [**6**]
- Ultra-sonic sound Propagation [**7**]
- Infrared projection [**8**] [**9**]

However, these techniques require their own infrastructure for both transmitting and receiving various signals that propagate through space. Moreover, the cost to fully implement the aforementioned techniques could be relatively high. This partially comes about from the large amount of error associated with the localization methods which in turn require a larger number of transmitters and receivers for continuous position updates throughout the environment. Computer vision solutions aim to provide cost efficient systems that offer a higher degree of accuracy or precision. The proposed solution presented in this thesis aspires to obtain an accurate estimation of the position of a rigid body in a 3D setting as well as offering robustness to partial occlusion.

The idea behind the suggested pose estimation technology is to become an integrated part of an autonomous robotic system used to navigate known

indoor environments. Estimating the robot's position in the environment is crucial to path planning and navigation. The existence of fiducial markers within the environment will provide a method of location synchronization between the robot and the marker.

Current techniques used to detect fiducially markers rely on identifying some visually significant key-points that can be extracted on a frame by frame basis. The main issue apparent in these techniques arises from improper feature point selection; such that not all feature points are unique. This thesis presents an extension of a pseudo-random encoding technique presented in [**10**], which is used to create a 2D array of colored grid points described by Pseudo-Random Arrays (PRAs).

Another important factor to consider in such localization systems is the fact that indoor scenes often contain dynamic objects, which are objects that are in constant motion within the environment. A planar 2D marker will often be subject to occlusion from these dynamic objects resulting in problematic marker detection. The design of the aforementioned marker goes even further to include 60 data bits, some of which are responsible for error correction in cases of occlusion. The data bits are encoded first using Cyclic Redundancy Check (CRC) and then Reed Solomon (RS) error correction.

The proposed marker consists of a 9 x 9 2D array containing 49 distinct PRA features. Each feature consists of 9 elements where elements are one of three possible colors: red, green, or blue. The marker system supports over 4096 planar markers for each unique pseudo-random array. Marker detection occurs

when a minimum of five feature points are detected within a color image, followed by the digital decoding of the 60 bit data payload. Data bits are represented on the marker by pink colored circles (digital ‘1’) or empty white space (digital ‘0’) and are located in between the elements of the PRA. Figure 1 illustrates the detection of the marker system within an office environment.

Figure 1.1: Sample image of marker detection. Detection of the proposed marker system used to navigate an autonomous robot within an office laboratory environment.

1.2. Objectives

The objective behind the research completed for this thesis is to design a robust fiducial marker system, whose primary purpose is to serve as reference points, or synchronization points, in an indoor localization method. The

localization system uses a basic known layout of the environment along with strategically place markers throughout the building. The system itself is to be designed using low cost consumer grade hardware as to minimize the overall price. The position of a mobile node/platform in the environment will be inferred through the detection of the fiducial marker, while other image processing techniques will be used to provide slight odometry corrections when a marker is not in the camera's field of view.

Throughout this work, the following goals will be satisfied:

1) A new fiducial feature space will be defined and integrated into the design of a planar marker.
2) Advance digital encoding and decoding techniques will be use to increase the marker's robustness to noise and data loss.
3) A resource efficient marker detection algorithm will be proposed, integrating conventional pose estimation techniques.
4) Additional image processing methods will also be proposed for full integration of the newly designed fiducial marker into an indoor localization system for autonomous robotic platforms.
5) Various perform measures (data restoration and position estimation) of the fiducial marker will be analyzed in order to determine its feasibility a localization system.

1.3. Thesis Organization

This work is separated into seven main chapters. The second chapter provides a review of existing techniques used for fiducial markers. In Chapter

three the theory for the newly designed fiducial marker system is derived, while the following chapter details the proposed marker detection algorithms. Chapter five goes on to explain various additional algorithms required to implement the fiducial into an odometry based localization system. The next section establishes the results of experimental data and analyzes it in order to determine the marker's practical implementation as a localization system. The final chapter concludes this work as well as providing suggestions for future work to be performed. Three of four attached appendices provide extra information on singular value decomposition, homography estimation, and finite fields, while the last appendix supplies the data set used to verify the inter-marker confusion test of chapter five.

Chapter 2. Background and Related Work

This chapter provides an introduction to many of the fiducial marker systems that are currently present in the research field, as well as any theory associated to marker detection and decoding. The basic detection process of various marker types will be outlined along with any downfalls and benefits. The chapter will then go on to explain how the position of a fiducial can be computed using the camera pin hole model. Lastly, embedding information into fiducials markers to improve performance will be discussed.

2.1 Introduction

A number of planar marker systems currently exist, embedding important information by encoding complex patterns. Prime examples of these general purpose markers are MaxiCode and Quick Response (QR), illustrated in figure 2.1 (a) and (b) [**11**].The MaxiCode system being an invention of the United States Postal Service used to convey shipping information, while the QR marker is used in industrial settings for parts labeling. The purpose of these types of marker systems is to carry information and not for localization. The above two encoding methods use error correction schemes in order to recovery data bits which have become corrupt. Even though they provide a slight robustness to occlusion, they are not suitable for use as fiducials. The dense composition of the markers creates an inability to be detected in a large field-of-view with perspective distortion [**11**]. Secondly, they require a fairly large area in the image, limiting the

range in which markers can be detected. The above two planar markers do however share common characteristics with various fiducial markers systems.

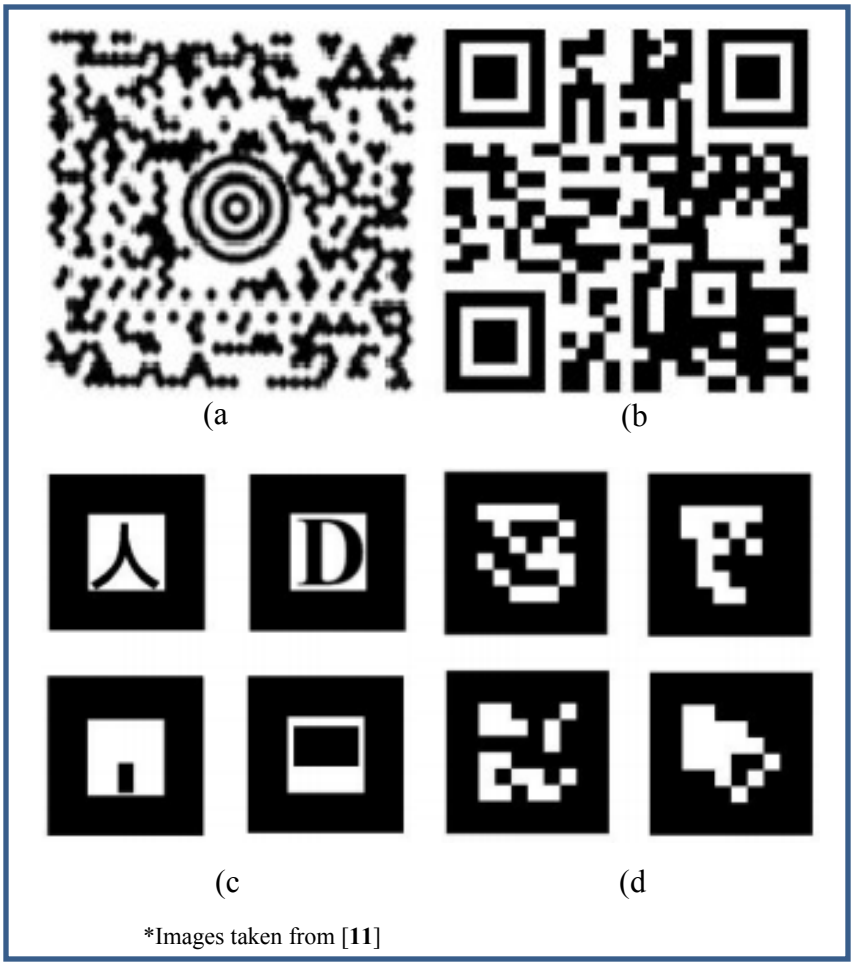


Figure 2.1: Sample planar marker systems (a) MaxiCode (b) Quick Response (QR) (c) Examples of ARToolKit (d) Examples of ARTag

A fiducial marker differs from a simplistic 2D planar marker through its use of position information. That is to say, fiducials act to recover some sort of pose information as well as a data portion, while planar markers tend to be used for data recovery only. The basic algorithms for maker detection, encoding and decoding are often comparable. Marker detection through machine vision utilizes

feature points to uniquely identify a marker in the scene. Error correcting codes are then used to encode and decode information onto the surface of the marker. The following procedure summarizes the detection process for fiducial markers:

1) Feature Extraction: extracts unique points within the image, which are then are used to identify whether a marker is contained within the scene.
2) Pose Estimation: uses registration techniques along with multiple uniquely identified key-points to recovery the relative position of the camera with respect to the marker.
3) Data Decoding: data bits from the marker are recovered using an image sampling procedure. These data bits are then processed to retrieve useful information, such as a marker identification number.

2.2 Feature Extraction

Marker detection begins with capturing an image of the scene. This image however, is not guaranteed to contain a marker. In order to determine whether or not a marker is contained within an image, various unique key-points must be identified. In the general purpose marker MaxiCode (figure 2.1 (a)), unique key-points are created by a series of black and white circles with increasing radius. The combination of all key-point is labeled as a feature, or feature point of the marker. Upon detecting the feature points, one can say with a certain probability, that the marker is contained within the image. In comparison with MaxiCode the QR marker system uses 3 distinct feature points to identify the marker. These feature points are recognized by a black quadrilateral, followed by a white border which is in turn surrounded by another thick black border.

Two well-known marker systems used in augmented reality applications that can be classified as fiducials are ARToolKit [**12**] [**13**] and ARTag [**14**] (Illustrated in figure 2.1 (c) and (d) respectively). Both markers are based on a bi-tonal system composed of black and white regions. It is often preferred to work with black and white markers as opposite to color because color images tend to be sensitive to illumination variations in the environment [**15**] [**16**]. Feature points of these two fiducials are more commonly seen than that of QR and MaxiCode. Feature points in ARTag are identified first by locating the outer black border quadrilateral. More specifically, each of the outer corners of the black border provides one key-point. Hence ARToolKit and ARTag contain a total of four key-points composing one feature. In fact many fiducial marker systems implement feature points similar to ARTag and ARToolKit as will be seen in the remaining sections of this work.

In [**17**] Zhang *et al.* compared the performance of four marker systems: ARToolKit, HOM system, IGD marker systems, and Siemens Corporate Research (SCR) marker system. All four evaluated systems made use of a large black border similar to that of ARTag. The exception to this was the SCR marker, which added the major advancement of using eight feature points as opposed to four, in attempt to assist in marker detection and pose estimation accuracy. Although the markers described in [**17**] used very similar feature points they differed in how information was encoded onto the markers. These encoding methods will be discussed section 2.4. Figure 2.2 illustrates the four markers

discussed by Zhang *et al*. Other black border markers are fairly common and are often seen in literature [**11**] [**12**] [**17**] - [**18**] some of which are shown in figure 2.3.

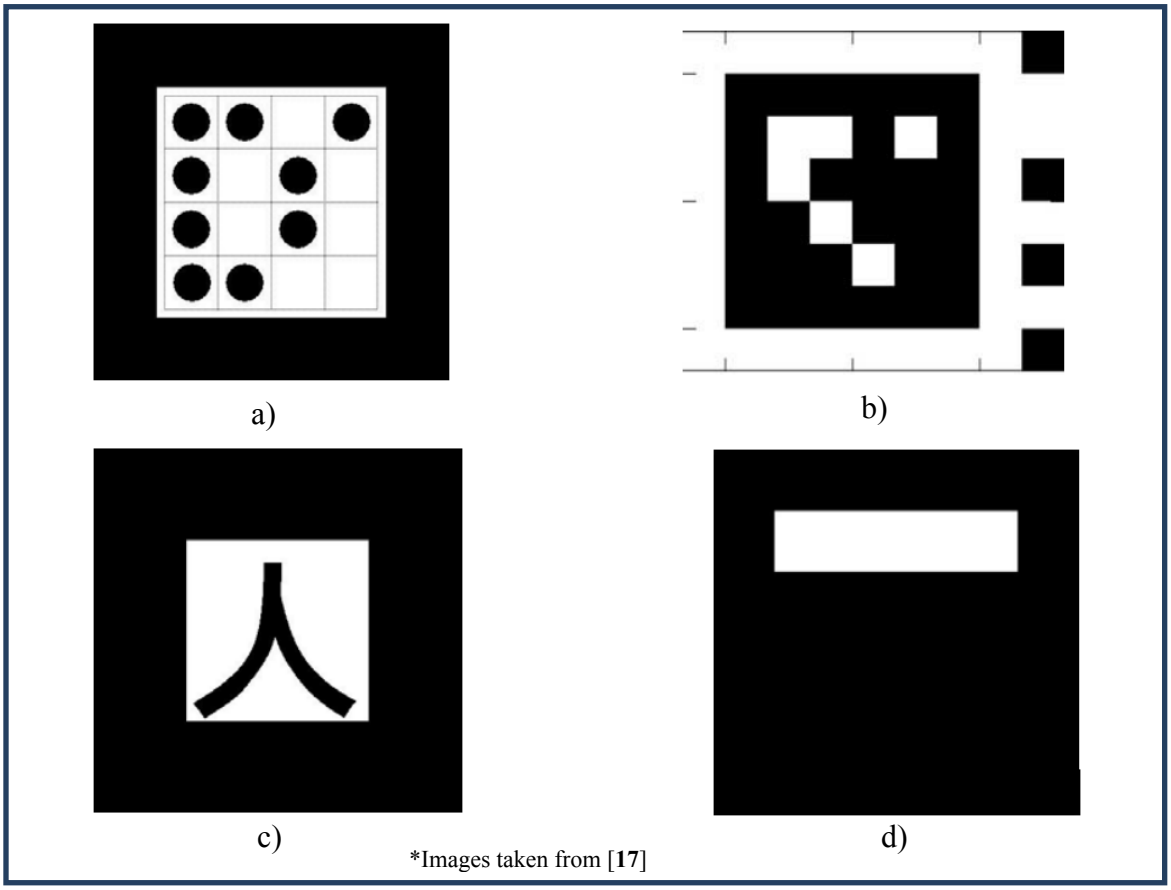
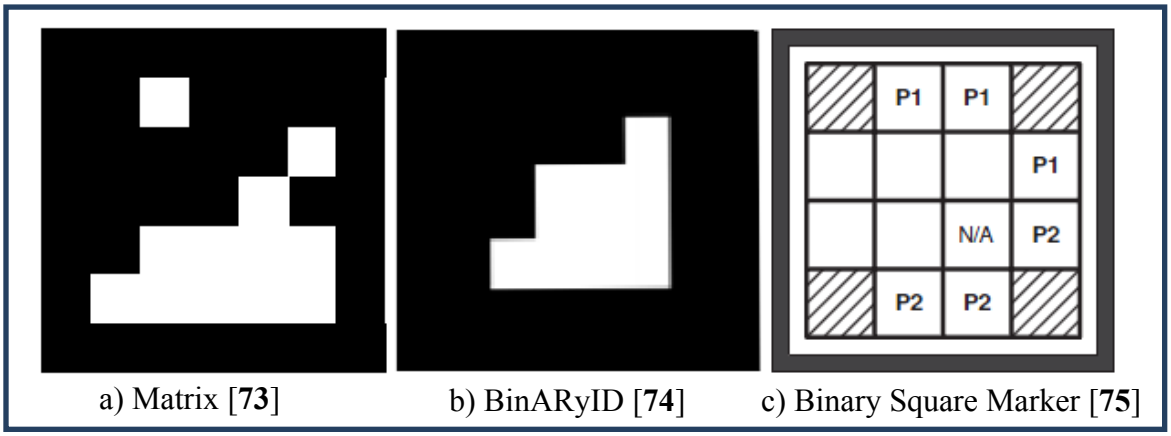


Figure 2.2: Black border markers presented in [**17**] a) SCR marker b) HOM marker c) ARToolKit d) IGD marker

Black border markers are not the only fiducials that are utilized for augmented reality applications. In fact many geometrical shapes can be used to identify a marker within a scene. In the following papers detailing Cybercode [**19**] and Visual Codes [**20**], the authors introduce the use of *blob detection*, finding close contours within a binary image, to filter out and identify long bars via the

second order moment statistics. In the following marker systems: Fourier tag [**15**], Cantag [**21**], Intersense [**22**], Isotropic marker [**23**], Multicolor ring [**24**], and RUNE-tag [**25**], ellipse fitting techniques are used to find circles within the image. The marker systems then use the detected circles as feature points in the identification process. However, when performing ellipse fitting on a natural scene it is common to find many ellipses that are not part of the fiducial but rather the environment. Circles and ellipses are frequently found in man-made structures as well as just about any indoor environment, which is why detected circles must go through a filtering procedure in other to quickly eliminate circles that are not part of the fiducial. This filtering process is accomplished through the analyzing of ellipse parameters such as size, perimeter, and compactness ratio. The last step in circular fiducial detection is to decode the marker's data payload.



a) Matrix [**73**] b) BinARyID [**74**] c) Binary Square Marker [**75**]

Figure 2.3: Various black border markers

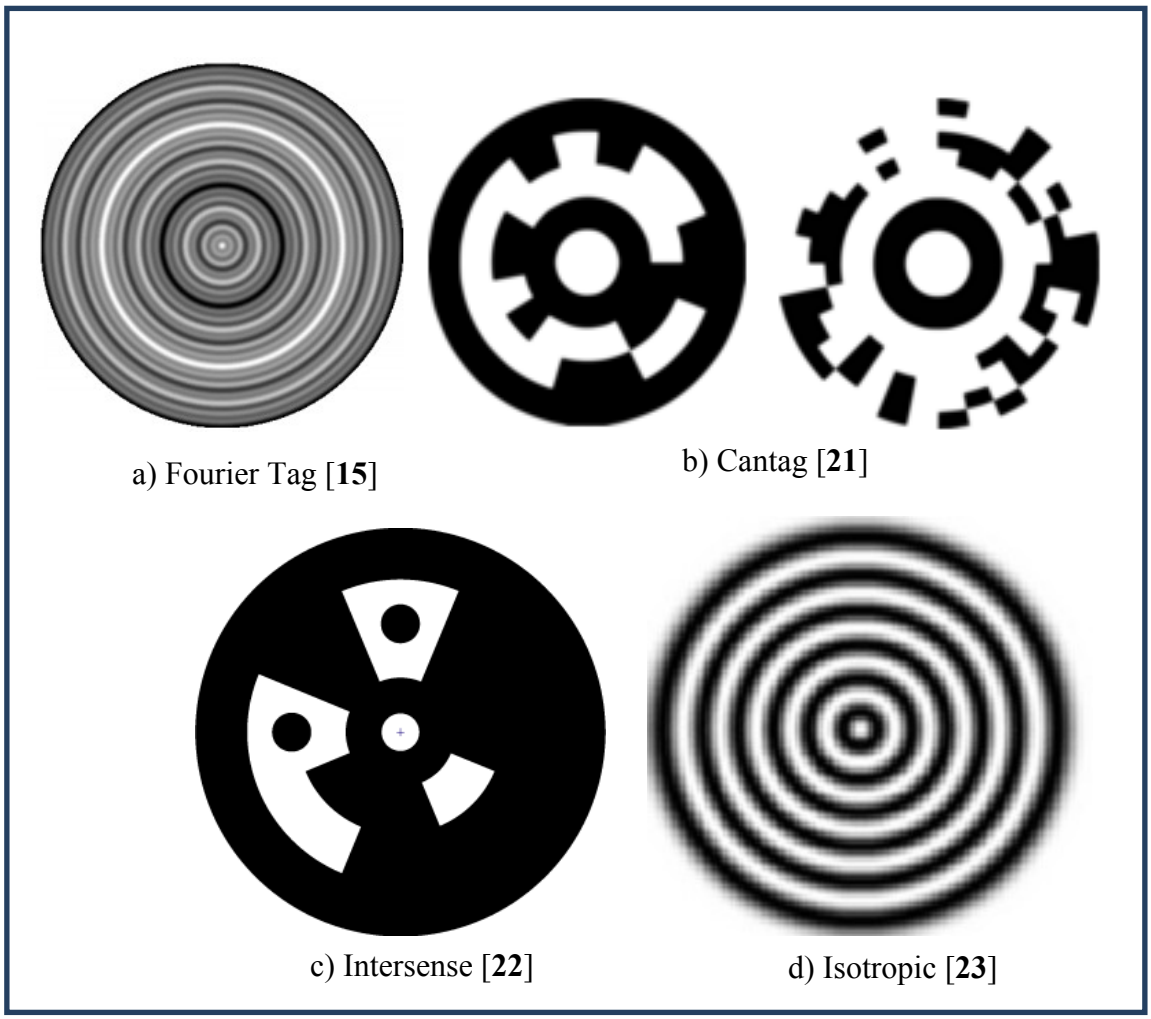


Figure 2.4: Various circular fiducial markers

Both ellipses and quadrilateral feature points are not very unique within indoor environments. One can expect to see shapes of various sizes within almost any complex scene. This makes it difficult to correctly identify markers within a scene and often leads to high false positive detection rates. A false positive detection occurs when a marker is detected but no physical marker is present. False positives pose huge problems when using fiducials for

autonomous navigation and localization, possibly causing the robotic platform to make incorrect navigation decisions. This is one of the issues that this thesis hopes to address. Inclusion of unique feature points will effectively low the false positive detection probabilities and define the most important step in the design process. Fiducials using simple feature points such as ARTag and Intersense tend to rely on one of two things: data encoding techniques to lower the probability of false positives or limiting the amount of squares/circles in the scene. Other planar markers will often implement distinct feature points to decrease the probability of confusing the marker with the environment. Equally important as feature point exclusivity is the marker uniqueness. Markers should be distinctive from other markers in the same fiducial marker library, such that they can be quickly distinguished from each other.

Feature points are used in machine vision to determine regions of interest. In the case of fiducials, a region of interest points or locates the data payload section of the marker. Feature points are also used in determining the position of the marker relative to the camera; this will be explained further in section 2.3. Various factors can affect the detection of key-points, they include:

- *Variations in lighting conditions*: The lighting in a scene can affect the amount of light that is reflected into the camera's aperture, which in turn may affect the appearance of colors and shades within an image.
- *Cluttered scenes*: When a scene contains objects of various shapes and sizes it is often difficult for an algorithm to correctly separate key-points of the marker from similar points in the image.

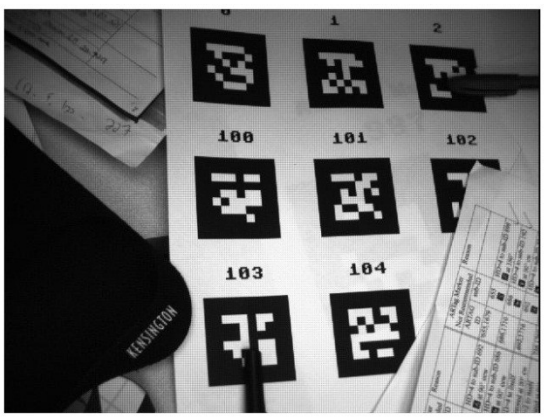
- *Camera focus*: An unfocused camera will often cause objects within an image to be blurred affecting the pixel intensities of feature points to be spread across a larger area.
- *Camera calibration*: Un-calibrated cameras suffer from various types of distortion, ie. Radial and perspective distortion. This leads to warping or twisting of feature points.

2.2.1 Detection of black border feature points

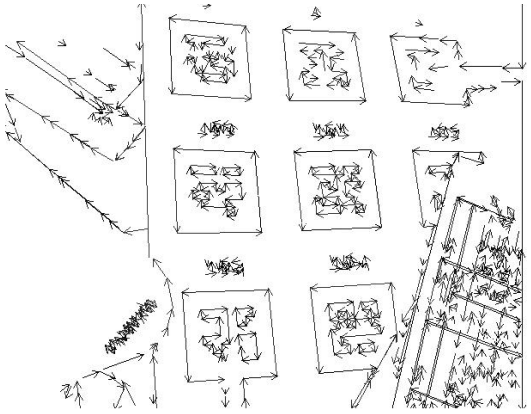
Thus far, most of the markers presented in this thesis are bi-tonal, black and white, which makes feature point extraction fairly simple. The black border, or black circle in the case of markers such as Intersense, can be identified using several techniques. One of the more precise detection methods is revealed in [**26**] through a fiducial marker known as Studierstube. The method is based on that of ARTag.

ARTag uses an edge based routine in order to determine the four corners of the quadrilateral that composes the black border feature point. Edges in the image are defined as sharp variations in pixel intensities. In the most basic scenes edges are found by scanning the image for these sharp changes between black and white, and vice versa. However, important to note is that this process in done on grayscale images. Figure 2.5 illustrates the edge detection computed on a set of ARTag markers. After all edges in the image are found, a filtering process begins to connect various line segments. Closed contours are then parsed and filtered for quadrilateral shapes. Quadrilaterals that fit certain predefined size requirements are considered to be feature points. The detection

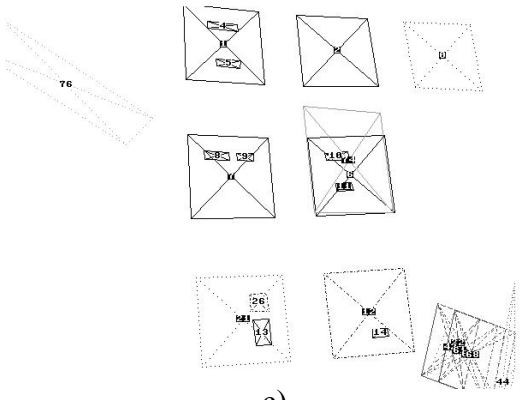
procedure can be very similar for detecting circular features as well, only that ellipse filtering is used instead of quadrilateral filtering.



a)



b)



c)

*Images taken from [**11**]

Figure 2.5: ARTag quadrilateral detection a) original greyscale image of various ARTags b) result of edge detection c) after filtering for quadrilaterals

Upon detection of the fiducial's feature points, an algorithm to determine the real world position of the marker is executed. Making the assumption that one knows where the camera lies in the environment, determining the location of the marker boils down to finding the rotation and translation between the fiducial and the camera. The resulting transformation is labeled as the marker's pose. A

planar marker's pose can be determined using a number of coordinates on the marker and the corresponding pixels in the image frame of the camera.

2.3 Pose Estimation

In order to fully grasp the concept of estimating the pose of a fiducial marker in 3D space using a correspondence between the image plane and the ideal maker, one must have a basic understanding of the camera pin-hole model. The pin-hole model describes the mathematical equations that relate a point in 3D space to its corresponding projection onto the 2D image plane of the camera. It is through this model that the key-points of a marker can be used to find the rotation and translation vectors between the fiducial and the sensing device. It is through this same model that camera calibration occurs. There are many approaches towards finding this 2D to 3D correspondence; however a basic explanation of the pinhole model will be discussed first.

2.3.1 Pin-Hole Model

The pin-hole camera model is used to describe the geometric mapping from 3D to 2D space. This mapping is label as the perspective projection. It is important to note that this model is used as a relatively simple way to explain the concept of how a camera is able to record 3D objects. The main assumption here is that light enters the camera through a very small aperture. One can think of a large vertical plane, or wall, with a small pin hole through it. On one side of this wall is the camera's photosensitive plane (ie. CCD array, CMOS array) that is used to record the image. On the other side of this wall is the scene that one

wishes to capture. Light enters the camera only through the very small pin hole. Thus, the camera is only able to see a certain viewing angle of the scene.

Regrettably, this model is not completely accurate. A pinhole is much too small to gather enough light to expose all pixels of the photosensitive plane, or image plane. It is for this reason that cameras make use of larger aperture sizes and lenses to focus more light onto the area of the photosensitive plane. The inclusion of the added light entering the aperture and the distortion effects of lenses move far beyond the simplistic geometry of this pinhole model. Thus for the time being we will be overlooking these problems by making the assumption that the camera is using a pinhole sized aperture. Figure 2.6 illustrates the pinhole model.

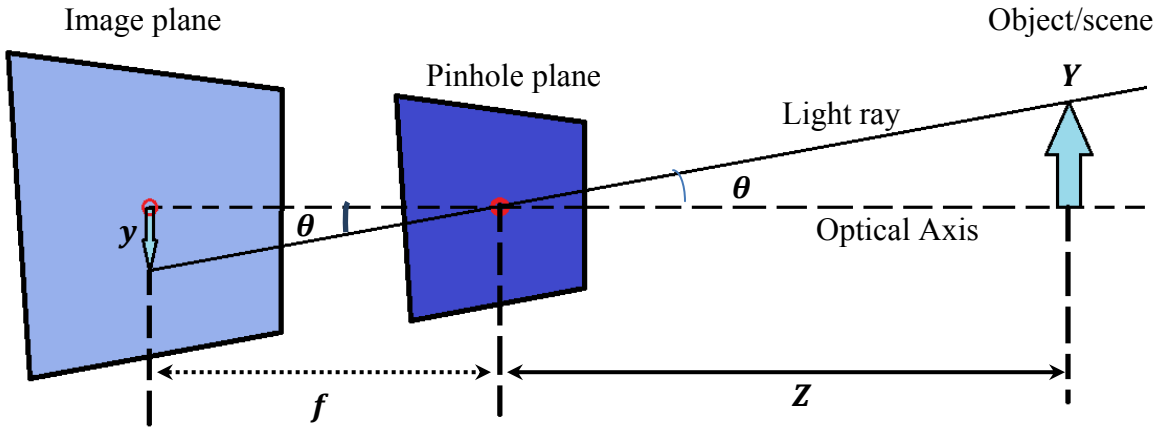


Figure 2.6: The original pinhole camera model

In the pin hole model light rays get reflected off various objects in the scene only to enter the camera through the pinhole. However, the assumption here is that only one light ray enters from a single point in 3D space. That is to say that diffraction is ignored. Thus points in 3D space are project onto the 2D

image plane forming a duplicate. The distance from the image plane and the pinhole plane is given by the focal length of the camera, f. The model can be seen in figure 2.6; where $\boldsymbol{Z}$ indicates the distance from the camera to the object in the scene, $\boldsymbol{Y}$ is the height of the object, and $\boldsymbol{y}$ is height of the object in the image plane. For convenience the model is often altered such that the image plane is placed on the opposite side of the pinhole plane. This is done to ensure the scene is projected right side up on the image plane. Figure 2.7 illustrates the modified pinhole model.

Using similar triangles and some trigonometry, a relationship between $\boldsymbol{y}$ and $\boldsymbol{Y}$ can be obtained from figure 2.6.

$$\tan\theta = \frac{-y}{f} \tag{2.1}$$

$$\tan\theta = \frac{Y}{Z} \tag{2.2}$$

Equation 2.1 is obtained from the intersection of the optical axis and the light ray incident on the image plane. Similarly equation 2.2 is acquired from the intersection of the optical axis and the light ray reflected from the top of the object in the scene. Substituting equation 2.1 into 2.2 yields equations 2.3 and 2.4.

$$\frac{-y}{f} = \frac{Y}{Z} \tag{2.3}$$

$$-y = \frac{fY}{Z} \tag{2.4}$$

Using the modified pinhole model of figure 2.7 equation 2.4 changes to become:

$$y = \frac{fY}{Z} \tag{2.5}$$

For the duration of this thesis we will be using the modified pinhole model of figure 2.7.

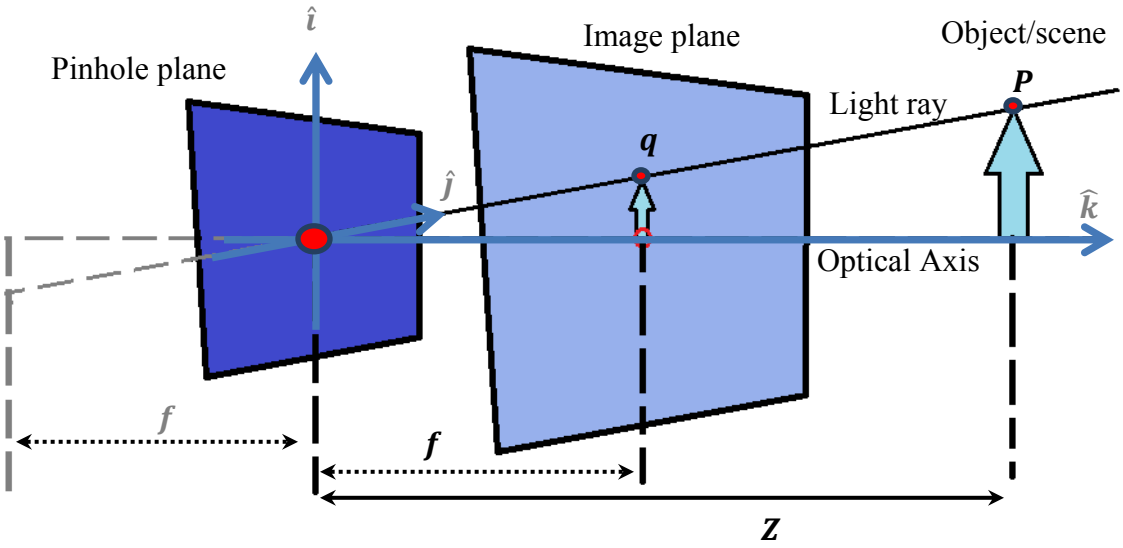


Figure 2.7: Modified pinhole model. The Image plane has been inverted and moved in front of the pinhole plane. One can think of this plane as being translucent and recording light intensity of light waves passing through the plane.

In the new modified pinhole model all physical measurements stay the same with the exception of the inversion of the image. The image plane now lies in front of the pinhole plane at same distance as the focal length. The pinhole is labeled as the center of projection, which is the point in space to which all light is directed. The intersection of the optical axis and the image plane is called the principal point. In this model we can think of light rays reflecting off distance objects and intersecting the image plane on their way to the center of projection.

The result is a variation of light intensities recorded on the image plane. These intensities can be interpreted as an image.

An important factor to consider in the pinhole model is that the principal point is not always located at the center of the image plane. This is mainly due to the manufacturing process of cameras. A point $\boldsymbol{P}$ located in space, with coordinate $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$, is projected onto the image plane at pixel value of (x_{pixel}, y_{pixel}). In order to express the pixel coordinate in terms of the 3D world coordinates we must use equation 2.5 and add a term to compensate for the fact pixel coordinates belong to the natural number set. The added terms $c_x and\, c_y$ specify the shifts in x and y pixel coordinates in order to place the origin of the image plane in the top left corner of the surface, which happens to be convention for image processing.

$$y_{pixel} = f\left(\frac{Y}{Z}\right) + c_y \tag{2.6}$$

A similar equation exists for the x coordinate pixel location and is define as:

$$x_{pixel} = f\left(\frac{X}{Z}\right) + c_x \tag{2.7}$$

It is common for most low cost photosensitive arrays not to be to perfectly square but rather rectangular [**27**]. Thus two new focal length terms must be added to our model in order to compensate, f_x and f_y. The focal length f_x is determined by the physical focal length of the camera and the size of individual pixels on the image plane's x-axis, s_x. A comparable relation exists for the focal length f_y and the parameter s_y. Thus we obtain the follow equation that relates pixel coordinates of the image plane to metric coordinates in 3D space.

$$x_{pixel} = f_x\left(\frac{X}{Z}\right) + c_x, \quad where\ f_x = fs_x \tag{2.8}$$

$$y_{pixel} = f_y\left(\frac{Y}{Z}\right) + c_y, \quad where\ f_y = fs_y \tag{2.9}$$

A 2D point on the image plane is denoted by $\boldsymbol{q} = [x_{pixel},\ y_{pixel}]^T$. A 3D point in space is denoted by $\boldsymbol{P} = [X, Y, Z]^T$. It is often convenient to express the relations of equations 2.8 and 2.9 in matrix form. The combined relations are known as the projective transform.

$$\boldsymbol{q} = \begin{bmatrix} x_{pixel} \\ y_{pixel} \end{bmatrix}, \qquad \boldsymbol{P} = \begin{bmatrix} X \\ Y \\ X \end{bmatrix}, \qquad \boldsymbol{q} = \boldsymbol{M}_{int}\boldsymbol{P} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.10}$$

The matrix M_{int} is used to transform a point from pixel coordinates to metric coordinates in the camera's reference frame. This matrix is also known as the camera's intrinsic matrix. Once pixel coordinates can be transformed to metric coordinates, a second transform is needed to describe the relation between two arbitrary reference frames in space.

2.3.2 Transforms in 3D space

Movement in 3D space can be broken down into a series of three translations and three rotations. Thus any transform is explained with six degrees of freedom. Translations and rotations are defined with respect to the X, Y and Z axes. A rotation about the Z axis is defined as roll, θ. A rotation about the Y axis is the pitch, φ, while a rotation about the X axis is defined as yaw, ψ. The definitions for transformations in 3D space are illustrated in figure 2.8.

For convenience all points will be written in homogeneous coordinates. This means that a point in 3D will be defined by 4 terms, and a 2D point by 3

terms. Converting $\boldsymbol{q}$ and $\boldsymbol{P}$ to homogeneous coordinates and multiplying by a scale factor leads to following homogeneous vectors defining 2D image points and 3D world coordinates. In most cases the scale factor is assumed to be 1, which is what will be assumed for this thesis.

$$s\bar{\boldsymbol{q}} = s\begin{bmatrix}\boldsymbol{q}\\1\end{bmatrix} = s\begin{bmatrix}x_{pixel}\\y_{pixel}\\1\end{bmatrix}, \qquad \bar{\boldsymbol{P}} = \begin{bmatrix}X\\Y\\Z\\1\end{bmatrix} \tag{2.11}$$

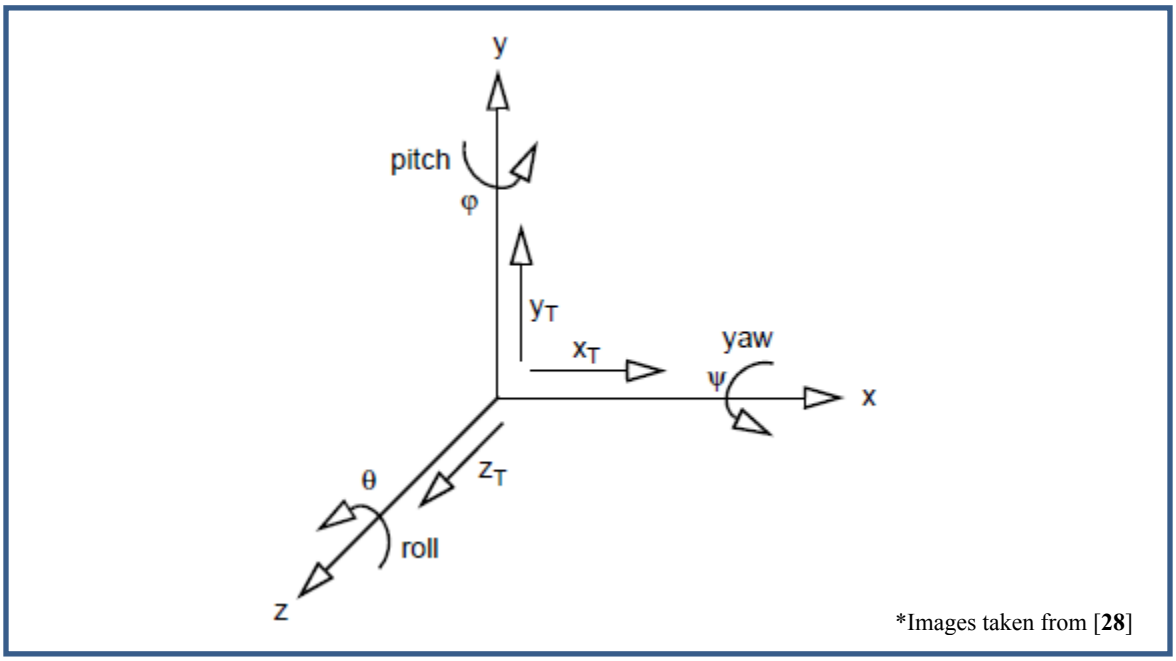


Figure 2.8: Six Degrees of Freedom (DoF) for motion in 3D space

Equation 2.10 relates a point in 3D space to point on the image plane. However, in order to properly calculate the pose between the camera and the fiducial marker we need to define the function between the reference frame of the camera and the reference frame of the marker, or more generally an arbitrary frame in space. Let us consider two reference frames in space, as illustrated in figure 2.9. The 'base' frame is considered to be fixed; we will think of this to be

the reference frame of the camera. The 'mobile' frame on the other hand is able to move around the environment freely. This will be the reference frame of the fiducial marker.

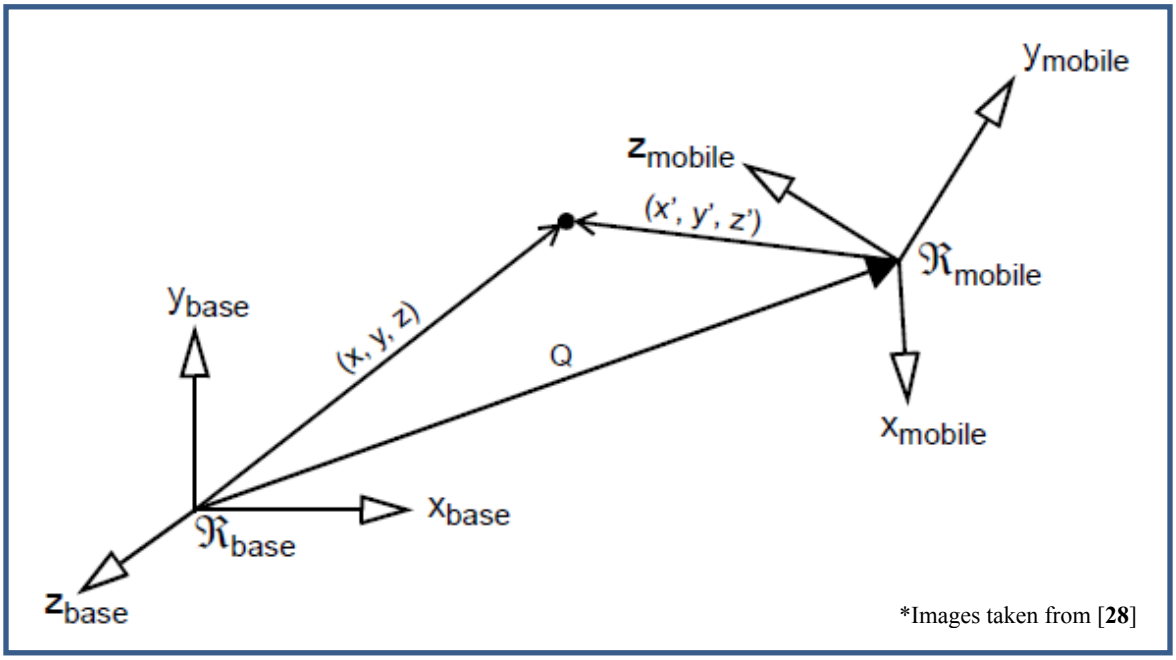


Figure 2.9: Visual relationship between two reference frames. Q defines the transformation between the base frame and the mobile frame.

From [**28**] the homogeneous matrix transform, Q, in figure 2.9 is defined as such:

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{T} \\ \boldsymbol{S} & \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

The matrix $\boldsymbol{Q}$ is comprised of four key elements. The sub-matrix $\boldsymbol{R}$ contains nine components. These components are used to encode the three rotation angles. The sub-matrix $\boldsymbol{T}$ contains three components which correspond to the three translations about the x, y and z axes. The sub-matrix $\boldsymbol{S}$ holds three scaling

elements which for this thesis will all be set to zero. The last element in the matrix $\boldsymbol{Q}$ has been placed to ensure a square and invertible matrix.

A point in the mobile frame can be converted to the base frame via the following formula:

$$\begin{bmatrix} x_{base} \\ y_{base} \\ z_{base} \\ 1 \end{bmatrix} = \boldsymbol{Q} \begin{bmatrix} x_{mobile} \\ y_{mobile} \\ z_{mobile} \\ 1 \end{bmatrix} \tag{2.13}$$

Let us now defined a matrix $\boldsymbol{M_{ext}}$, which is a modification of the matrix $\boldsymbol{Q}$. Eliminating the sub-matrices $\boldsymbol{S}$ and $\boldsymbol{I}$ from the transform $\boldsymbol{Q}$ leads to the matrix, $\boldsymbol{M_{ext}}$. The matrix $\boldsymbol{M_{ext}}$ is used in the same way as $\boldsymbol{Q}$ however the base frame coordinates are no longer homogeneous.

$$\begin{bmatrix} x_{base} \\ y_{base} \\ z_{base} \end{bmatrix} = \boldsymbol{M_{ext}} \begin{bmatrix} x_{mobile} \\ y_{mobile} \\ z_{mobile} \\ 1 \end{bmatrix}, \quad where\ \boldsymbol{M_{ext}} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{2.14}$$

Combining equations 2.14 and 2.10, a new equation is developed which relates the pixel coordinates of the image frame to the 3D coordinates of a mobile reference frame.

$$s\overline{\boldsymbol{q}} = s\boldsymbol{M_{int}}\boldsymbol{M_{ext}}\widetilde{\boldsymbol{P}} = s \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x_{mobile} \\ y_{mobile} \\ z_{mobile} \\ 1 \end{bmatrix} \tag{2.15}$$

In the equation of 2.15 a point $\widetilde{\boldsymbol{P}}$ in the scene is represented in the coordinate system of some mobile reference frame, while the point $\overline{\boldsymbol{q}}$ represents the image coordinates, in pixels, within the reference frame of the camera. Equation 2.15 can be rewritten as the following:

$$s\begin{bmatrix} x_{pixel} \\ y_{pixel} \\ 1 \end{bmatrix} = s\boldsymbol{M_{int}}[\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{r}_3 \quad \boldsymbol{t}]\begin{bmatrix} x_{mobile} \\ y_{mobile} \\ z_{mobile} \\ 1 \end{bmatrix} \tag{2.16}$$

- $\boldsymbol{r}_1$ is a 3 by 1 vector containing the first column of $\boldsymbol{M_{ext}}$

- $\boldsymbol{r}_2$ is a 3 by 1 vector containing the second column of $\boldsymbol{M_{ext}}$

- $\boldsymbol{r}_3$ is a 3 by 1 vector containing the third column of $\boldsymbol{M_{ext}}$

- $\boldsymbol{t}$ is a 3 by 1 vector containing the forth column of $\boldsymbol{M_{ext}}$

Equation 2.16 can be used to describe the relation between the camera's reference frame and the reference frame of a fiducial marker.

2.3.2 Homography between the Model Plane and its Image

The problem of determining the matrix $\boldsymbol{M_{ext}}$ is very similar to camera calibration. There are three main methods in literature that have been purposed to solve this problem:

1) Linear Algorithms [**29**] [**30**]
2) Non-linear Algorithms [**31**] [**32**]
3) Iterative Algorithms [**33**]

A popular non-linear process purposed by Zhang in [**34**] allows one to determine the rotation and translation of a planar object within a camera's field of view. A key assumption to this approach is that all points in the mobile reference frame have a z_{mobile} value of zero. That is to say, the reference frame has its x-y axis in line with the plane of the fiducial marker. Hence the equation of 2.16 changes to the following:

$$s\begin{bmatrix} x_{pixel} \\ y_{pixel} \\ 1 \end{bmatrix} = s\boldsymbol{M_{int}}[\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{r}_3 \quad \boldsymbol{t}]\begin{bmatrix} x_{mobile} \\ y_{mobile} \\ 0 \\ 1 \end{bmatrix} = s\boldsymbol{M_{int}}[\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{t}]\begin{bmatrix} x_{mobile} \\ y_{mobile} \\ 1 \end{bmatrix} \tag{2.17}$$

A new matrix $\boldsymbol{H}$ is defined such that

$$s\begin{bmatrix} x_{pixel} \\ y_{pixel} \\ 1 \end{bmatrix} = \boldsymbol{H}\widetilde{\boldsymbol{P}} \tag{2.18}$$

Hence,

$$\boldsymbol{H} = s\boldsymbol{M_{int}}[\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{t}] = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = [\boldsymbol{h}_1 \quad \boldsymbol{h}_2 \quad \boldsymbol{h}_3] \tag{2.19}$$

The matrix $\boldsymbol{H}$ is a 3 by 3 matrix known as the homography matrix. A homography matrix defines a transform between two planes in space. There are many ways to estimate the homography matrix between a plane and the image that captured it. One approach presented in [**34**] uses a maximum-likelihood estimation of $\boldsymbol{H}$. A separate approach using homogeneous linear least squares is shown in Appendix B. Since $\boldsymbol{M_{int}}$ is already known through camera calibration of the system, the rotation and translation parameters can be found through the following equations.

$$\boldsymbol{r}_1 = s\boldsymbol{M_{int}^{-1}}\boldsymbol{h}_1 \tag{2.20}$$

$$\boldsymbol{r}_2 = s\boldsymbol{M_{int}^{-1}}\boldsymbol{h}_2 \tag{2.21}$$

$$\boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2 \tag{2.22}$$

$$\boldsymbol{t} = s\boldsymbol{M_{int}^{-1}}\boldsymbol{h}_3 \tag{2.23}$$

$$s = \frac{1}{\left\|\boldsymbol{M_{int}^{-1}}\boldsymbol{h}_1\right\|} = \frac{1}{\left\|\boldsymbol{M_{int}^{-1}}\boldsymbol{h}_2\right\|} \tag{2.24}$$

However the downside of the calculated $\boldsymbol{R}$ matrix is that it is not unique, due to noise in the data used to calculate it. The best rotation matrix can be found through techniques that take advance of Singular Value Decomposition. A basic concept of Singular Value Decomposition is explained in appendix A.

Once the rotation and translation matrices are determined the fiducial marker's position with respect to the camera is known. If a marker is fixed in the environment, in a known configuration, then the position of the camera can be determined. This application is very important in developing navigation of vision based robotic systems where cameras are mounted to the mobile platform. In [**35**] and [**36**] planar fiducial markers are used as landmarks for real-time single camera Simultaneous Localization And Mapping (SLAM). Mulloni *et al*. in [**37**] present the idea of using fiducial markers for indoor navigation with cell phones. The central idea of using fiducial markers as a tool in indoor navigation is not new.

To properly determine the homography matrix and in turn the pose of the marker at least four points are needed. More points will help eliminate noise and increase accuracy of the results especially when using linear techniques. Many of the square black border markers explained in this thesis use four points with sub-pixel accuracy to compute the planar homography. The four points used are usually the four corners of the outer thick black border of the fiducial. However a problem arises when part of the marker is occluded. ARToolKit and ARToolKitPlus [**38**] lack the robustness for occlusion of any of corner of the marker. ARTag on the other hand provides a limited robustness to occlusion by

estimating missing segments of the border. Studierstube builds on the approach ARTag using line detection, line extension and line grouping to detect corners. It still however has the problem of occlusion for more than one corner.

Topology based fiducial markers such as reacTIVision [**39**] tend to have irregular shapes, thus accurate pose information can't be retrieved. Square and circular markers have their own advantages. Circular markers use the whole circular contour to determine pose, as opposed to only four vertices. Thus making them more robust to occlusion while offering a tendency to be more accurate in pose estimation [**40**]. However, pose information in not always unique due to rotation invariance of circles. The pose information computed from quadrilaterals often provides unique solutions [**41**]. Once pose estimation of the marker is obtained, it is used to sample the inner data payload of the marker. This payload contains important information that can be used to distinguish between markers in the system.

2.4 Data Decoding

When there is more than one marker in the fiducial's database, homography estimation is not the last step in marker detection. A method to differentiate between markers is needed. This is especially important in applications that require large fiducial databases, such as indoor navigation. One popularized solution is to encode a series of data bits within the marker [**11**] [**38**] [**40**]. Another approach that could be applied to marker separation is some sort of pattern recognition and correlation. In either approach, a data region must be added to the marker. This added data region not only aids in differentiating

between markers in the same system but it also assists in decreasing the false positive probability.

As mention previously, choosing unique feature points decreases the false positive detection rate. Marker systems such ARTag and ARToolKit however do not provide unique feature points, since quadrilateral shapes are fairly common in complex environments. Thus without the inclusion of the data area, one would expect to see fairly high false positive probabilities. In case of ARToolKit, a correlation method is used to distinguish between markers. The four corners of the marker's border are used to calculate the homography of the planar surface. This homography then allows for the projection of predefined marker sample coordinates into the image plane of the camera. The coordinates of marker are taken from the inner data region and compose a total of 256 (or 1024) points. Once the samples are recorded, they are correlated with a set of stored marker prototypes. Each stored marker prototype is a 16 x 16 (or 32 x 32) grid of sample points from an ideal marker. The correlation process can be thought of a ranking process between markers by computing the dot product of the recorded sample points and the stored marker prototypes. Figure 2.10 illustrates how a grid of sample points can be obtained from an ARToolKit marker.

The approach used by ARToolKit has proven to be susceptible to noise leading to a large false positive detection rate and inter marker-confusion. Generally correlation based fiducials are the least robust [**40**]. Users for this type of marker system are required to create and capture ideal prototypes for each marker under the conditions that the markers will be utilized. Also important to

note is that as the marker library size increases, the uniqueness of individual markers decreases. The processing time for detection also increases with the increase in the marker system's library. This is due to the correlation process of comparing the current sample points to all marker sample points in the library. Thus, ARToolKit is not suitable for applications requiring reliability. In an attempt to decrease the false positive detection rate of ARToolKit, ARToolKitPlus was developed. ARToolKitPlus adopts a new data payload technique.

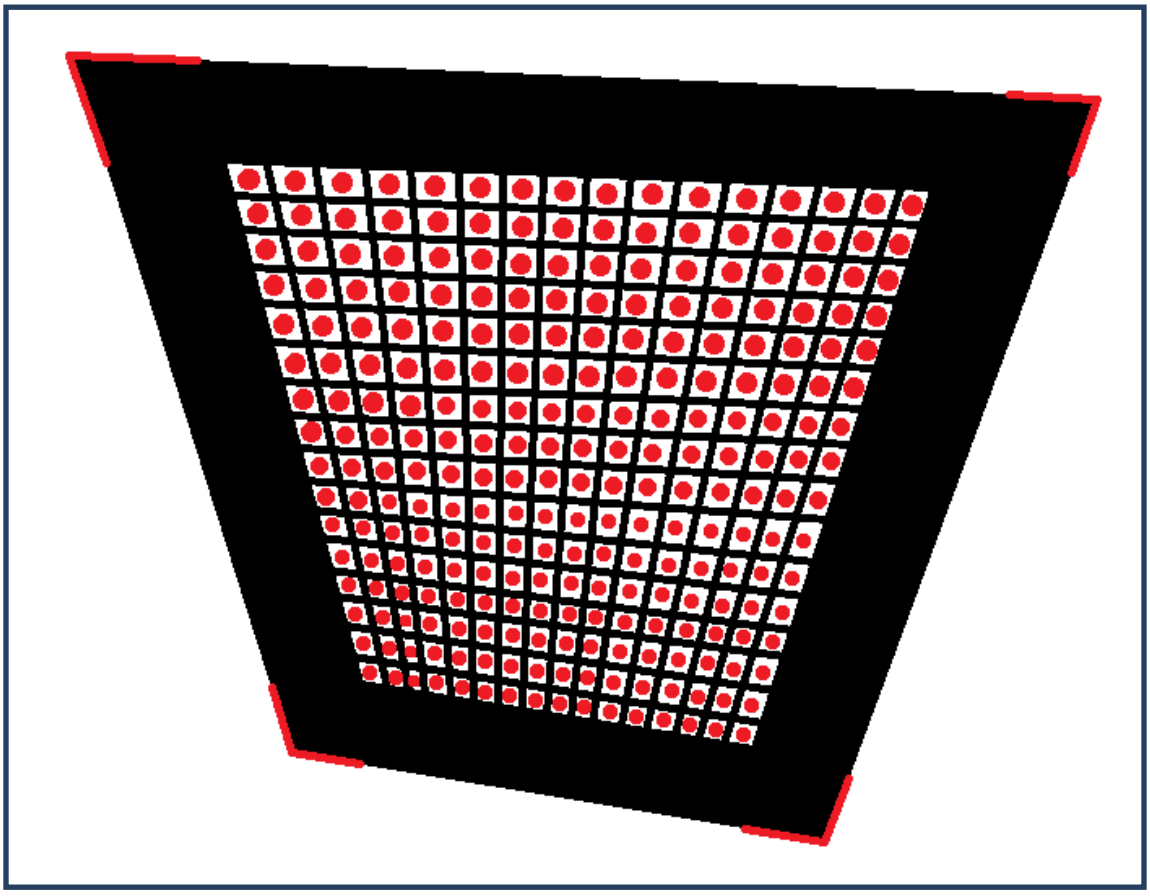


Figure 2.10: ARToolKit sampling pattern for correlation matching. Red dots indicate the 256 sampling points used for correlation.

ARToolKitPlus and ARTag both use a 6 by 6 inner grid region to distinguish between markers. Each cell in the grid is filled with one of two shades, black or white, and indicates a data bit. For ARTag, ten of the 36 data bits are reserved for the marker identification number. The remaining 26 bits are used for Cyclical Redundancy Check (CRC) and forward error correction. Forward error correction bits are able to correct up to two bit errors, while the CRC bits are added as a check sum to ensure that the data sequence has indeed been corrected properly. ARToolKitPlus on the hand has three options, one for each marker type available. The first approach repeats the correlation process of ARToolKit. The second is labeled “Simple ID” and supports only 512 markers (9 bits), where the 36 data bits are filled by repeating a 9 bit marker ID four times. The third approach “BCH encoding”, allows up to 4096 markers [**38**]. Similar techniques are used by many bi-tonal marker systems. Binary Square Marker for example uses a 4 x 4 grid. The corner cells of Binary Square Marker’s grid are reserved to determine orientation of the marker, leaving 12 bits to encode data on the marker. Only five of the twelve bits are used to represent the marker identification number. The remaining bits are implemented as a checksum to correct single bit errors.

The marker detection technique used for CyberCode [**19**] on the other hand implements markers of arbitrary size. Orientation of the marker is determined by a black bar located on one of four sides of the marker. Corners of the marker can then be determined using a simple thresholding approach and the inner data sampling region. The HOM, IGD and SCR marker systems all use

data regions comparable to ARToolKitPlus. IGD and SCR both use a 4 x 4 data region encoding up to 16 bits. While HOM includes a 4 x 4 data area with an additional 6 bits place one side of the marker [**17**]. The sizes of these marker libraries are not detailed. Circular fiducial markers also utilize data regions, however in a different manner.

Circular markers such as Cantag and Intersense use concentric circular rings split into sectors in order to store data bits. Each ring is split evenly into an arbitrary amount of sectors and data bits are expressed by either a black or white filled sector. Cantag implements four marker types: one square and four circular. The first of the circle markers is called CircleInner and contains a solid black ring on the inside of the marker, which is surrounded by the data payload. The second of the circular markers is called CircleSplit. CircleSplit has an inner and outer black circular ring. The data payload of this marker is stored in between the two rings. The last of the three is labeled CircleOuter. This fiducial has an outer circular ring that surrounds the data payload. Each solid black ring is used to determine the homography of the planar marker. Once a homography has been obtained the projective transform is used to sampling data sectors. Figure 2.11 illustrates the three circular marker types of Cantag, as well as the sampling pattern for the data payload. Each cell containing a red dot indicates a cell that contains a data bit. The Cantag marker itself does not define a specific data payload or encoding technique. Intersense marker on the other hand supports multiple predefined payload configurations.

Similar to CircleSplit, Intersense uses an inner and outer ring design. The diameter of the marker is designed to be 8 length units in diameter, *8u*. By default both the outer and inner rings are colored as black and have a thickness of *1u*. The inner ring makes up a circle of diameter *2u*, which is colored white. This leaves two rings of *1u* in thickness to be used for the data payload. The data rings are split into 8 even sectors, two of which are colored white and contain two small black circles. These two cells are separated by one black sector and are used to indicate the x and y axis of the marker. The black sector functions as a separator as well as to provide full connectivity between all black portions of the marker. The remaining 5 sectors are used for information bits. Each of the data sector contains 3 data bits, which are indicated by black and/or white quasi-trapezoidal cells. This translates to $2^{(3*5)}$ = 32768 possible binary codes. However if a larger marker library is needed, the colors of the outer and inner rings can be alternated to one of the following configurations: black-white, black-black, and white-black. This further increases the marker size to 3 x 2^{15} = 98304. Encoding techniques for this marker are not specified and would be left for the user to define. Figure 2.4 c) illustrates an example of an Intersense marker.

Different fiducial markers each provide their own advantages and disadvantages. For an example, Intersense can possibly provide a marker library of 98304 distinct markers. However, this library would not include any robustness to occlusion and therefore would not be suitable for indoor localization. Other fiducial markers such as ARTag are able to handle some occlusion however

even this is limited to 2 bits. Robustness to occlusion is one of the main objectives to the data payload and encoding techniques. Another objective is to lower the false positive probabilities and inter marker confusion rates. This is accomplished by encoding the marker identification number over a larger data payload. There is however a trade-off between the densities of fiducial markers and their range of detection. All these factors are important in consider in the design of a fiducial for indoor localization.

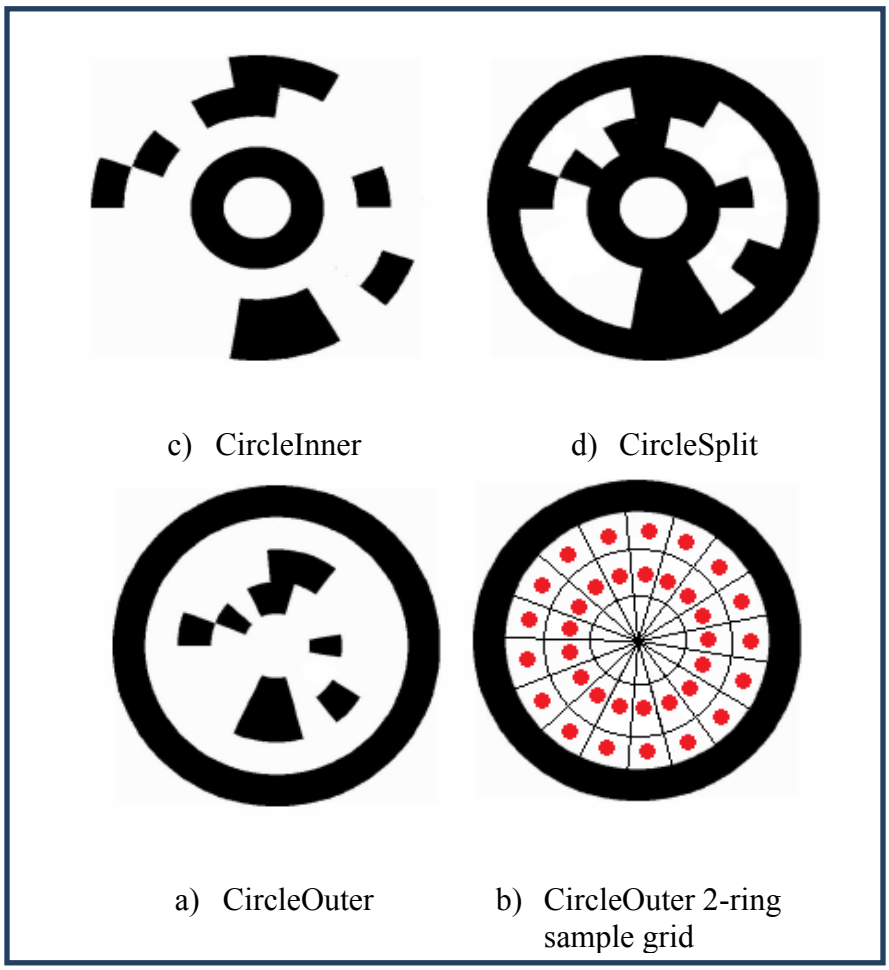


Figure 2.11: Three circular markers of Cantag

.

2.5 Chapter Summary

This chapter provides an overview of fiducial markers and the detection process necessary to identify them in a natural scene. The first section in this chapter distinguishes the difference between fiducial markers and simple planar markers. It also outlines the three components that make up a fiducial. The remaining sections in the chapter analyze these three components by providing examples that can be found in literature.

The first component of a fiducial was identified to be feature detection. This section dealt with identifying the marker within a scene and stressing the importance of choosing unique key-points. More specifically, various fiducial markers were introduced while classifying their feature points and isolating factors that might affect feature point detection. The second section of the chapter emphasized the significance of pose estimation. This part analyzed the projection of a scene onto the image plane of a camera, as well as providing key relationships and equations for mapping a planar marker in space to its corresponding image points in the camera's reference frame. The last section introduced various marker identification and data payloads techniques, which play an important role in marker detection rates, false positive probabilities and inter marker confusion.

Chapter 3. Design of a Pseudo-Random Fiducial

This chapter dives into the design and construction of a unique pseudo-random fiducial marker. A general overview of Pseudo-Random Arrays and sequences will be introduced as well as their uses in image processing. A new unique feature space will be identified, providing 49 distinctive feature points. Next, the integrated data region of the proposed marker will be illustrated, defining the marker library size and its data restorability properties. Cyclic Redundancy Check and Reed Solomon encoding techniques are then explained as well as their use in the proposed fiducial marker.

3.1 Pseudo-Random Arrays as a Unique Feature Space

One of the major downfalls of current fiducial systems is the use of poor feature points to uniquely identify the marker. Chapter 2 provides many examples of fiducial markers that use a black border as the feature point in marker detection. This black border approach has two major downfalls. The first is that both quadrilaterals and circles are common shapes that can be found in many complex environments, thus the feature points are not unique. The second fault occurs under partial occlusion of the marker where not enough key-points are found in the image to distinguish the feature point. The designed marker proposed in this thesis utilizes various key concepts seen in many popularized fiducial, such as ARTag, while adopting the feature space of Pseudo-Random Arrays (PRAs). The use of Pseudo-Random Arrays will aim to solve issues of

unique feature point selection and problematic detection under partial marker occlusion.

A Pseudo-Random Array (PRA) is an *M*-ary array or matrix of arbitrary size φ x ψ, containing a number of β x λ unique *M*-ary sub arrays. That is, each sub-array should appear only once within the overall PRA. Similarly, Pseudo Random Sequences (PRS) are defined as an *M*-ary series of M^{β}-1 elements, where each β-tuple of consecutive elements is unique. This concept of uniqueness is known as the window property for PRAs and PRSs; each element along with a number of predefined neighboring elements can be uniquely identified within the overall data structure. Thus, global position information for each element in the PRA or PRS can be obtained by analyzing its neighboring elements. The distinctive window property has made PRSs/PRAs ideal for many applications in absolute position recovery for automated guided vehicles [**42**] [**43**] [**44**] [**45**] [**46**] [**47**] as well as 3D object model recovery using encoded structured light [**48**] [**49**] [**50**] [**51**] [**52**]. This work plans to extend the implementation of PRAs to fiducial markers, comparable to the work of Kim in [**53**].

3.1.1 Pseudo-Random Array Generation

PRAs can be constructed using one of two methods. The first of the two consists of constructing a pseudo random sequence using shift registers followed by a folding procedure, yielding a two dimensional array. This method is explained well by MacWilliams and Sloane in [**10**]. However a disadvantage to this technique occurs when dealing with multi-valued sequences. Each element in a multi-valued sequence takes one of *M* possible values. For example, a

binary sequence (2-ary sequence) consists of a series of 1's and 0's, while a ternary sequence (*M* = 3) consists of a series of 0's, 1's and 2's. Thus any mathematical calculations must be performed in base *M* which may not be as straight forward as binary mathematics. The second approach is one of brute force. This technique is presented by Morano *et al*. in [**50**] and is the one that has been chosen in the construction of the fiducial marker presented in this thesis. For the purpose of this thesis PRAs will be assumed to have the following parameters *M* = 3, ψ = φ = 9, β = λ = 3.

Each element, $\mu_{i,j,}$ within the PRA, *A*, is chosen from one of *M* = *3* possible values (ie. '0', '1', '2'; 'red', 'green', 'blue'; or 'α', 'β', 'γ'). The location of each element within the matrix *A* can be determined via a number of its neighboring elements, with the exception of any bordering elements of the array. This is an outcome of the unique window property of PRAs. Since each unique sub-array, or window, is defined by β x λ elements. The absolution location of any element within the array is defined by a number, χ, of its neighboring elements.

$$\chi = \beta \; x \; \lambda - 1 \tag{3.1}$$

$$\chi = 3 \times 3 - 1 = 8 \tag{3.2}$$

Consequently, the position of each element μ_{ij} is indexed by a nine element vector K_{ij} defined in equation 3.3.

$$K_{ij} = \begin{bmatrix} \mu_{i-1,j-1} & \mu_{i-1,j} & \mu_{i-1,j+1} \\ \mu_{i,j-1} & \mu_{i,j} & \mu_{i,j+1} \\ \mu_{i+1,j-1} & \mu_{i+1,j} & \mu_{i+1,j+1} \end{bmatrix} \tag{3.3}$$

The brute force method described in [**50**] consists of manually creating each overlapping window, K_{ij}, while at the same time ensuring windows are unique. Figure 3.1 illustrates the creation process.

Beginning at the top left of the empty φ x ψ = 9 x 9 matrix, a 3 by 3 window is created. This window is known as the initial 'seed'. The 'seed' is created randomly by choosing one of 3 possible values for each element in the window, as seen in figure 3.1 (b). Elements of this PRA are defined by colored square-like shapes that take on one of three possible colors: red, blue or green. An example of a PRA and its elements can be seen in figure 3.1 (f). Consecutive random columns are then added to the right of the initial 'seed' until no more can be added (Figure 3.1 (c)). A random column consists of random choosing three elements from the palette of the three colors. For each new column added, a new 3 x 3 window is created. New windows are compared with previously created windows to ensure that each 3 x 3 sub-array of the whole PRA is unique. A similar process occurs by adding random 1 x 3 rows of elements below the initial seed. Again this is repeated until no more rows can be added (Figure 3.1 (d)). The horizontal and vertical operations can then be replicated by incrementing the initial 'seed' coordinates by one, and randomly choosing a color for the lower right element of the sub-array (see figure 3.1 (e)). The procedure is completed iteratively until the grid has been completely filled. If any two or more 3 x 3 sub-arrays are identical in their color configuration, then the marker is scrapped and the generation procedure starts again from the beginning.

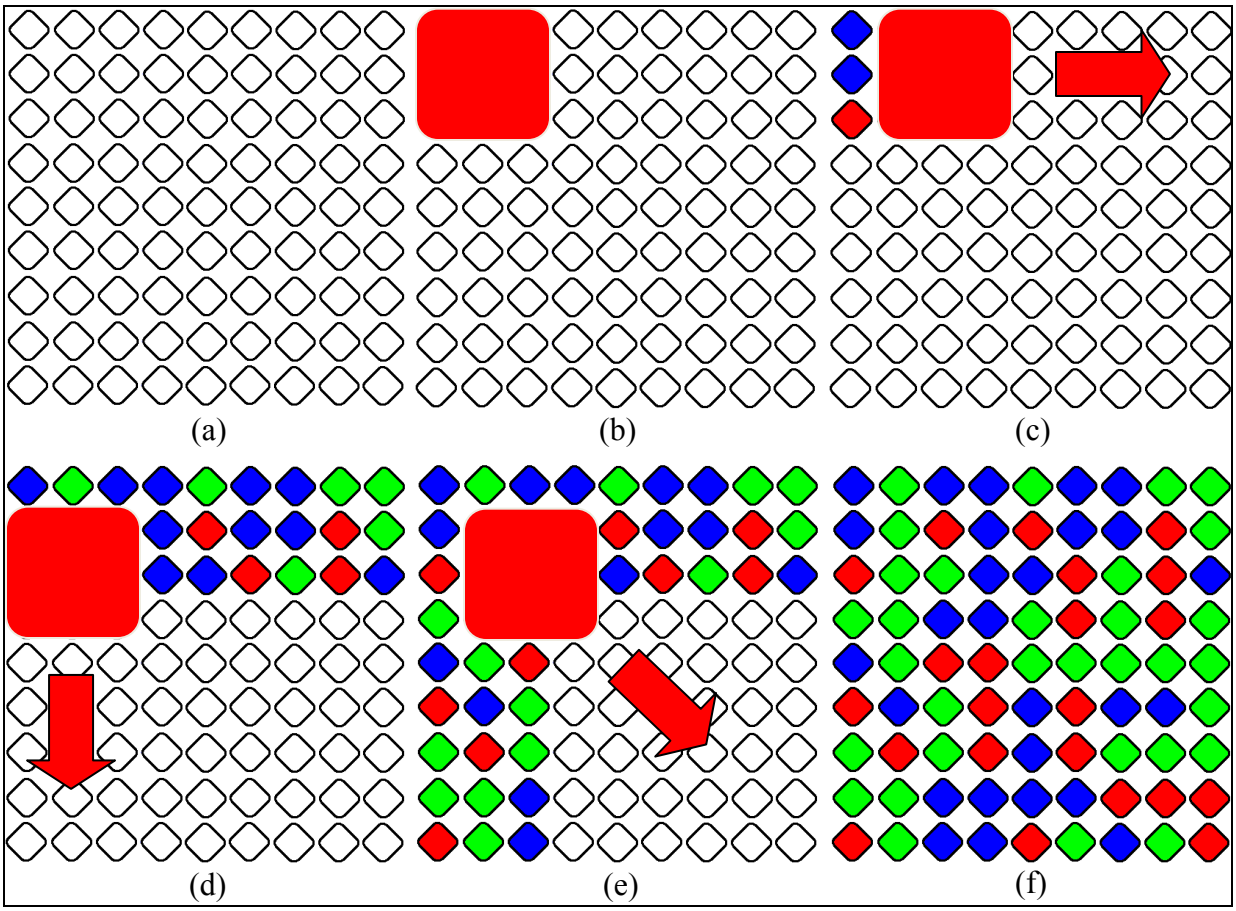
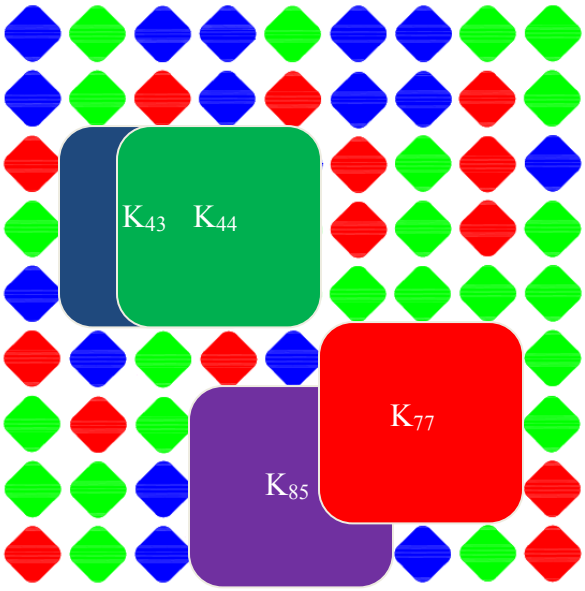


Figure 3.1: Brute force method for creating a Pseudo Random Array. a) the starting empty 9 x 9 array b) Initial setup of the 'seed' used to fill PRA c) Addition of random vertical columns d) Addition of random horizontal columns e) Incrementing of the initial seed to continue the filling process f) The final PRA once element filling has been completed

3.1.2 Pseudo Random Feature Space

Generation of the 9 x 9 PRA of figure 3.1 creates exactly 49 distinct, 3 x 3 sub-arrays. It is these unique sub-arrays that will define a new feature space of the fiducial marker proposed in this work. Each one of the 49 sub-arrays will define one feature point for marker detection. Thus a marker will consist of 49

feature points which can be used to identify whether a marker is present in the scene. Figure 3.2 identifies 4 feature points of the PRA while table 3.1 lists all feature vectors K_{ij} defined for the PRA of figure 3.2.



$$K_{43} = \begin{bmatrix} G & G & B \\ G & B & B \\ G & R & R \end{bmatrix}$$

$$K_{44} = \begin{bmatrix} G & B & B \\ B & B & G \\ R & R & G \end{bmatrix}$$

$$K_{77} = \begin{bmatrix} R & B & B \\ R & G & G \\ B & R & R \end{bmatrix}$$

$$K_{85} = \begin{bmatrix} R & B & R \\ B & B & B \\ B & R & G \end{bmatrix}$$

R = Red, B = Blue, G = Green

Figure 3.2: Identification of PRA feature points and corresponding vectors

Each fiducial marker within this newly proposed marker systems will be equipped with a PRA. Therefore each marker will contain exactly 49 distinctive feature points. The key here is that each feature point provides absolute position information with respect to the marker as a whole, which can be used to determine a correspondence relationship between detected points on the image plane of the camera and points on the ideal planar marker. For an example, upon detection of feature point K_{43} of figure 3.2 one can determine that its center element corresponds to the element in the 4th row and 3rd column of the global PRA, and from there all corresponding elements of the feature can be

determined. The idea here is that once five corresponding feature points have been identified in the image plane of the camera, the marker is assumed to be detected. In other words only five of the total 49 feature points need to be identified in order for marker detection to occur. As a result providing an inherit robustness to occlusion.

	1	2	3	4	5	6	7	8	9		1	2	3	4	5	6	7	8	9
K_{22}	B	G	B	B	G	R	R	G	G	K_{56}	G	R	G	G	G	G	B	R	B
K_{23}	G	B	B	G	R	B	G	G	B	K_{57}	R	G	R	G	G	G	R	B	B
K_{24}	B	B	G	R	B	R	G	B	B	K_{58}	G	R	G	G	G	G	B	B	G
K_{25}	B	G	B	B	R	B	B	B	R	K_{62}	B	G	R	R	B	G	G	R	G
K_{26}	G	B	B	R	B	B	B	R	G	K_{63}	G	R	R	B	G	R	R	G	R
K_{27}	B	B	G	B	B	R	R	G	R	K_{64}	R	R	G	G	R	B	G	R	B
K_{28}	B	G	G	B	R	G	G	R	B	K_{65}	R	G	G	R	B	R	R	B	R
K_{32}	B	G	R	R	G	G	G	G	B	K_{66}	G	G	G	B	R	B	B	R	G
K_{33}	G	R	B	G	G	B	G	B	B	K_{67}	G	G	G	R	B	B	R	G	G
K_{34}	R	B	R	G	B	B	B	B	G	K_{68}	G	G	G	B	B	G	G	G	G
K_{35}	B	R	B	B	B	R	B	G	R	K_{72}	R	B	G	G	R	G	G	G	B
K_{36}	R	B	B	B	R	G	G	R	G	K_{73}	B	G	R	R	G	R	G	B	B
K_{37}	B	B	R	R	G	R	R	G	R	K_{74}	G	R	B	G	R	B	B	B	B
K_{38}	B	R	G	G	R	B	G	R	G	K_{75}	R	B	R	R	B	R	B	B	B
K_{42}	R	G	G	G	G	B	B	G	R	K_{76}	B	R	B	B	R	G	B	B	R
K_{43}	G	G	B	G	B	B	G	R	R	K_{77}	R	B	B	R	G	G	B	R	R
K_{44}	G	B	B	B	B	G	R	R	G	K_{78}	B	B	G	G	G	G	R	R	R
K_{45}	B	B	R	B	G	R	R	G	G	K_{82}	G	R	G	G	G	B	R	G	B
K_{46}	B	R	G	G	R	G	G	G	G	K_{83}	R	G	R	G	B	B	G	B	B
K_{47}	R	G	R	R	G	R	G	G	G	K_{84}	G	R	B	B	B	B	B	B	R
K_{48}	G	R	B	G	R	G	G	G	G	K_{85}	R	B	R	B	B	B	B	R	G
K_{52}	G	G	B	B	G	R	R	B	G	K_{86}	B	R	B	B	B	R	R	G	B
K_{53}	G	B	B	G	R	R	B	G	R	K_{87}	R	B	B	B	R	R	G	B	G
K_{54}	B	B	G	R	R	G	G	R	B	K_{88}	B	B	G	R	R	R	B	G	R
K_{55}	B	G	R	R	G	G	R	B	R										

Table 3.1: A list of all feature points contained within PRA of figure 3.2

In the best case situation where all five detected feature points are overlapping and adjacent to each other (as seen in figure 3.3 (a)) a total of 19 elements of the PRA are used for detection. That is 19 out of 81 elements are

needed for marker detection. From equation 3.4 a total of 76.54% of the marker can be occluded without affecting PRA detection. In the worst case situation where all five detected feature point of the marker share no common elements (as seen in figure 3.3 (b)) a total of 9 x 5 (45) elements are needed for marker detection. From equation 3.5, 44.44% of the marker can be occluded while still maintaining enough visible elements for detection. Hence, designing the fiducial marker around PRAs provides a build in robustness to occlusion.

$$1 - \left(\frac{19}{9\ x\ 9}\right) = 1 - \frac{19}{81} = 1 - 0.234568 = 0.765432 \rightarrow \ \ 76.54\% \qquad (3.4)$$

$$1 - \left(\frac{9\ x\ 5}{9\ x\ 9}\right) = 1 - \frac{45}{81} = 1 - 0.555556 = 0.444444 \rightarrow \ \ 44.44\% \qquad (3.5)$$

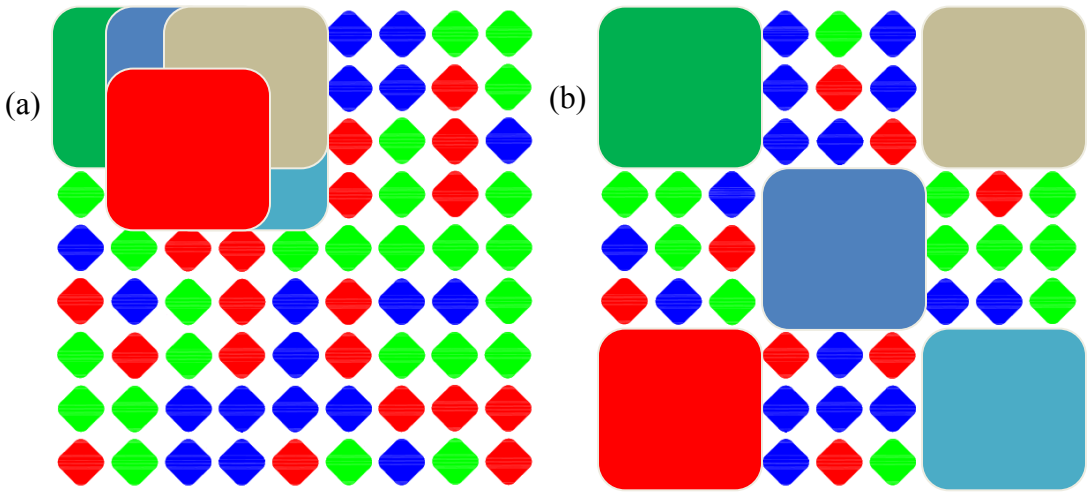


Figure 3.3: Minimum feature points needed for marker detection (a) Best case detection occurs when all detect feature points are adjacent and overlapping (b) Worst case detection occurs when feature points share no common elements

Up to this point the proposed marker systems contains 81 colored square-like shapes arranged in a 9 x 9 grid that composes a PRA. A feature point of the marker is defined as a 3 x 3 gird of elements. It is after the detection of 5 feature points in which a marker is assumed to be identified within a scene. However, one problem that exists is, how various markers in the same system can be distinguished one from another. A feature is comprised of 9 elements, where each element can be one of 3 colors. That leaves a total of 3^9, or 19 683, possible unique feature points that can be created. Assuming that each marker would contain its own unique PRA, that leaves approximately 401 (~19 683/49) markers in the system. The size of this marker system is fairly small when compared against the 2002 library size of ARTag and the 4096 marker size of ARToolKitPlus [**54**]. Thus some modification to the fiducial is necessary. The multi-valued PRA will be used for marker detection while a separate data structure will be implemented for data encoding to distinguish between markers in the same system. That is, each fiducial will contain the exact same PRA of 81 element while markers will be differentiated by the inter data structure described with an internal 60 bit data sequence.

3.2 Internal Data Structure

There are a select few fiducial marker systems using digital encoding techniques to enhance marker performance. One of the many popularized fiducial in the field of Augmented Reality to do so was ARTag. Some of the enhancements ARTag offers as a result of digital encoding include:

- Low inter-marker confusion rates: any one marker in the system is well distinguish from all other markers in the same system
- Low False Positive detection probabilities: the probability of detecting a marker when there is no physical marker present in the scene is low
- Robustness to occlusion: Less of the marker can be visible and still achieve marker detection
- Larger Marker Library: Support for a large number of distinct markers

All of the above attributes are ones which should be addressed in the design of a marker for localization. Thus a total of 60 binary bits have been included in the design of the proposed fiducial. Each of the data bits is conveniently located in between elements of the PRA. The data bit locations within the already design PRA pattern are specified in figure 3.4 (a) by pink and yellow circles. This fiducial marker structure implements two digital encoding/decoding techniques largely influenced by QR codes [**55**] as well as the work of Querini *et al.* in [**56**], which will specify whether bit locations are colored or left as white space. A colored circle represents a logic 'HIGH', or digital '1', while white space indicates a logic 'LOW', or digital '0'.

For the time being PRAs embedded on the marker are used as a means to quickly and uniquely identify feature points within a scene and not specifically to distinguish between markers within the marker system. Each marker will contain the exact same PRA pattern. Markers can be differentiated using the encoded 60 bit data pattern. The purpose behind this is to increase the number of distinct markers available. Twelve of the 60 bits are associated with a marker

identification number, or a Marker ID. Since each bit has one of two possible values the Marker ID ranges from 0 to $2^{12} - 1$ (4095). Consequently the system supports a total of 4096 unique markers.

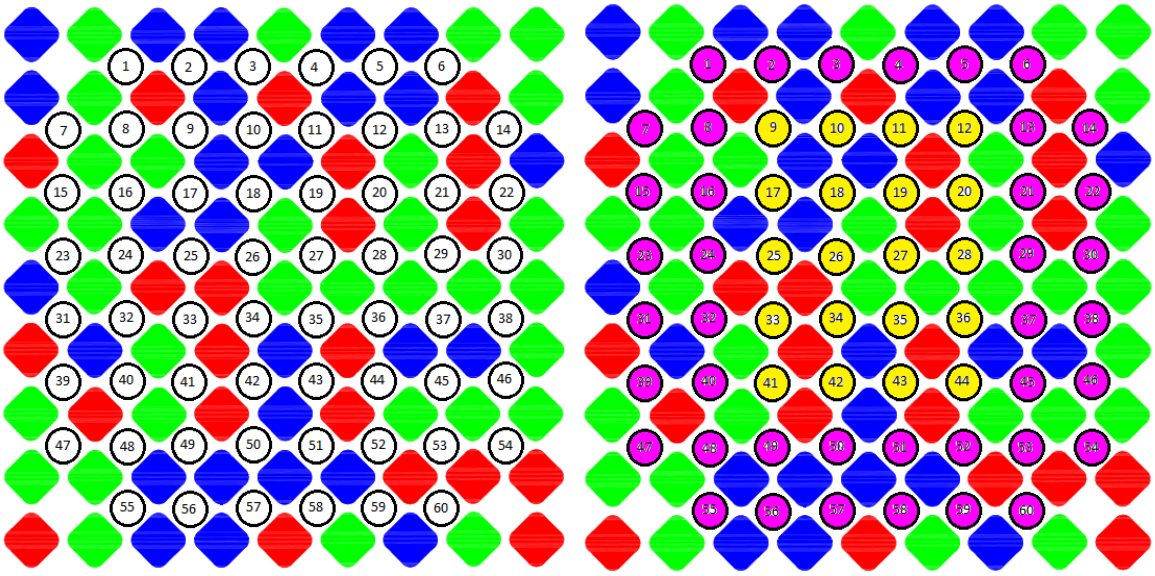


Figure 3.4: The digital bit pattern of the PRA fiducial; *left*: locations of the full 60 bits of data located on the marker; *right:* separation of Reed Solomon error correcting bits and CRC checksum bits

In the first stage of the encoding process, the twelve marker ID bits are encoding using a Cyclic Redundancy Checksum (CRC). This will add an additionally 8 bits to the marker leaving the first 12 bits unchanged. The next stage is to further encoding the marker such that larger error correction capabilities are obtained, providing added robustness to occlusion. Additions of 40 Reed Solomon (RS) error correcting bits are then added. The first twenty bits of the encoding process are indicated in figure 3.4 by yellow circles, while the additional 40 error correcting bits are indicated by pink circles. Important to note

is that in the actually marker implementation all bits are represented by the same pink color. The yellow circles are used here to emphasize the separation of the CRC encoded marker ID versus the redundant RS error correction bits.

3.2.1 Data Encoding Stage 1: Cyclic Redundancy Code

Like many encoding techniques in code theory, CRC is based on polynomial mathematics. More specifically it involves the computation of the remainder when one polynomial is divided by a second polynomial known as the generator. The idea of CRC is to divide a binary message by a large prime number before any data transmission occurs, then appending its remainder to the original message as a checksum. Upon receiving the message one would be able to use this same remainder value to ensure the message was not corrupted.

Both polynomials in the division calculation are included in what is known as a Galois Field (GF) with two elements, GF(2). A brief overview of Galois Fields can be found in Appendix C. A polynomial in GF(2) is a polynomial with a single variable, in this case, *x*. The coefficients of variables contained within the polynomial are either one of two values, a 0 or a 1. Both addition and subtraction within this field are done in modulo 2, which is more commonly known as the exclusive ‘OR’ operator. Multiplication on the other hand is computed as the logical ‘AND’ operator. Division of polynomials in GF(2) are compute in a very similar fashion to long division of integers. This is known as modulo 2 division of the polynomials.

The CRC codification can be summarized through the following steps:

1) *Choose a polynomial to be used as the generator*. This polynomial has width of *n*. Choosing a polynomial is not a straight forward task, it is common to choose polynomials that have been well documented and tested in literature. For an example the polynomial 0x97 = $x^8 + x^5 + x^3 + x^2 + x + 1$ for CRC-8 has been documented in [**57**] to work well with message of length 119 bits.

$$G(x) = x^8 + x^5 + x^3 + x^2 + x + 1 \tag{3.6}$$

2) *Multiple the message polynomial by* x^n. A binary message can be interpreted as the coefficients of a polynomial with variable *x.* In order to find the checksum, the message is first multiplied by a polynomial, x^n. This step is meant to append *n* zeros to the original message, *M(x)*, to ensure the reminder value is a whole number as appose to being a fraction.

$$M(x)\, x^n \tag{3.7}$$

3) *Compute the remainder*. The whole process of CRC encoding is to compute the remainder from the division between the message polynomial and the generator polynomial. Therefore a general formula for the computation of CRC checksums is as follows:

$$M(x)\, x^n = R(x)\, G(x) + C(x) \tag{3.8}$$

Here *M(x)* is the message to be encoded, *R(x)* is the quotient polynomial from the division operation, *G(x)* is the generator polynomial, and finally *C(x)* is the remainder. The division between original message and the generator sequence is performed. The checksum is then formed by taking

the coefficients from each of terms in the remainder polynomial with degree less than n. In this case the quotient is of no use.

4) *Append the remainder to the original message.* The original message must be appended with the remainder bit sequence, such that upon detection one can determine if errors have occurred.
5) *Decode modified message*. When the message needs to be read, the modified message is divided by the same generator polynomial. The way the math works out is, if no bits are in error the remainder will result in a value of zero. If the remainder is anything other than zero, at least one bit error has occurred.

For the purpose of this thesis a CRC checksum of 8 bits will be considered, which translates to the use of an 8th degree polynomial for the generator sequence. The example below and in figure 3.5 and 3.6 illustrates the encoding and decoding procedure of the message 1024 base 10, with the generator polynomial 0x97. Note: for simplification only the coefficients of the polynomials are used for the calculations.

Example CRC-8)

$$M(x) = 1024_{10} = 0100\ 0000\ 0000_2, \quad G(x) = 97_{16} = 1001\ 0111_2$$

Step 1: Multiple message by the degree of the generator polynomial

$$M(x)\ x^8 = 1024_{10}\ (256_{10}) = 262144_{10} = 0100\ 0000\ 0000\ 0000\ 0000_2$$

Step 2: Generate the new encoded message by carrying out the division of the $M(x)x^8$ and the generator polynomial (see figure 3.5).

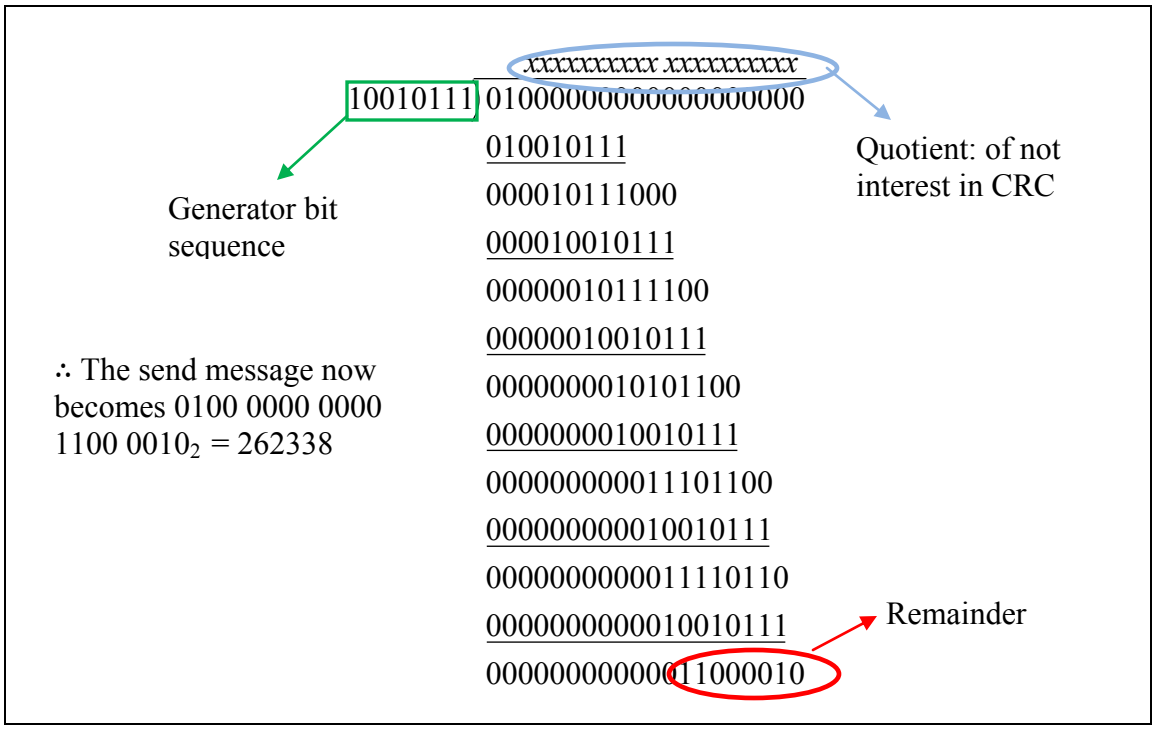


Figure 3.5: CRC encoding example

Step 3: Verify that the message was encoded properly by decoding the checksum (see figure 3.6)

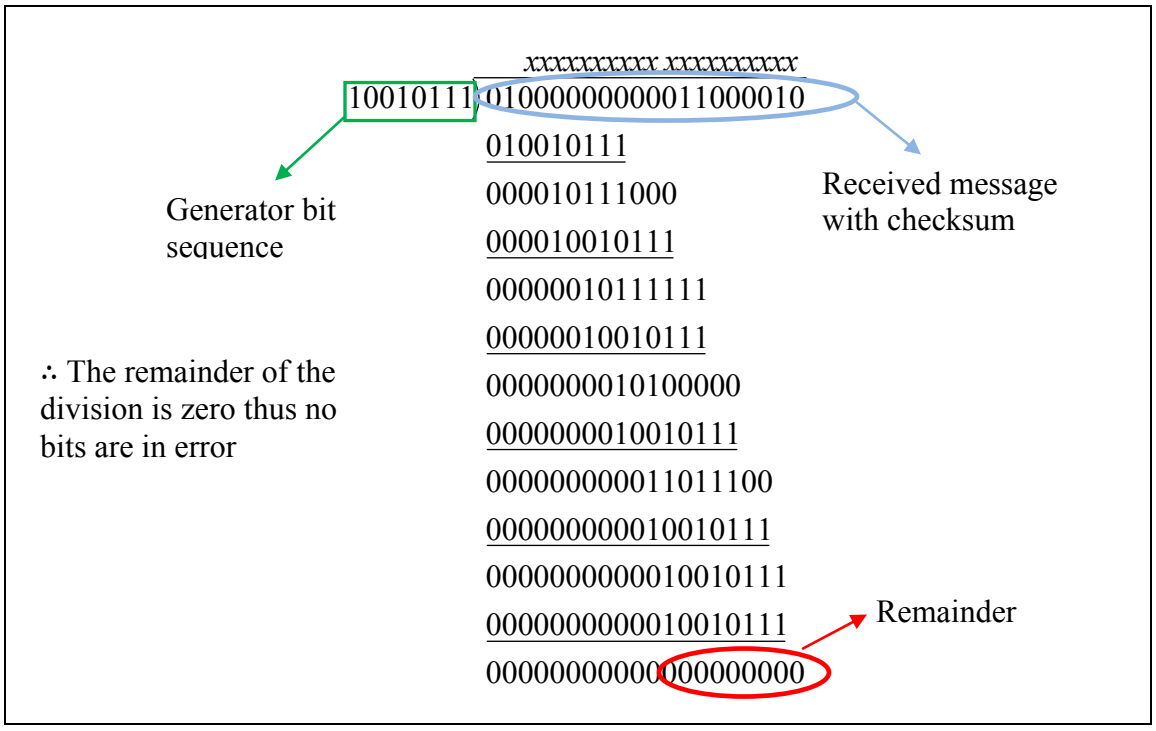


Figure 3.6: CRC decoding example

CRC is able is detect a certain amount of bit errors within the overall message sequence. However if the number of bit errors are too large then there is a finite probably that error will go undetected. The minimum number of bit errors that are require to avoid this issue is directly related to the Hamming Distance between codes. Hamming Distance (HD) is defined as the minimum number of bit inversions required to change one binary code to another. The larger the HD between CRC codes the more bit errors the code is able to detect. Altering the generation polynomial will affect the HD between code words thus increasing or decreasing the fiducials inter-marker confusion rate, as well as robustness to occlusion. Although CRC checksums are able to detect bit errors they are unable to correct any, which is where a second encoding technique for the fiducial is required. This second technique will wrap the 12 bit Marker ID as well as the 8 bit checksum adding a total 40 redundant bits to the marker. The second encoding method chosen for this fiducial marker is known as Reed Solomon error correction codes.

3.2.2 Data Encoding Stage 2: Reed Solomon Codes

The second data encoding scheme implemented in this fiducial marker is Reed Solomon codes. Reed Solomon (RS) codes are linear block-based codes, which split the message sequence into a number of data blocks. The encoding algorithm takes as an input the message consisting of *s* bits and split it up into *k* data symbols. Once the message has been encoded, new parity information gets added to make a codeword, *n* symbols in length. We can think of a RS codeword as being systematic because the original data is left unchanged and parity

symbols are only appended to the end of the message. The structure of RS codes is shown in figure 3.7.

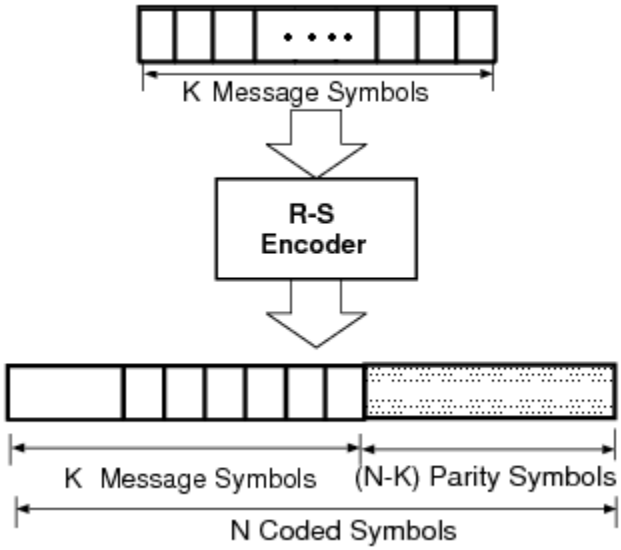


Figure 3.7: Reed Solomon code structure

Reed Solomon codes are defined mainly by 2 parameters (*n, k*). For all *m*-bit symbols *n* and *k* satisfy the following equation:

$$0 < \mathrm{k} < n < 2^{\mathrm{m}} + 2 \qquad (3.8)$$

where *k* is the number of data symbols that are to be encoded, and *n* is the total number of code symbols the block will have. For the purpose of this thesis $n = 2^m - 1$ and $k = n - 2t$, where *t* is the number of symbols that the code is able to correct. The number of parity symbols included in the codeword is defined by $n - k = 2t$.

RS codes are known to have the largest possible code minimum distance for any linear block code with the same input and output sizes [**58**]. The minimum distance is defined for non-binary sequences and is the number of symbols in

which the sequence differs (analogous to Hamming Distance). For RS codes minimum distance is define by equation 3.9, while the number of correctible symbol errors can be computed using 3.10 [**59**].

$$d_{min} = n - k + 1 \tag{3.9}$$

$$t = \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor = \left\lfloor \frac{n-k}{2} \right\rfloor \tag{3.10}$$

The advantage toward using RS codes as opposed to a simpler binary code can be seen in the following comparison as illustrated in [**58**]. A binary (*n, k*) code has an n-tuple space containing 2^n elements. Only 2^k of the total n-tuples are codewords. Now consider a non-binary (*n, k*) code, where symbols are composed m bits. The tuple space of this code is 2^{nm}, of which 2^{km} are codewords. In a direct comparison using parameters *n = 7, k= 3,* and *m = 3* the fraction of n-tuples that are codewords largely differ; 1/16 for the binary code and 1/4096 for the non-binary code. Thus non-binary codes can achieve large d_{min} values and in turn larger error correction capabilities.

For the proposed marker system we have the following RS parameters:

$$m = 4, n = 15, k = 5, and\ t = \left\lfloor \frac{n-k}{2} \right\rfloor = 5 \tag{3.11}$$

The 20 CRC encoded bits are split into symbols sizes of 4 and are encoded using RS(15,5) to produce an extra 40 parity bits that will be used to correct a maximum of 5 symbols.

Much like CRC code, RS code relies on Galois Fields (explained in Appendix C) and a generator polynomial, ***g(x)***. The Galois Field used for the proposed fiducial marker is GF(2^4) and its field element can be found in table 3.2

as derived in Appendix C. The degree of the generator polynomial used for a RS(*n, k*) code is equal to the number of parity symbols, 2*t*. Thus, there must be exactly 2*t* = 2(5) = 10 successive powers of β (elements of the Galois field) that are roots of the polynomial ***g(x)***. The generator polynomial can be written as in equation (3.12), where ξ indicated the first successive root index.

$$g(x) = \prod_{i=\xi}^{\xi+9} (x + \beta^i) \tag{3.12}$$

Field Elements for primitive polynomial $x^4 + x + 1$, GF(2^4)			
Index	Polynomial	Binary representation	
0	0	0000	
β^0	1	0001	
β^1	x	0010	
β^2	x^2	0100	
β^3	x^3	1000	
β^4	x + 1	0011	
β^5	x^2 + x	0110	
β^6	x^3 + x^2	1100	
β^7	x^3 + x + 1	1011	
β^8	x^2 + 1	0101	
β^9	x^3 + x	1010	
β^{10}	x^2 + x + 1	0111	
β^{11}	x^3 + x^2 + x	1110	
β^{12}	x^3 + x^2 + x + 1	1111	
β^{13}	x^3 + x^2 + 1	1101	
β^{14}	x^3 + 1	1001	

Table 3.2: Elements of the field GF(2^4)

The *k* = *5* symbols of the original message to be encoded can be represented as such:

$$\begin{aligned} M(x) &= m_{k-1}x^{k-1} + m_{k-2}x^{k-2} + \cdots + m_1x^1 + m_0 \\ &= m_4x^4 + m_3x^3 + m_2x^2 + m_1x^1 + m_0 \end{aligned} \tag{3.13}$$

where each of the coefficients m_i is a 4 bit message symbol.

The encoding procedure is very similar to that of CRC code. Equation 3.14 defines the relation between the message, generator polynomial, and remainder polynomial. Notice its similarity with equation 3.8. *M(x)* is first multiplied by x^{n-k} to shift the message *n-k* positions to the left, such that the parity bits can be appended to the message. Next, the shifted message $x^{n-k}\,M(x)$ is divided by the generator polynomial ***g(x)***.

$$x^{n-k}M(x) = R(x)g(x) + C(x) \tag{3.14}$$

where C(x) is the remainder and R(x) is the quotient. The remainder in this case consists of the parity error correcting symbols. Rearranging 3.14 yields:

$$C(x) = x^{n-k}M(x)\ modulo\ g(x) \tag{3.15}$$

And the resulting codeword is expressed in (3.16)

$$U(x) = C(x) + x^{n-k}M(x) \tag{3.16}$$

When a codeword is received, or detected in case of the fiducial marker, there is a certain amount of error present. We can represent this received polynomial by the transmitted codeword and an error polynomial (as in 3.17).

$$RX(x) = U(x) + e(x) \tag{3.17}$$

where the error polynomial has the following form:

$$e(x) = \sum_{n=0}^{14} e_n x^n \tag{3.18}$$

A key to the decoding process come from the fact that every valid codeword polynomial is a multiple of the generator polynomial, g(x). Consequently the roots of g(x) are also the roots of $U(x)$, and $RX(x)$ evaluated at each of the roots will

yield zero if and only if it is a valid codeword. This computation can be expressed through syndrome symbols, S_i.

$$S_i = RX(x)|_{x=\beta^i} = RX(\beta^i) \qquad for\ i = 1, \ldots., 10 \tag{3.19}$$

If all values of S_i are equal to zero, then it is safe to assume no error has occurred. Once it has been verified that an error has occurred, both the error value and error location must be determined in order to correct it. Let's define an error-locator polynomial, σ(x), for an arbitrary number of errors v.

$$\sigma(x) = (1 + \gamma_1 x)(1 + \gamma_2 x) \ldots (1 + \gamma_v x) = 1 + \sigma_1 x + \cdots + \sigma_v x^v \tag{3.20}$$

where the roots of σ(x) are $1/\gamma_1$, $1/\gamma_2$, …, $1/\gamma_v$. The reciprocals of these roots are the error-locations. A matrix, *A*, is then formed from the syndromes using an autoregressive technique detailed in [**60**].

$$A = \begin{bmatrix} S_1 & S_2 & \ldots & S_{t-1} & S_t \\ S_2 & S_3 & \ldots & S_t & S_{t+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ S_t & S_{t+1} & \ldots & S_{2t-2} & S_{2t-1} \end{bmatrix} \tag{3.21}$$

The values of σ_1 through σ_t are computed using the following equation:

$$\begin{bmatrix} S_1 & S_2 & \ldots & S_{t-1} & S_t \\ S_2 & S_3 & \ldots & S_t & S_{t+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ S_t & S_{t+1} & \ldots & S_{2t-2} & S_{2t-1} \end{bmatrix} \begin{bmatrix} \sigma_t \\ \sigma_{t-1} \\ \vdots \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -S_{t+1} \\ -S_{t+2} \\ \vdots \\ -S_{2t} \end{bmatrix} \tag{3.22}$$

Taking the inverse of the matrix A and rearranging equation 3.22 yields 3.23.

$$\begin{bmatrix} \sigma_t \\ \sigma_{t-1} \\ \vdots \\ \sigma_1 \end{bmatrix} = A^{-1} \begin{bmatrix} -S_{t+1} \\ -S_{t+2} \\ \vdots \\ -S_{2t} \end{bmatrix} \tag{3.23}$$

The polynomial σ(x) can then be constructed using values calculated from 3.23. The roots of the error locator are then determined by brute force, or an exhaustive search. That is, all 2^4 elements of the Galois Field are substituted into

the error-locator polynomial one by one. If any one of the substitutions yields a value of zero, then this element is a root. Inverses of the roots indicate the location of an error.

Now that the error locations are known, we need to determine what they are. This step involves using the syndromes and the error-locator polynomial root's to derive the error. The first step involves calculation of what is labeled the error evaluator polynomial, Ω(x). This is accomplished by convolving the syndromes, $S_i(x)$, with the error-locator polynomial, σ(x). We then determine Ω(x) for each error location and divide by the derivative of sigma. Each calculation will result in the corresponding bit that is in error for the specific symbol location. Correction occurs simply by inverting the calculated bit errors. The easiest way of doing so is to XOR the error symbol with the received symbols.

Reed Solomon coding is the last step in the marker design process. Markers are created by picking a number of marker ID values that range from 0 to 4095. The number of IDs chosen will depend on the application. The marker is then printed on a large 27 cm by 27 cm sheet of white paper with all CRC bits place in the bit positions specified by the yellow color circles of figure 3.4. All RS parity bits are place sequentially as produced by the RS codeword in the remaining bit positions of the marker, skipping any locations that have already been filled by the CRC bits. The key approach to this technique is that the original 12 bit marker ID is placed in the center of the marker while all parity bits surround it. This has been done considering the marker has been design to be used for indoor localization in environments with larger dynamic objects. Under

partial occlusion these larger objects will first occlude outer bit locations of the marker before corrupting the inner data pattern.

3.4 Chapter Summary

This chapter provided an overview of the design process for a robust Pseudo-Random Array fiducial marker. The first section in this chapter defines the proposed marker by 49 unique sub-arrays within a 9 by 9 grid of colored elements. It then introduces the complex data region responsible for distinguishing markers in the marker system. This data region was identified to house three main portions. The first beginning with a unique 12 bit marker identification number, the second an 8 bit error detection sequence, and the third, 40 error correction parity bits. The next section of the chapter introduced and explained the use of Cyclic Redundancy Checksums as a method of error detection. Lastly, Reed Solomon error corrections codes were detailed along with the markers ability to recover misread data.

Chapter 4. Marker Detection Algorithm

The purpose of this chapter is to fully explain the proposed marker detection algorithm. First a few assumptions for the system are introduced. Color blob analysis and classification are specified, followed by feature point detection and filtering. The calculation of 3D pose relative to the maker is also defined, allowing for global position discovery of a mobile platform. Lastly, data sampling techniques are mentioned, leading to the decoding and identification of a valid marker ID.

4.1 Algorithm Description

The detection process involved for the proposed fiducial marker differs slight from the conventional approaches discussed in Chapter 2. Firstly, the feature points of this marker are slightly more complex then the black border quadrilateral or the black border circle seen in ARTag and Intersense, respectively. Secondly, there are a greater number of feature points in the designed marker. Thirdly, the marker design takes advantage of color images which translates to processing three images: red, blue and green, as opposite to one grayscale. Lastly, the detailed approach will take advantage of 3 dimensional data (if available), which is fairly uncommon for planar fiducials. A few key assumptions for this detection algorithm are as follows:

- The camera(s) is/are calibrated, with all calibration parameters known.

- Each image of the scene has a corresponding rectified depth image that is available. This can be obtained through use of a stereo setup, or as in the case of this thesis, a 3D sensor such as the Xbox Kinect [**61**]. (this is not absolutely necessary, however, it greatly improves pose estimation)
- Markers are mounted in the upright position. That is to say, the marker is relatively vertical.
- All implemented cameras support color image outputs.
- The lighting conditions in the working environment do not drastically change, ie. controlled lighting environment.

The marker detection algorithm under evaluation can be split up into the following steps:

1) *Adaptive Thresholding and Blob Analysis*: Received RGB images from the camera are combined to a grayscale image and then undergo adaptive thresholding resulting in a binary image of contours. Closed contours are then filtered based on a number of predefined parameters.

2) *Color Classification*: In parallel to the adaptive thresholding procedure, a copy of the input color image is converted to the HSV color space. At the end of color classification each closed contour is classified as being one of 3 colors: red, blue or green.

3) *Feature Point Detection:* Based on the results of blob and color analysis, blobs undergo tests for spatial distribution and color configuration. These tests determine whether or not clusters of 9 elements match any one of the 49 predefined feature points of the marker.

4) *Pose Estimation:* Once at least 5 feature points have been found in the image, the fiducial is assumed to be present within the scene. A correspondence between each element of detected feature points in the image plane and the ideal marker plane are used to determine the extrinsic parameters of the camera with respect to the marker.

5) *Data Sampling and Decoding:* Using the pose of the camera, ideal data bit locations are re-projected to the image plane of the camera and sampled to determine bit values. The message symbols are then feed through a RS and CRC decoder to determine the Marker ID number.

6) *Marker Tracking:* Upon successful detection of the marker, tracking is initialized. This is implemented in order reduce the amount image processing, by limited the region of the image to be analyzed.

4.1.1 Adaptive Thresholding and Blob Analysis

Images are received from the camera(s) in the common RGB format. This means that an image is composed of three channels, one for each of the 3 primary colors red, blue and green. We can think of an image as being three

separate two dimensional arrays or matrices with elements representing light intensity values from 0 to 255. For every capture image, a copy is made and depth values of each pixel are calculated, or in the case of 3D sensors, are converted to metric units. Next contour analysis begins. This involves applying local adaptive thresholding on the image followed by filtering the resulting edges.

The copied image is first convert to a single channel grayscale image. This is accomplished by the following weighted addition formula:

$$grey(x,y) = 0.299 \cdot R(x,y) + 0.587 \cdot G(x,y) + 0.114 \cdot B(x,y) \quad (4.1)$$

where $grey(x,y)$ represent the intensity value of the element in the x^{th} row and y^{th} column of the grayscale image. Similarly $R(x,y)$ represents the corresponding element in the red channel image, $G(x,y)$ in the green channel image, and $B(x,y)$ in the blue channel image. The weights assigned to each channel are related to percentage of blue, red and green pixels in the Bayer filter used in image capturing.

The local adaptive thresholding technique chosen for this work is known as Local Mean C Adaptive Thresholding [**27**]. It operates on the grayscale image where $grey(x,y) \in [0, 255]$. Adaptive thresholding can be obtained using the following formula:

$$edges(x,y) = \begin{cases} 0, & if\ grey(x,y) \leq t(x,y) \\ 255, & otherwise \end{cases} \quad (4.2)$$

where $t(x,y)$ is the threshold value computed by using the mean pixel values in a $s \times s$ window centered about the pixel location at (x, y) and then subtracting a value C. Where C is used as a fine tuning parameter for the resulting image.

$$t(x,y) = mean(x,y) - C \quad (4.3)$$

$$mean(x,y) = \frac{1}{s^2} \sum_{i=x-\lfloor s/2 \rfloor}^{x+\lfloor s/2 \rfloor} \sum_{j=y-\lfloor s/2 \rfloor}^{y+\lfloor s/2 \rfloor} grey(i,j) \tag{4.4}$$

The mean of all pixels in the $s \times s$ window is conventional expressed as in 4.4 where window sizes are always odd. However, a slight enhancement to this mean calculation was implemented as detailed by Shafait *et al.* in [**62**]. An integral image can be used to more efficiently calculate the mean of the window. Intensity values of pixels in integral images are equal to the sum of all pixels intensities to the left and above that position in the image (as expressed in equation 4.5).

$$I(x,y) = \sum_{i=0}^{x} \sum_{j=0}^{y} grey(i,j) \tag{4.5}$$

The mean value can then be simply expressed as:

$$mean(x,y) = \frac{\left(I\left(x+\frac{s}{2},y+\frac{s}{2}\right)+I\left(x-\frac{s}{2},y-\frac{s}{2}\right)-I\left(x+\frac{s}{2},y-\frac{s}{2}\right)+I\left(x-\frac{s}{2},y+\frac{s}{2}\right)\right)}{s^2} \tag{4.6}$$

Figure 4.1 (a) contains a sample image illustrating the results of the adaptive thresholding technique described above.

In parallel to the process of computing the adaptive thresholding, a separate copy of the captured image is converted to the Hue, Saturation and Value color space. The conversion is calculated using the following formulas:

$$R_n(x,y) = \frac{R(x,y)}{255}, G_n(x,y) = \frac{G(x,y)}{255}, B_n(x,y) = \frac{B(x,y)}{255} \tag{4.7}$$

$$\hat{V}(x,y) = \max\left(R_n(x,y), G_n(x,y), B_n(x,y)\right) \tag{4.8}$$

$$\hat{S}(x,y) = \begin{cases} \dfrac{\hat{V}(x,y) - \min\left(R_n(x,y), G_n(x,y), B_n(x,y)\right)}{\hat{V}(x,y)}, & if\ V \neq 0 \\ 0, & otherwise \end{cases} \tag{4.9}$$

$$\hat{H}(x,y) = \begin{cases} \frac{60\left(G_n(x,y) - B_n(x,y)\right)}{\hat{V}(x,y) - \min\left(R_n(x,y), G_n(x,y), B_n(x,y)\right)}, & if\ \hat{V}(x,y) = R_n(x,y) \\ 120 + \frac{60\left(B_n(x,y) - R_n(x,y)\right)}{\hat{V}(x,y) - \min\left(R_n(x,y), G_n(x,y), B_n(x,y)\right)}, & if\ \hat{V}(x,y) = G_n(x,y) \\ 240 + \frac{60\left(R_n(x,y) - G_n(x,y)\right)}{\hat{V}(x,y) - \min\left(R_n(x,y), G_n(x,y), B_n(x,y)\right)}, & if\ \hat{V}(x,y) = B_n(x,y) \end{cases} \quad (4.10)$$

$$H(x,y) = \begin{cases} \frac{255(\hat{H}(x,y) + 360)}{360}, & if\ \hat{H}(x,y) < 0 \\ 255, & if\ \hat{H}(x,y) > 360 \\ \frac{255\left(\hat{H}(x,y)\right)}{360}, & otherwise \end{cases} \quad (4.11)$$

$$V(x,y) = \begin{cases} 0, & if\ \hat{V}(x,y) < 0 \\ 255, & if\ \hat{V}(x,y) > 1 \\ 255\left(\hat{V}(x,y)\right), & otherwise \end{cases} \quad (4.12)$$

$$S(x,y) = \begin{cases} 0, & if\ \hat{S}(x,y) < 0 \\ 255, & if\ \hat{S}(x,y) > 1 \\ 255\left(\hat{S}(x,y)\right), & otherwise \end{cases} \quad (4.13)$$

Succeeding thresholding, the resulting binary image will contain a number of edges, which are expressed by the white pixels in figure 4.1 (a). These edges represent contours that define sharp changes in image contrast. Thick over-defined edges are converted to contours with thickness of one pixel (see figure 4.2 (b)). Any contour that closes upon itself is denoted as a ‘blob’. The binary image is then parsed for all blobs, such that non-closed contours are eliminated as well as blobs that don’t fit various parameter requirements (figure 4.1 (c)). These parameter requirements include:

- Perimeter: the number of pixels that make up the closed contour
- Area: the number of pixels contained within the contour
- Elongation: a ratio of longest and smallest axis’ of the blob

- Compactness: a function of how circular a blob is
- Valid Depth: a measure of the invalid depth pixels within the contour

While blobs are being filtered out, the average depth value and center of mass points of valid contours are calculated and recorded for later use. At the end of blob analysis we have a list of blobs, their corresponding center points and average depth values. However, before feature point detection can begin, the color value of each blob must be obtained. Rather, blobs will be placed into one of four color categories as defined by color classification.

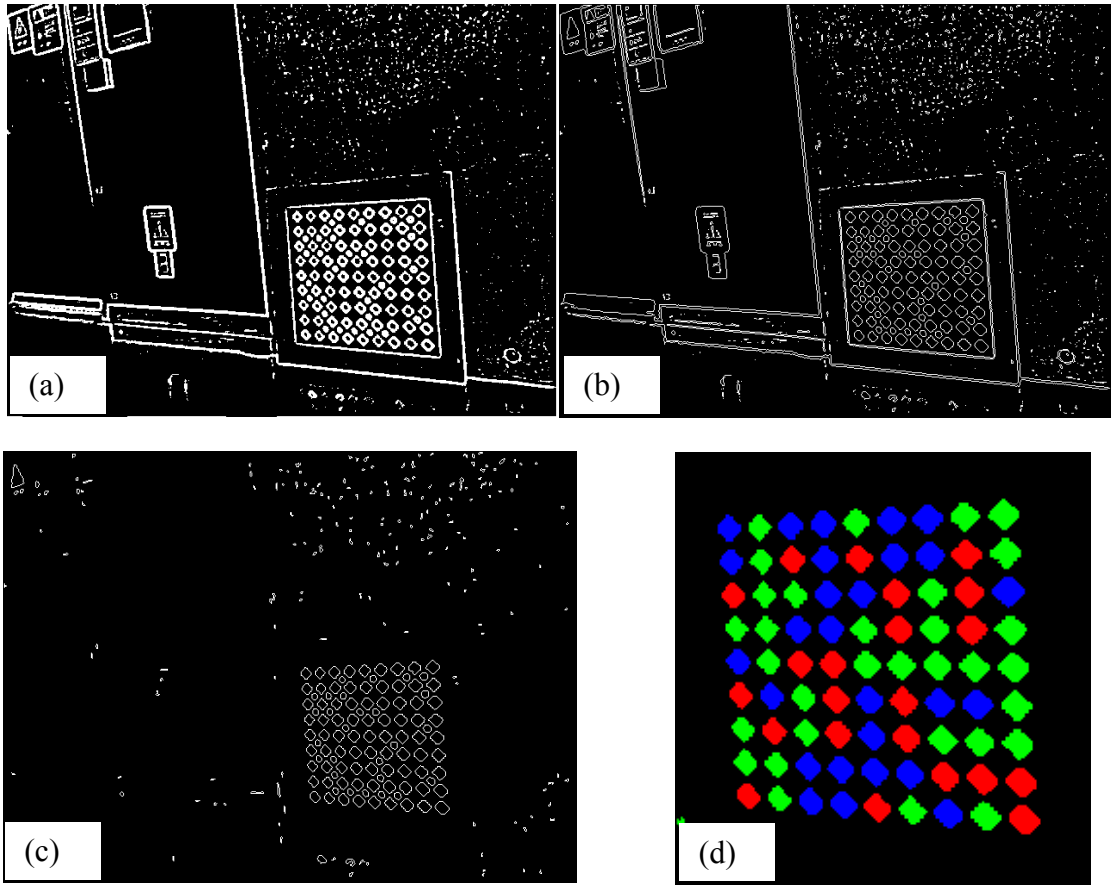


Figure 4.1: Contour detection. (a) a binary image of resulting edges (b) all edges converted to well defined contours (c) image depicting all blobs after filtering (d) resulting marker elements after color classification

4.1.2 Color Classification

The color classification procedure is not necessarily straight forward since it is a multi-step process. The hue color space of the HSV converted image is first split into four regions, each region representing one of the colors red, blue, green or 'other'. The color group label 'other' defines any color value that does not fit the previous three color groups. Figure 3.9 shows the H color space and how regions can be split up by choosing a few broad threshold values. Since the remaining two channels of the HSV color space are ignored this will give a fair amount of robustness to various lighting conditions. For every detected blob, each of the containing pixels is then defined as belonging to one of the four predefined regions using equation 4.14. The color of the blob is then defined by region that contained the largest number of pixels, while removing any blobs defined by the region labeled as 'other'. This is repeated iteratively until all blobs have been parsed.

$$p_{color}(x,y) = \begin{cases} Green, & if\ g_l < H(x,y) < g_u \\ Blue, & if\ g_u < H(x,y) < b_u \\ Red, & if\ H(x,y) < r_l\ or\ H(x,y) > r_u \\ other, & otherwise \end{cases} \quad (4.14)$$

where $H(x,y)$ defines the hue value of a pixel (x,y), $p_{color}(x,y)$ is the resulting region of the pixel (x,y), and the variables g_l, g_u, b_u, r_u, r_l define threshold values in the Hue color space. The threshold values are chosen as illustrated in figure 4.2.

To further distinguish the color of each blob, a k-means clustering method is used. That is, each of the blobs will be split into 3 clusters based on the Euclidean distance between their average hue and saturation values. The initial

guess as to which cluster each blob belongs to will be identified from the results of the above thresholding technique. A sample plot of the H and S values in 2D space can be seen in figure 4.3.

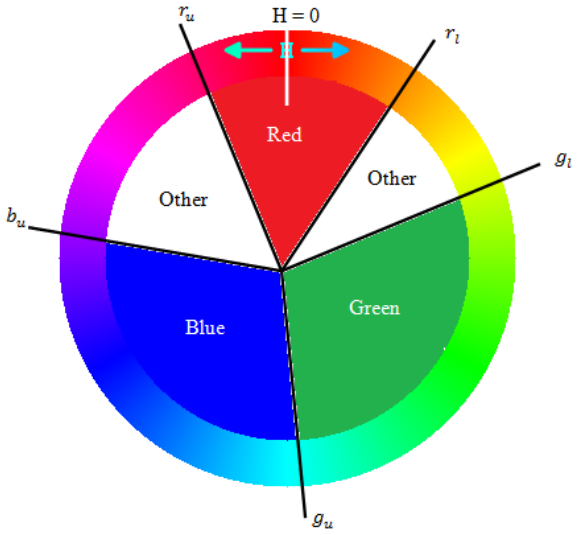


Figure 4.2: Region splitting of the Hue color space

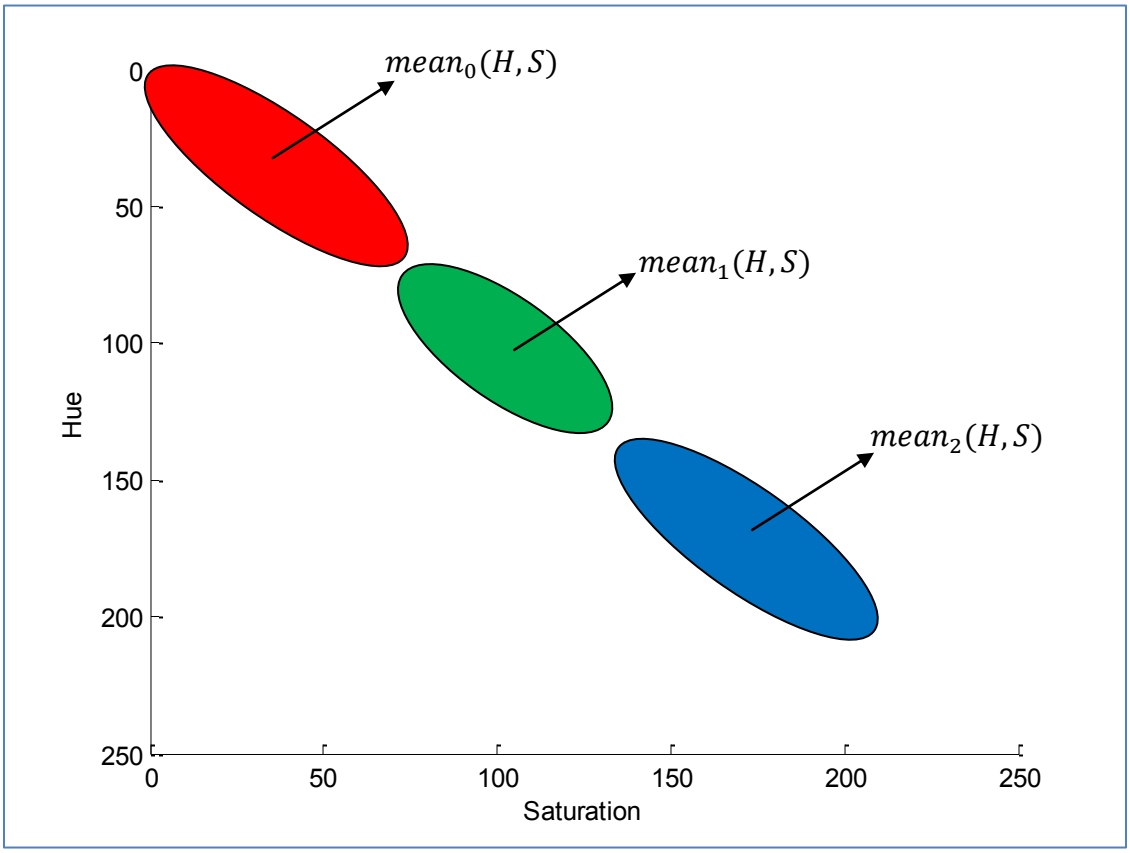


Figure 4.3: 2D plot of average hue and saturation values for detected blobs.

The implemented k-means clustering algorithm can be explained in the following way:

Step 1: Calculate the mean (H, S) vector for each of the three clusters, C_i where $i = 0, 1, 2$. Let x_j define an element of the cluster C_i.

$$m_{H,i} = \frac{1}{sizeof(C_i)} \sum_{x_j \in C_i} Avg\_H_j \tag{4.15}$$

$$m_{S,i} = \frac{1}{sizeof(C_i)} \sum_{x_j \in C_i} Avg\_S_j \tag{4.16}$$

$$mean_i(H,S) = \{m_{H,i} \quad m_{S,i}\} \tag{4.17}$$

where Avg_H_j, Avg_S_j indicate the average hue and saturation of pixels in the j^{th} blob of the i^{th} cluster.

Step 2: Assign each of the blobs to the cluster with the closest mean, $mean_i(H,S)$. This is done by calculating the Euclidean distance between each blob's hue/saturation values and the cluster means. Blobs then redefine their color according to the cluster number with the closest mean value. Euclidean distance from a blob, x_j, to a cluster, C_i, is calculated in the following manner:

$$d_{j,i}(H,S) = \sqrt{(Avg\_H_j - m_{H,i})^2 + (Avg\_S_j - m_{S,i})^2} \tag{4.18}$$

Step 3: Calculate the mean value of the newly created clusters using the formula of step 1.

This algorithm repeats, only to stop if after step 2 the blobs have not changed clusters. However, a maximum iteration count has been implemented since this technique is used only as a refinement of the initial guess.

4.1.3 Feature Point Detection and K-Nearest Neighbors

The next phase of marker detection determines whether a marker is present in the captured scene. Each of the 49 feature points of the marker have distinct spatial and color configurations making them unique and unlikely to found in random objects within the scene. This is illustrated in figure 4.4. A feature point can be uniquely defined by 9 elements, a center blob and 8 surrounding blobs. The surrounding blobs are uniformly distributed in 360 degrees around the center blob. Feature point detection exploits these qualities for simply and reliable marker detection.

The current algorithm uses an iterative minimum distance clustering for each detected blob. Colored blobs are clustered into groups of 9 elements based on relative position in an image. This clustering can be performed in a very similar manner to the k-means clustering used for color classification. The major differences are: that only one 9 element clusters is created at each iteration and the procedure is repeated for all detected blobs in the image. This search method is denoted as Approximate Nearest Neighbors (ANN) and uses multiple k-dimensional tree data structures.

A k-dimensional tree, or kd-tree for short, is a data structure used in the partitioning of points within a k-dimensional space. Kd-trees are very similar to binary search trees however every node in a kd-tree has k dimensions. At the highest level of the tree (the root) data is split into two halves by a hyperplane. A hyperplane is denoted as the generalization of a plane but in an n-dimensional space. This hyperplane is orthogonal to a defined dimension and located at a

specific threshold value. Each half of the partitioned space is then split recursively into another two halves following a similar manner. It is common to have this split occur in the dimension of greatest variance, with the threshold value equal to the mean value of all points in that dimension. However for this implementation the threshold values are chosen random between the maximum and minimum values of the dimension with greatest variance. All points that have coordinate values lower than the selected threshold will be created as nodes in a sub tree located to the left of the root. Points above the threshold will be located in the right sub tree of the root. When there are no points in either partition of the space, the node that created the hyperplane is labeled a 'leaf'. Leaves of a kd-tree form a complete partition of the space under analysis. Important to note is that each node in the tree corresponds to a cell in the k-dimension space.

To find the nearest neighbor, the tree is searched using a best bin first technique seen in [**63**]. This search makes use of a priority queue. Starting at the root of the tree, the squared Euclidean distance between the query point and each of the branching nodes is evaluated. The branch with the greatest distance is added to the top of the queue while the other branch is used to continue searching for the nearest neighbor. This process is repeated recursively until a leaf node is reached, in which case the search continues by removing the top entry in the queue and resuming in that branch. In order to limit the amount of time spend searching for the nearest neighbor is often useful to set a cap on the maximum amount of nodes traversed. In this case an exact nearest neighbor might not be found but rather the best one for the pre-allocated time frame. To

further improve this search method, multiple randomized kd-trees have been implemented and searched in parallel as detailed by Muja and Lowe in [**64**]. Figure 4.5 indicates feature point detection using 4 randomize kd-trees and an approximate nearest neighbor search of 9 elements.

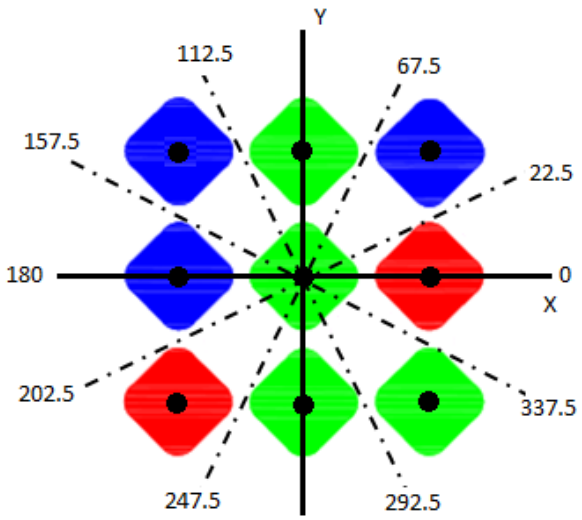


Figure 4.4: Spatial distribution of unique feature points contained within the marker. The center point of each element lies within a predetermined range of angles with respect to the x-axis.

After the kd-trees are created using all detected blobs, the search for features points begins. For every detected blob an approximate of the 8 nearest neighbours are located based on their center of mass points. Then these 9 elements are examined and compared to existing feature points. First a validation of the spatial distributing is performed. The 8 neighboring blobs must be uniformly distributed in 360 degrees around the query point. If this condition is not satisfied the search continues with the next blob. Otherwise the procedure persists with a color comparision. Based on the blobs locations in the spatial

configuration of figure 4.4, an ordered 9 element vector can be created. At each index of the vector a color values are stored, where red is represented by 1, green by 2, and blue by 3. This color vector is then compared with the color vector list of the 49 pre-defined feature points of the marker. A feature point is said to be identified if a match between color vectors exits. Once a match occurs, a correspondence between the center points of each of the 9 elements in the image and the marker plane is stored for later use. This corresponds is 3D to 3D when using depth sensors, or stereoscopic cameras, and 2D to 3D when only one camera is implemented. The search for feature points continues until a minimum of 5 are located, in which case the marker can be clearly identified.

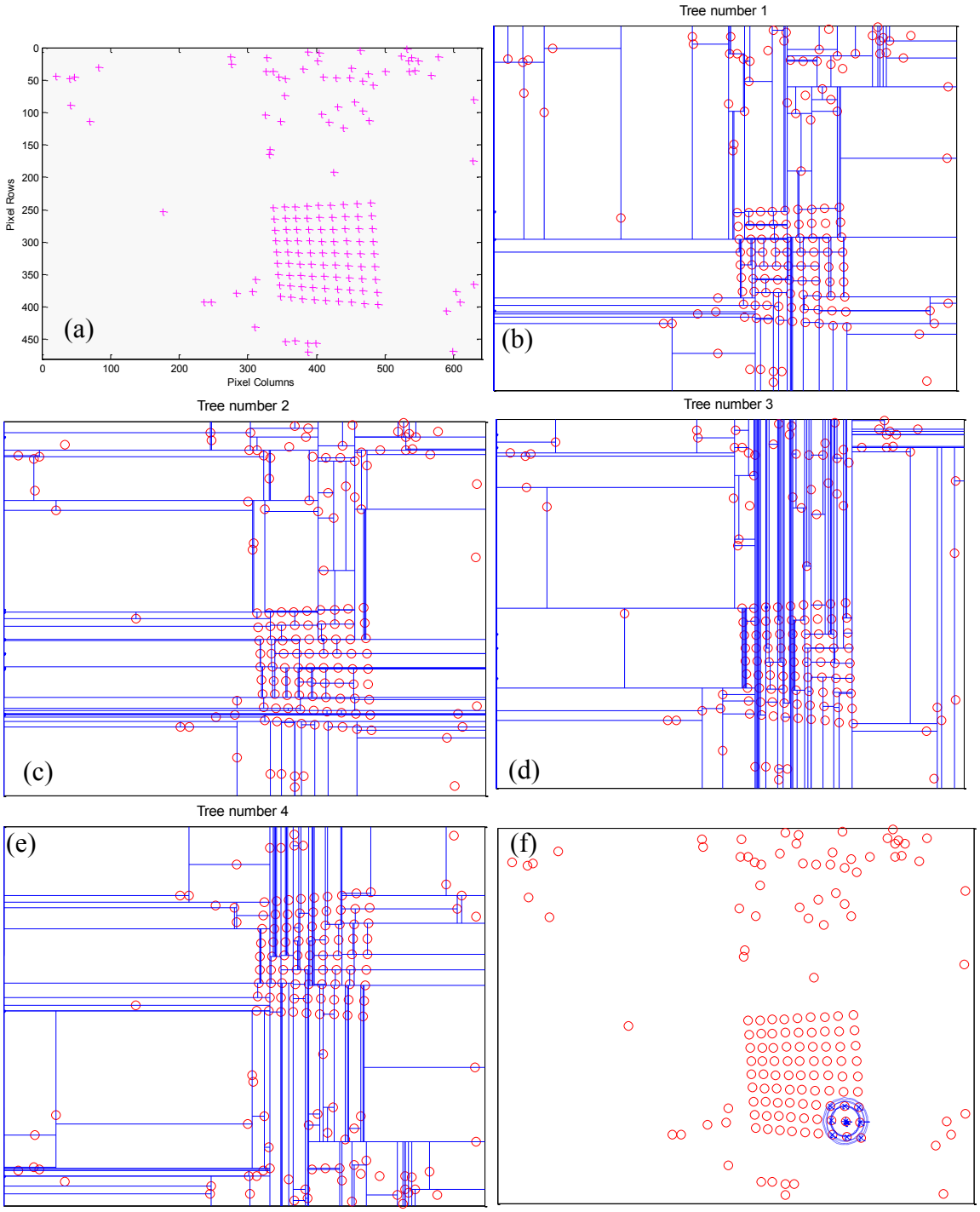


Figure 4.5: Sample of randomized kd-trees and nearest neighbor search. (a) a plot of the original data set (b)-(e) the first four randomize search trees used (f) the resulting 9 nearest neighbor search.

4.1.4 Pose Estimation and Rigid Transform Estimation

Feature point detection not only identifies whether a marker is present in the scene, it also determines a 3D to 3D correspondence between the reference frame of the camera and that of the marker. This correspondence can be used to identify the pose of the camera with respect to the marker, which is extremely useful for the target application of indoor localization. When five feature points are identified this yields a maximum of 45 corresponding key-points, one for each of the 9 elements within a feature. The minimum number of points utilized in pose estimation is 19 and this occurs with all 5 detected feature points are adjacent to each other, as explained in section 3.3.2. There are many approaches toward finding the rotation and translation between reference frames. The chosen one is based on the work of Arun, Huang and Blostein in [**65**]. Rotation and translation parameters are estimated using a least squares solution based on the singular value decomposition of a 3 x 3 matrix. Note that although the marker system supports a monocular camera implementation, only the 3D to 3D correspondence will be considered for this section of pose estimation. This has been done because a 3D sensor was used for most of the work completed in this thesis.

The feature point detection algorithm yields two points sets:

$$\{\eta_i\}\ and\ \{p_i^{'}\}\ for\ i = 1, 2, \dots, N \tag{4.19}$$

where η_i indicates a 2 dimensional point $(x_{image}, y_{image})^T$ in the image plane of the optical device, and $p_i^{'}$ a 3 dimensional point $(x_{marker}, y_{marker}, z_{marker})^T$ in the reference frame of the marker. These point sets are obtained from the elements

of detected features in the image plane of the camera and the plane of the fiducial marker, respectively. The value of z_{marker} is always equal to zero because marker elements are defined on a planar surface, the x-y axis of the reference frame. The variable N is used to define the number of corresponding key-points found in the feature detection process.

The next step involves transforming the point set $\{\eta_i\}$ into metric units. This is accomplished by using the intrinsic matrix of the camera. A new point set $\{p_i\}$ is obtained by using the following equations:

$$x = \frac{x_{image} - c_x}{f_x}, \qquad y = \frac{y_{image} - c_y}{f_y} \tag{4.20}$$

$$x_m = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 xy + p_2(r^2 + 2x^2) \tag{4.21}$$

$$y_m = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_2 xy + p_1(r^2 + 2y^2) \tag{4.22}$$

$$r^2 = x^2 + y^2 \tag{4.23}$$

$$p_i = \left(x_{m,i}, y_{m,i}, z_{m,i}\right)^T \tag{4.24}$$

where k_1, k_2, k_3, p_1, p_2 are distortion coefficients from the camera model and $c_x, f_{x,} c_y, f_y$ are camera parameters from the intrinsic matrix. These parameters should be already known from camera calibration. x_m, y_m, z_m are the x, y, and depth values obtained from the elements within the detected features points (in metric units). The third data set can be constructed as:

$$\{p_i\} \, for \; i = 1, 2, \ldots, N \tag{4.25}$$

Points from the cameras reference frame can be related to the markers reference frame by the following formula:

$$p_i^{'} = Rp_i + T + N_i \tag{4.26}$$

R represents a 3 x 3 rotation matrix, T is a 3 x 1 translation vector and N_i is a noise vector. The noise vector is used here to represent the error present in the given correspondence.

The below algorithm will act to minimize equation 4.27 and obtain a valid estimation of the rotation and translation between the camera and the marker.

$$\Sigma^2 = \sum_{i=1}^{N} ||p_i^{'} - (Rp_i + T)||^2 \tag{4.27}$$

From [**66**] is clear that $\{p_i^{'}\}$ and $\{p_i^{''} \triangleq \hat{R}p_i + \hat{T}\}$ have the same centroid when $\hat{R}$ and $\hat{T}$ are the least squared solution to equation 4.27. In other words the average of all points $p_i^{'}$ is equal to $p_i^{''} \triangleq \hat{R}p_i + \hat{T}$. Thus the first step in determine the extrinsic parameters is to calculate the mean values of $p_i^{''} and\, p_i^{'}$, $p^{''} and\, p^{'}$ respectively.

$$p^{'} \triangleq \frac{1}{N}\sum_{i=1}^{N} p_i^{'} \tag{4.28}$$

$$p^{''} \triangleq \frac{1}{N}\sum_{i=1}^{N} p_i^{''} = \hat{R}p_i + \hat{T} \tag{4.29}$$

$$p \triangleq \frac{1}{N}\sum_{i=1}^{N} p_i \tag{4.30}$$

Now if we let q_i = p_i - p and $q_i^{'}$ = $p_i^{'}$ - $p^{'}$ equation 4.27 can be redefined to:

$$\Sigma^2 = \sum_{i=1}^{N} ||q_i^{'} - (Rq_i)||^2 \tag{4.31}$$

This means finding $\hat{R}$ is as easy as minimizing equation 4.31 by using singular value decomposition (SVD). The next step involves the calculation of a matrix H that will be used in the SVD.

$$H = \sum_{i=1}^{N} q_i \, q_i^{'t} \tag{4.32}$$

The 3 x 3 matrix H serves as an input to a SVD algorithm which yields the following:

$$H = U \Lambda V^t \tag{4.33}$$

The estimated rotation matrix can then be calculated in the following manner:

$$X = V U^t \tag{4.34}$$

$$\hat{R} = \begin{cases} X, & if \det(X) > 0 \\ no\ solution, & if \det(X) < 0 \ and \ \lambda_l \neq 0 \ for \ l = 1,2,3 \\ V^{'} U^t, & otherwise \end{cases} \tag{4.35}$$

The case of no solution is rare and has not occurred in the thousands of tests performed for marker validation. A reason for this is because all points used to determine the transformation R are coplanar (lie of the same plane in 3D space). To calculate the translation between the marker and the camera, equation 4.36 is used.

$$\hat{T} = p^{'} - \hat{R} p \tag{4.36}$$

Therefore, using the corresponding image points and marker points from feature point detection, the pose of the camera with respect to the marker can be calculated with a least square minimization approach. Following pose estimation the inner data pattern of the marker is sampled.

4.1.5 Data Sampling and Decoding

Data sampling is fairly straight forward once the camera's pose is obtained. It involves a projection of the data bit positions from the ideal marker of figure 3.4 to the image plane of the camera. Equations 4.37 – 4.41 show the calculations needed to be performed in order to compute the projection of 3D data points to the camera's image plane, given the intrinsic and extrinsic parameters of the camera. The extrinsic parameters for these calculations are derived from the camera pose with respect to the marker.

$$\begin{bmatrix} x_{ip}{}' \\ y_{ip}{}' \\ w \end{bmatrix} = \hat{R}^t \left(\begin{bmatrix} x_{data} \\ y_{data} \\ z_{data} \end{bmatrix} - \hat{T} \right), x_{ip}{}'' = \frac{x_{ip}{}'}{w}, y_{ip}{}'' = \frac{y_{ip}{}'}{w} \tag{4.37}$$

$$x_{ip}{}''' = x_{ip}{}''(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x_{ip}{}'' y_{ip}{}'' + p_2 \left(r^2 + 2x_{ip}{}''^2 \right) \tag{4.38}$$

$$y_{ip}{}''' = y_{ip}{}''(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_2 x_{ip}{}'' y_{ip}{}'' + p_1 \left(r^2 + 2y_{ip}{}''^2 \right) \tag{4.39}$$

$$r^2 = x_{ip}{}''^2 + y_{ip}{}''^2 \tag{4.40}$$

$$\begin{bmatrix} x_{ip} \\ y_{ip} \end{bmatrix} = \begin{bmatrix} f_x x_{ip}{}''' + c_x \\ f_y y_{ip}{}''' + c_y \end{bmatrix} \tag{4.41}$$

where $[x_{ip}, y_{ip}]$ are pixel coordinates and $x_{data}, y_{data}, z_{data}$ are marker coordinates representing locations of data points. Note that z_{data} = 0 since the marker has no depth to it.

Each circular bit position of the ideal marker contains 5 uniformly distributed sampling locations. A total of 300 (60 * 5) sample points are passed through equations 4.37 – 4.41 yielding an equivalent amount of pixel coordinates. The intensity values of the five pixels corresponding to the 5 samples points per bit position are averaged. If the average value is below a pre-defined threshold,

that bit is assumed to be ‘1’ (digital ‘HIGH’) or otherwise ‘0’ (digital ‘low’). When all data bits are determine the RS codeword can be constructed. From figure 3.4, bit positions 25 – 28 are placed in the far left locations of the codeword, followed by bits 33 – 36, 41-44, 9-12, 17-20, 1-8, 13-16, 21-24, 29-32, 37-40, and 45-60. The created codeword is first inserted into the RS decoder correcting up to 5 symbols errors. The three possible outcomes from the decoder are:

Case 1) No Errors: No errors were detected and the 20 CRC bits are passed to the CRC decoder

Case 2) Less than 5 symbol error: Errors were detected but successfully corrected. The 20 CRC bit a passed to the next decoding stage

Case 3) More than 5 symbol errors and decoder fails to properly correct: This case actually has two possibilities. The first is when the decoder attempts to correct errors and fails but thinks it succeeded. With this possibility, the 20 CRC bits are passed to the next decoder where it will attempt to catch the error. In the second option, the decoder realizes that there are still errors after attempted correction, in which case the algorithm is halted and no marker is detected.

If the RS decoder is successful, the top most 20 bits of the resulting codeword are passed to the CRC decoder where a simple binary division is performed. Marker detection is considered successful when the result of this division yields zero. In this case, the marker identification number is taken as the top most 12 bits of the CRC codeword. Otherwise marker detection fails and no

marker are present in the scene. The decoding procedure is summarized by the flow chart of figure 4.6.

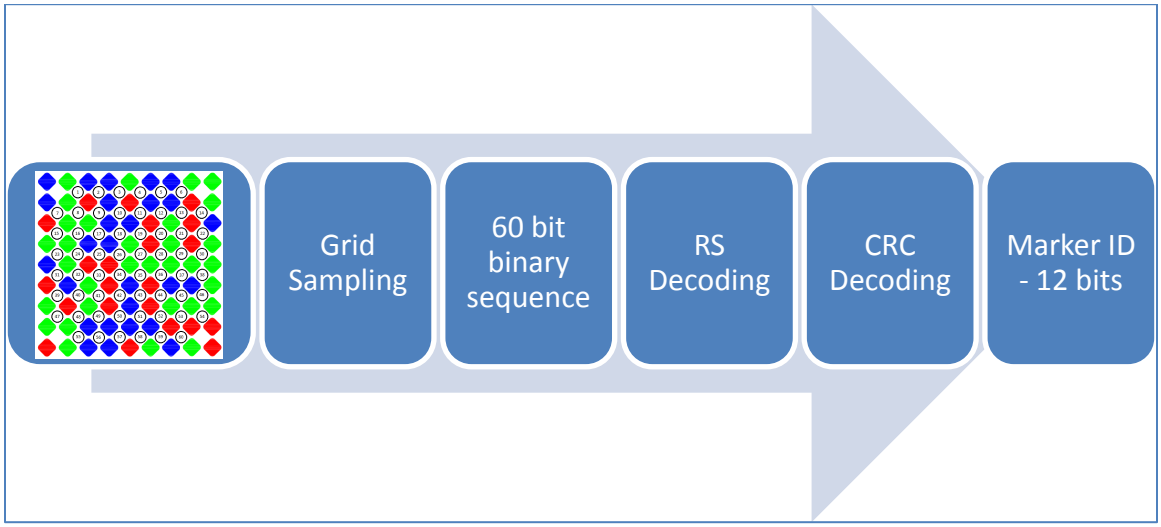


Figure 4.6: Marker decoding flow chart

4.1.5 Marker Tracking

The last step in marker detection is marker tracking. This only occurs when a marker was detected in the previous frame. In this situation, instead of performing image analysis on the whole image, only a section of the image is processed. The image sub section that is processed corresponds to one and half times the pixel area of the marker, which happens to be located at the center point of the previously detected marker. When a marker ID is detected in one frame but not in the following frame, the detection algorithm reverts back to processing the whole image. Figure 4.7 illustrates a sequence of images where the region of the picture that is being processed is identified by a colored square.

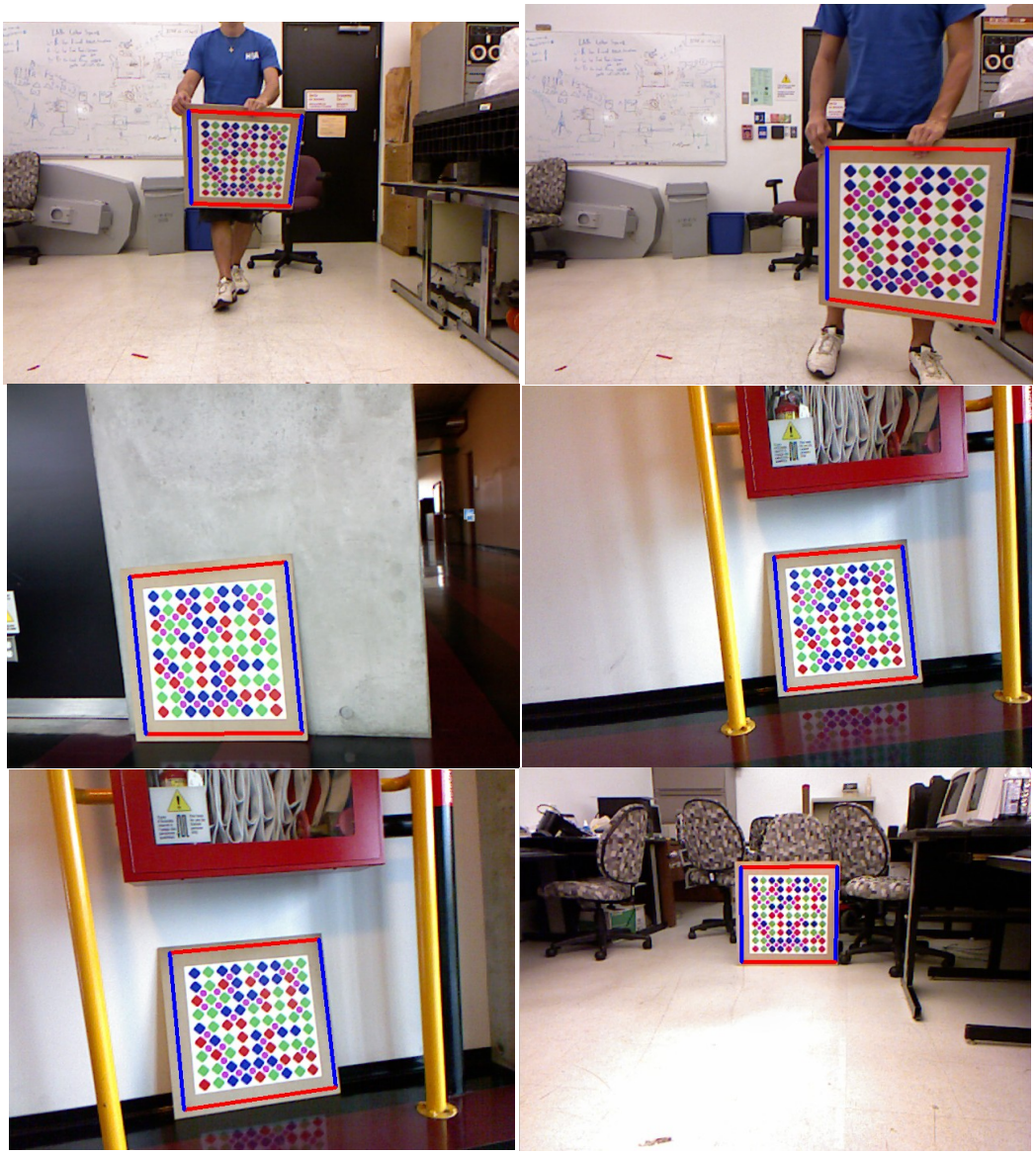


Figure 4.7: Various sample marker detections images

4.2 Chapter Summary

This chapter provides an overview of the detection process for a robust Pseudo-Random Array fiducial marker. It was observed that the detection algorithm made used of advance image processing techniques such as kd-trees,

adaptive thresholding, and k-means clustering to quickly and accurately determine if a marker is present in the scene. The proposed algorithm went further, defining a least square function with the purpose of determining the precision rotation and translation parameters of the camera with respect to the marker. Lastly, as an optimization step, a marker tracking procedure was briefly mentioned.

Chapter 5. Integration of Marker Localization

This chapter provides an overview on how the designed fiducial marker can be integrated into an indoor localization system. Marker detection is not enough to allow a robotic platform to avoid obstacles and move about an environment. Other processes such as floor elimination, grid occupancy maps, and navigation must be addressed. Firstly, the estimation of the ground plane is detailed using the images recovered from a 3D sensor. Next, the construction of a grid occupancy map is explained using a modified depth image. Lastly, an overview of the mapping and navigation procedure is outlined.

5.1 Floor Elimination

The first stage involved in the integration of the fiducial marker towards indoor localization is ground elimination. The robotic platform that has been constructed for testing is equipped with an Xbox Kinect depth sensor. This sensor outputs two images: an RGB image and a depth image. The RGB image is a 3-channel color image similar to images one would expect to receive from off the shelf digital cameras. The depth image on the other hand represents the depth values (in meters) of all pixels in the RGB image. That is, the RGB and depth images are rectified such that any pixel in the RGB is mapped to a corresponding distance value in the depth image. Figure 5.1 illustrates the robotic platform used for testing as well as a sample of RGB and depth images obtained from the depth sensor. Important to note is that the depth image will often contain pixels that have invalid depth values, ‘nans’. These pixels can be

identified in figure 5.1 (b) by black pixels. Invalid pixels tend to be caused by large amounts of objects clustered in the scene and glossy surfaces. Smooth planar surface rarely contain invalid pixels.

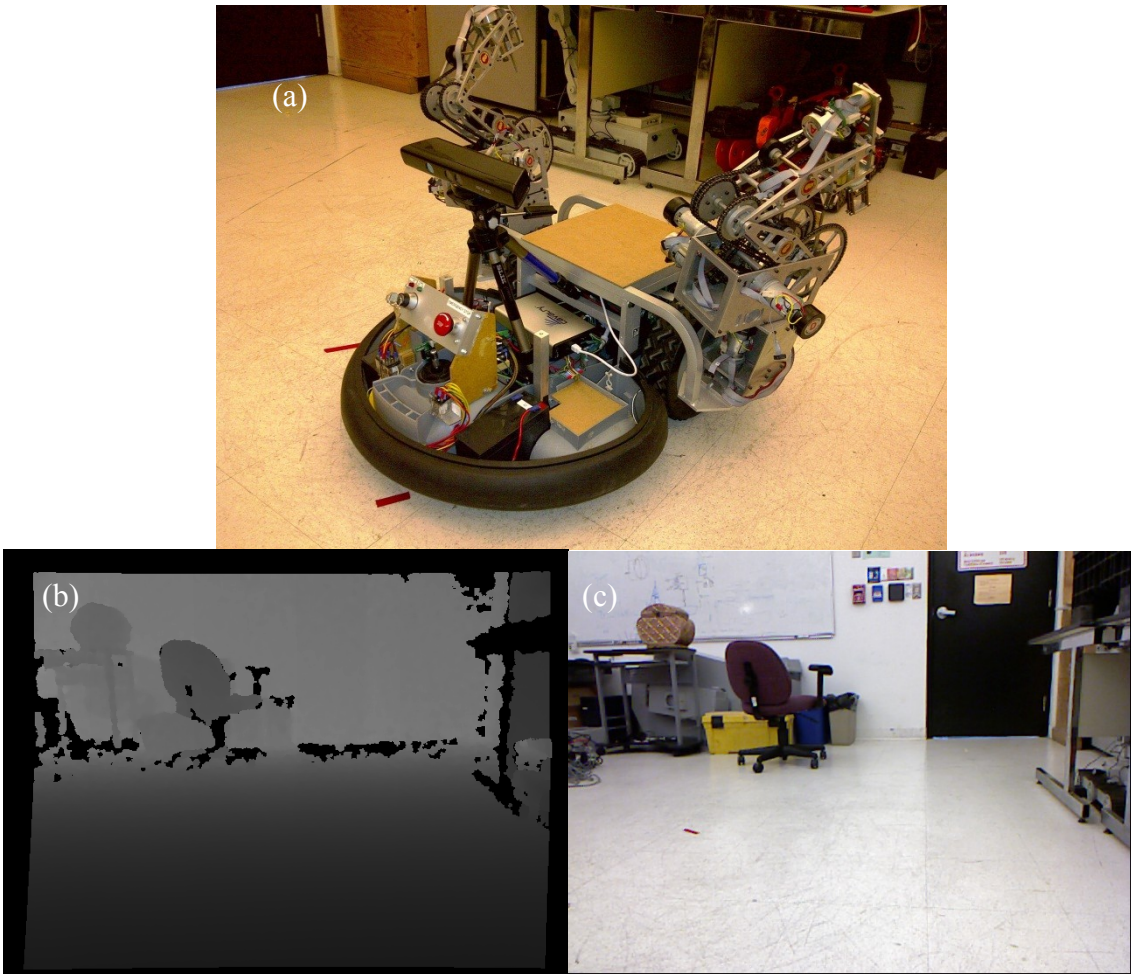


Figure 5.1: Robotic platform and sample Kinect images (a) the robotic platform used for testing (b) sample of a depth imaged obtained from the Kinect sensor (c) sample of an RGB image from the Kinect sensor

The robotic system will interpret objects in the environment using the depth image. Figure 5.2 illustrates a single 3D point cloud that combines the

RGB and the depth images obtained from the sensor. The main issue that arises when using this combined point cloud for navigation is that the floor is detected as an object. This poses a great problem because the robot will always see the floor plane as the closest object in the scene and the control algorithm will not allow movement in the forward direction. A solution to the problem is to estimate the equation that represents the planar surface of the floor and then to elimination any depth points that lie on this plane. In effect this would remove the floor plane leaving only objects that need to be avoided. The procedure to do this begins when the robot is first turned on.

Figure 5.2: RGB point cloud sample from Kinect sensor

During the initialization stage of the robotic system a rectangular area in front of the robot must be clear. The size of the region is illustrated in figure 5.3. Then using the intrinsic matrix of the sensor, all valid 3D points within this rectangular region are put into matrix form. The following least squares planar

estimation is employed to obtain the 4 coefficients that describe the planar surface, where a plane is defined by equation 5.1.

All the sampled points, S, are taken from $\mathbb{R}^3$, that is to say, each point has an x, y and z parameter. Since the area in front of the robot was assumed to be cleared, points are approximately located on the floor hyperplane. The model of this hyperplane satisfies the follow equation:

$$Ax + By + Cz + D = 0 \tag{5.1}$$

where A, B, C and D are constants that define the normal of the hyperplane.

Step 1: The first step consists of determine a point on the fitted plane. From [**66**] a least square solution has the same centroid as the initial data set. Thus a point on the fitted plane can be defined as follows:

$$P = \big(mean(X), mean(Y), mean(Z)\big) \tag{5.2}$$

where X, Y, Z are the sets of all x, y, and z values for points in the data set, S.

Step 2: The next step is to use SVD to minimize the error between all points in the sampled data set and the centroid, P.

$$[U, \Sigma, V^T] = SVD\big(X - P(1), Y - P(2), Z - P(3)\big) \tag{5.3}$$

Step 3: Find the smallest eigenvalue of SS^T that is contained in U. This eigenvalue will represent the normal of the best fit hyperplane.

$$A = U_{min}(1), \quad B = U_{min}(2), \qquad C = U_{min}(3) \tag{5.4}$$

$$D = A \cdot P(1) + B \cdot P(2) + C \cdot P(3) \tag{5.5}$$

Once the four plane parameters are determined all 3D points from the depth sensor can be inputted to equation 5.1 to determine if they line on the fitted plane or not. The hyperplane equation is recalculated every frame using all point that fit the previous frame's hyperplane model. Therefore, this algorithm provides an adaptive floor eliminator. The algorithm is robust enough such that the camera can be slowly rotated without affecting the floor segmentation.

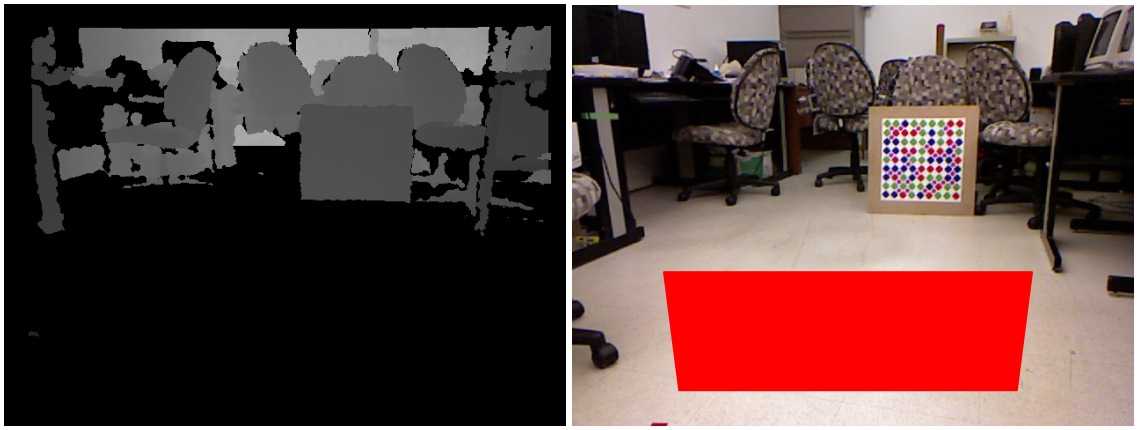


Figure 5.3: Depth and RGB floor elimination. The red box in the RGB image indicates the region of the image that is used to sample floor points and determine the equation of the hyperplane.

5.2 Grid Occupancy Map and Fake Laser Scan

Once the ground plane of the depth image has been removed, all non-zero pixels are assumed to be objects located at some distance, d_{ij}, from the camera. The value of d_{ij} is given by the magnitude of the pixel in the i^{th} row and j^{th} column of the depth image. From the newly modified depth image a grid occupancy map can be created. A grid occupancy map is defined by a 2D array of cells, or in the case of voxel maps a 3D grid of cells, where each cell in the

grid can take on one of three values: occupied, empty, or unknown. Cells have predetermined dimensions and are usual square (or cubic for 3D data). An occupied cell represents a small area in the scene that is filled by an object. Empty cells represent an area in the scene in which the mobile platform is free to move around in. Unknown cells are cells which have not yet received sensor data.

Grid occupancy maps are a simple way of mapping the area directly in front of the robot. Grid cells are filled according to data available from any active sensor. Multiple grid occupancy maps at different poses can be combined in attempt to map out an environment. Figure 5.4 (b) illustrates a voxel grid of a sample laboratory scene while figure 5.4 (a) shows a grid occupancy map of a single image within this scene. Notice that in the case of the voxel grid (figure 5.4 (b)) it is not necessary to eliminate the floor plane since objects hold 3D information. Figure 5.4 (a) on the other hand requires floor elimination; otherwise most of the cells in the grid would become occupied, since the floor is present in a large percentage of the scene. We can think of the 2D grid occupancy map as a simplified bird's eye-view of the scene. In order to speed up computations during navigation we will be taking the 2-dimensional approach as opposed to creating voxel grids in each frame. Thus navigation maneuvers can operate in real time with current low cost system hardware.

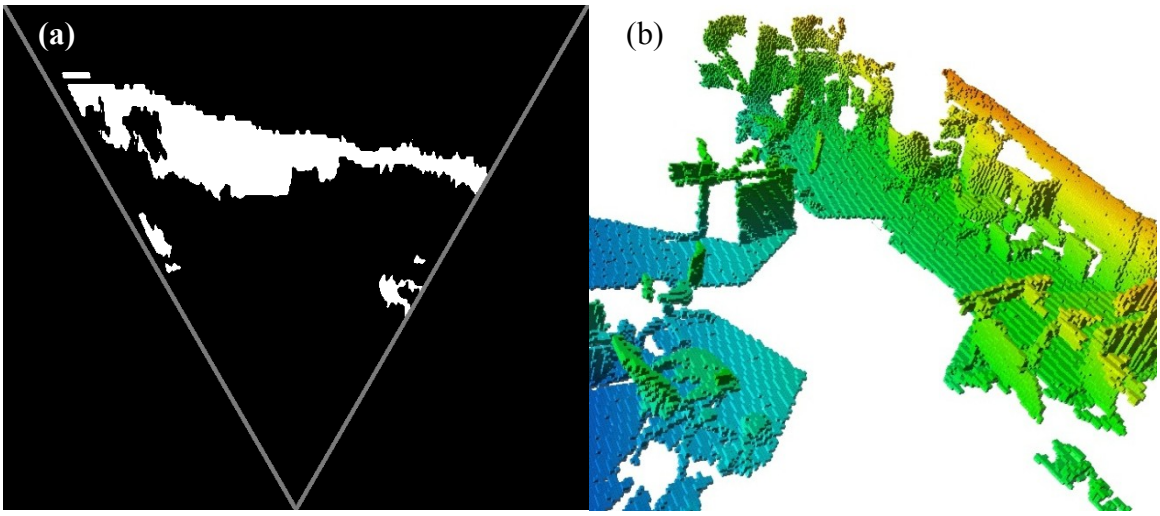


Figure 5.4: 2D and 3D grid occupancy maps (a) a sample 2D grid occupancy map where occupied cells are white, while empty and unknown cells are black. (b) a sample voxel grid (3D grid occupancy map) where white represents empty and unknown cells, while colored cells are occupied. Note: Occupied cells in the voxel grid have multiple colors to highlight the depth of objects in the scene, while occupied cells in the 2D grid map of (a) have a single color value.

To further improve system performance, 2D grid occupancy map can be simplified. This simplification occurs when only the nearest points of the grid are considered as occupied. The simplification comes about from the fact that data received from the Kinect is actually only 2.5D. Although points are expressed in 3 dimensions, only the front faces of objects can be seen by the sensor. That is, the sensor has no way of determining the full 3D description of objects in the scene when only part of the object is in view. In a single shot of the scene, the system can only identify the nearest 3D points of the object. Thus the best approach is to create the grid occupancy map using only the nearest data points

for the detected surfaces. For voxel grids, this process is straight forward since each measurement retrieved from the depth sensor represents the closest object point for that measurement in 3D space. Cells values can then be determined by splitting the scene into predefined sized cubes and taking a weighted average of the number of sensor point that fall into the cube. It is customary to define the cell as occupied when at least one sensor reading falls into the corresponding cube. 2D grid occupancy maps on the other hand differ slightly.

In order to create a 2D grid occupancy map, the 3D sensor data retrieved by the Kinect is projected onto a planar surface. This planar surface, mainly the ground plane, corresponds to the plane that the robot will be navigating on. Since the floor itself is assumed to be already removed, none of the floor points from the scene will occupy cells in the map. The improved and modified grid occupancy map requires the creation of a fake laser scan. Much like the Kinect sensor, a laser can only retrieve the close points of a detected object. The only difference is that laser scanners can only obtain a single horizontal (or vertical) scan line of the scene at one time, while the Kinect obtains multiple horizontal/vertical scan lines. Using the project points we will be constructing a new grid occupancy map consisting of a single scan line. This is accomplished by taking only the nearest depth value of each of the columns in the previously created 2D grid occupancy map. The result yields a simplified map of all objects within the cameras field of view and directly in front of the robot platform. Figure 5.5 and 5.6 illustrate the projection of the Kinect's 3D data points to a 2D surface

and the simplification of the original 2.5D grid occupancy map to a simplified fake laser scan one.

The simplification of the grid occupancy map allows for localization and mapping algorithms to process a significantly smaller amount of data. A four channel 640 x 480 (1, 228, 800 element) data structure gets reduced to a single data structure containing a maximum of 640 occupied cells or points.

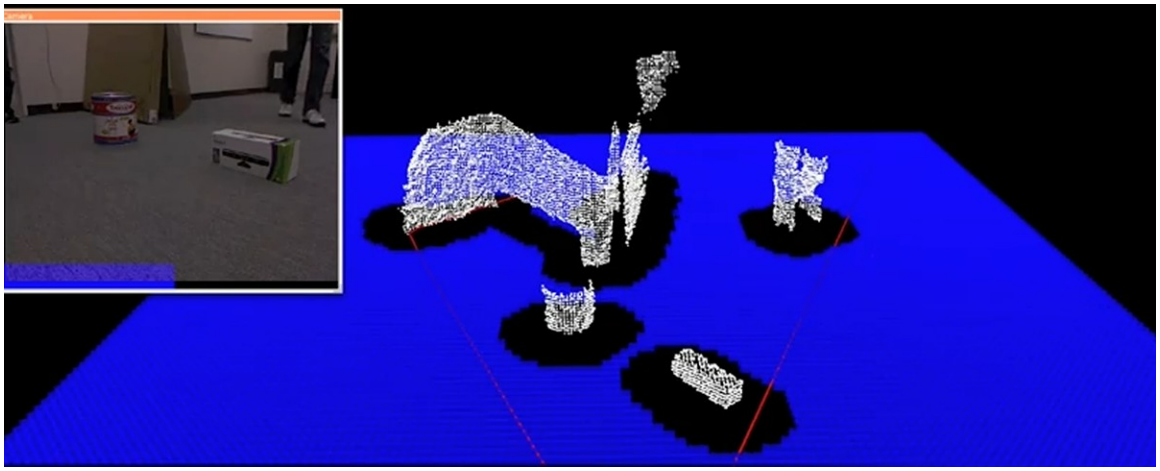


Figure 5.5: Projection of 3D grid occupancy map to floor plane. The floor plane is indicated by the blue quadrilateral. White points indicate the 3D grid occupancy cells while black cells represent the 2D projected points. *Image taken from [**67**]

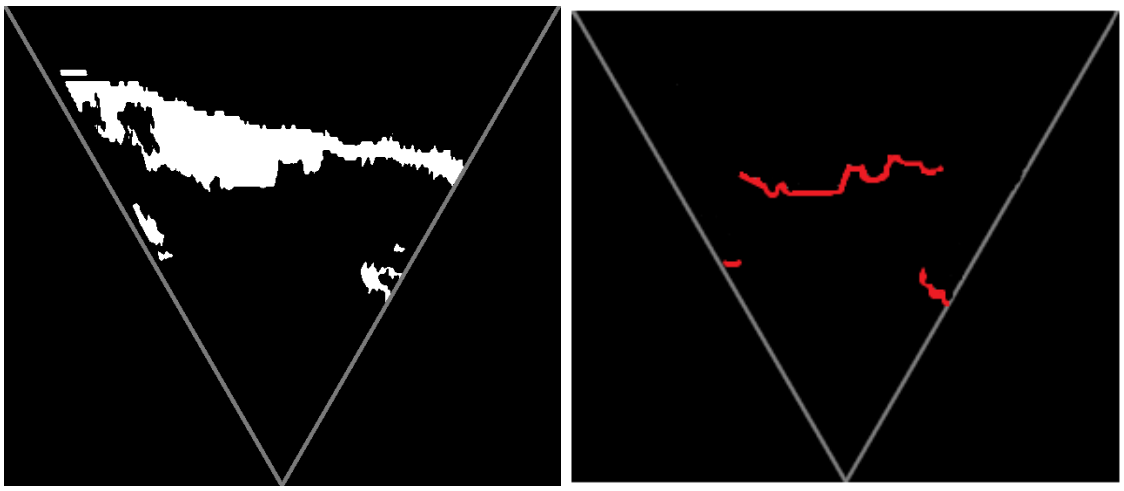


Figure 5.6: Conversion of 2.5D grid occupancy map to a simplified fake laser scan grid occupancy map

5.3 Localization

Localization can be described as the process of determining, or estimating, an object's location in space. Indoor localization is applicable to objects contained within a structured environment, such as a building. Localization implemented in the work of this thesis consists of two stages:

1) Stage 1 – Mapping: An offline stage in the localization process that consists of construction a basic static map of the environment.
2) Stage 2 – Real Time Navigation: The online stage of indoor localization which uses the static map obtained in stage one in order estimate the objects location based on sensor data output.

The following two sub chapters will outline each of the two localization stages as well as how the designed fiducial marker is integrated into the whole process.

5.3.1 Mapping the Environment

The localization implemented in this work is built on the concept of position synchronization. A number of markers are distributed within the environment at some key locations. These markers are used to correct any position errors that have accumulated through use of odometry, where odometry is implemented using two wheel encoders on a differential drive robotic platform. Upon detection of a marker the absolute position of the robot with respect to the marker becomes known. This transform is then used in combination with the pre-define location of the marker within the structured environment to calculate the absolute position of the robot in the global world reference frame.

Unfortunately it would be infeasible to physical cover every square inch of the environment with fiducial markers, not to mention there is a limit to the marker library size. Thus is it impossible to calculate absolute position of the robot in the entire space of the environment. The robot will rely on encoder counts to estimate its global position between the detection of markers. However, it is commonly known that optical wheel encoders accumulate error as the total distance traveled increases. The fiducial markers are strategically placed in order to remove these accumulated errors. Once a marker is detected, the global position of the robot will be updated using the pose obtained from the detected marker points. Then, when the marker leaves the image frame of the camera, the position of the robot once again relies on encoder counts.

The mapping stage consist of setting up markers in the scene, recording their global position values and building a basic map of static objects contained

within the scene. The mapping stage is an offline stage, meaning that is done before any autonomous navigation can be performed. It involves combining grid occupancy maps from the fake laser scans. The maps are combined using the information from the encoder to estimate the rotation and translation transform between successive scans. Assuming that the robot will be operating on a planar surface, we will align the x-y plane of the global and robot's reference frames with the floor plane of the building. This is illustrated in figure 5.7 below. Doing this simplifies rotation to a revolution about the z-axis, and translation to a movement in the x-y plane. When combining laser scans, an iterative approach is used along with an initial estimate calculated from encoder counts.

We begin the mapping process with a large 2 dimensional grid of cells. We can think of this as a large grid occupancy map, except that all cells are initialized with a value of 'unknown', meaning that no sensor data has been recovered yet. This grid is denoted as the global reference map. Each cell in the global reference map is of predetermined size, (ie. 1 cm by 1 cm). The initial position of the robot is located in the middle cell of the global map. This position takes on the value $x = 0$, $y = 0$ and $\theta = 0$, where x, y, θ are parameters that describe the robot's pose in the environment. Once fake laser scans are created they are first converted to local grid occupancy maps. These local maps can then be used to update the value of cells in the global reference map. At each new image frame, the encoder position information is converted to a single rotation and translation, which is in turn used as an initial guess in a registration matching of successive laser scans. The result obtained from registration matching is then

used to place the next local grid occupancy map into the global map. Figure 5.8 depicts this process well, as a sequence of map building frames are illustrated.

Thus far only the positions of static objects within the scene are recorded. However, it is also desired to mapping out the locations of fiducials. Once a marker is detected its global position is than added to a database along with the marker identification number. Before the addition is completed however, the detected marker ID is compared with all current entries in the database. If a marker with the same ID is already present in the database we update the database entry with an average of the previous entry's pose and the newly calculated pose information. The idea here is that the same marker can be detected in multiple frames as the robot traverses the scene. If we simply add new entries to the database every time the same marker is detected, we would have many duplicates with slight differences in pose information due to error in marker detection and odometry drift. At the end of this mapping stage we have a map of the scene as well as a database of all markers present within the scene.

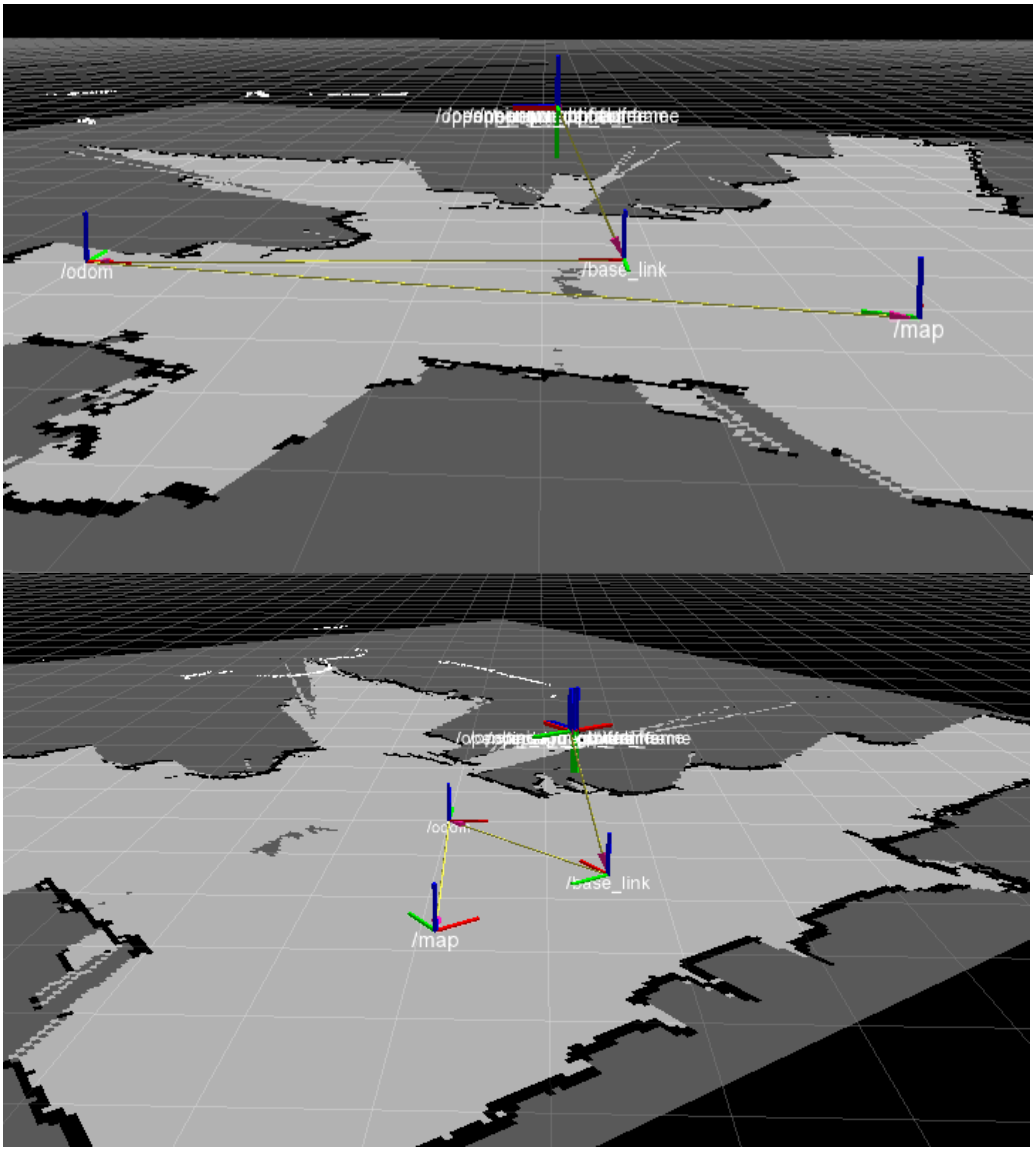


Figure 5.7: Alignment of global reference (/map) frame with robot reference frame (/base_link)

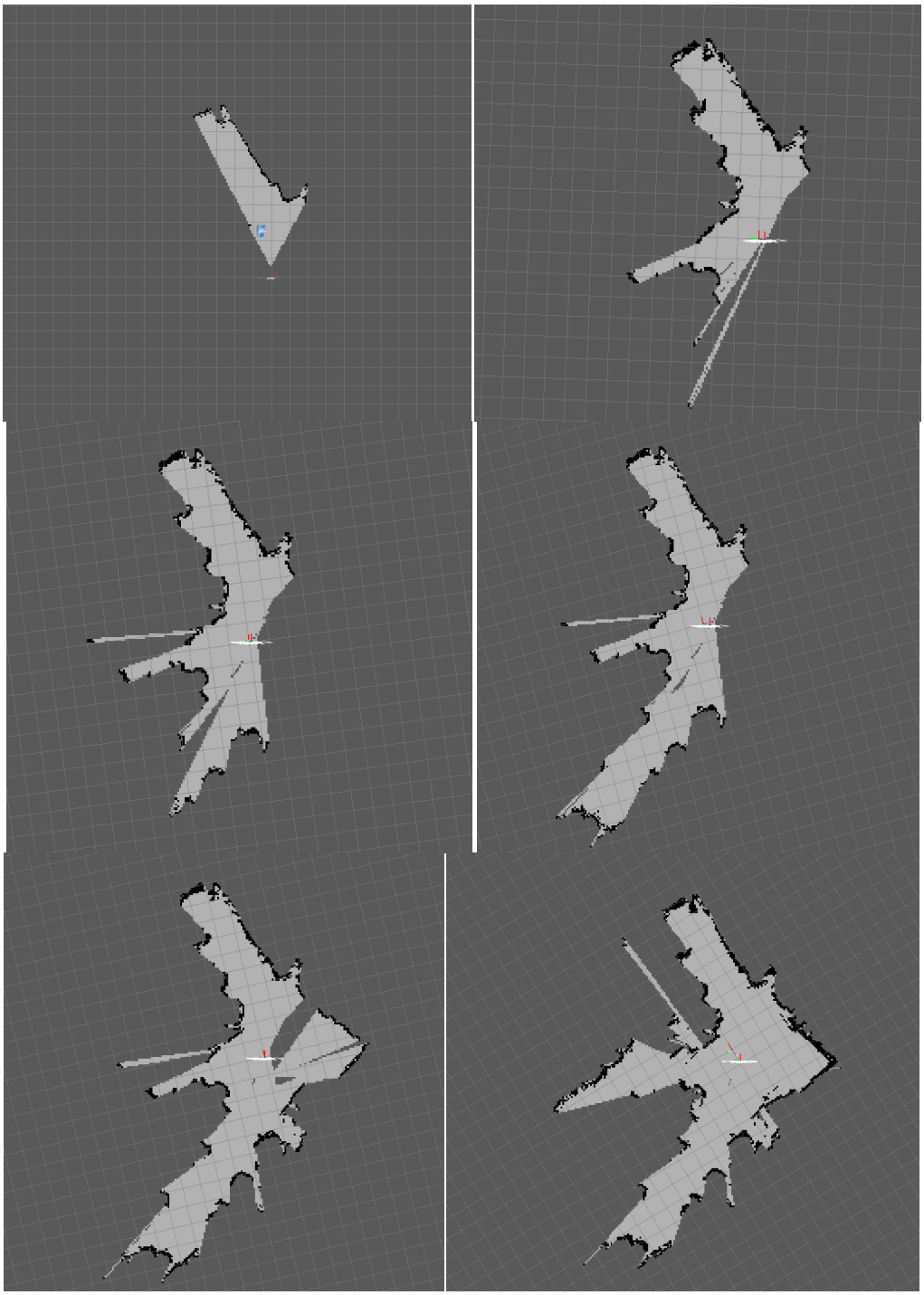


Figure 5.8: Fake laser scan matching to produce static map. Unknown cells are colored as dark grey, occupied cells are black and empty cells are defined by light grey.

5.3.2 Real Time Navigation

Navigation of robotic platforms using the fiducial marker scheme is meant to be a real time process that utilizes the static map and marker database created in stage one of localization. Similarly to stage one, this stage realizes on encoder counts to first estimate a rotation and translation between captured frames from the camera. This estimate is then refined by attempting to match successive laser scans using an iterative approach. Although the refinement process works well in areas of strong laser features, such as corners, it performs poorly in area where no noticeable features exist. Therefore, errors begin to accumulate over long distances. Fiducial markers are used to zero out this error by updating the robots pose information using the stored marker positions from stage one. Upon detection of a marker, a feature point correspondence allows us to obtain the transform between the marker and the camera of the robot. The marker position database then allows us to look up the transform that describes the markers position and orientation in the global reference frame of the map. A simple transformation multiplication is then used to determine the position of the camera with respect to the global reference frame. Thus, correcting any accumulated errors as result of the dead reckoning system. Figure 5.9 illustrates a sample set of RGB images obtained from the online navigation of the designed localization system, while figure 5.10 illustrates the corresponding laser scans overlaid atop a static map.

Figure 5.9: Sample set of RGB images from online navigation

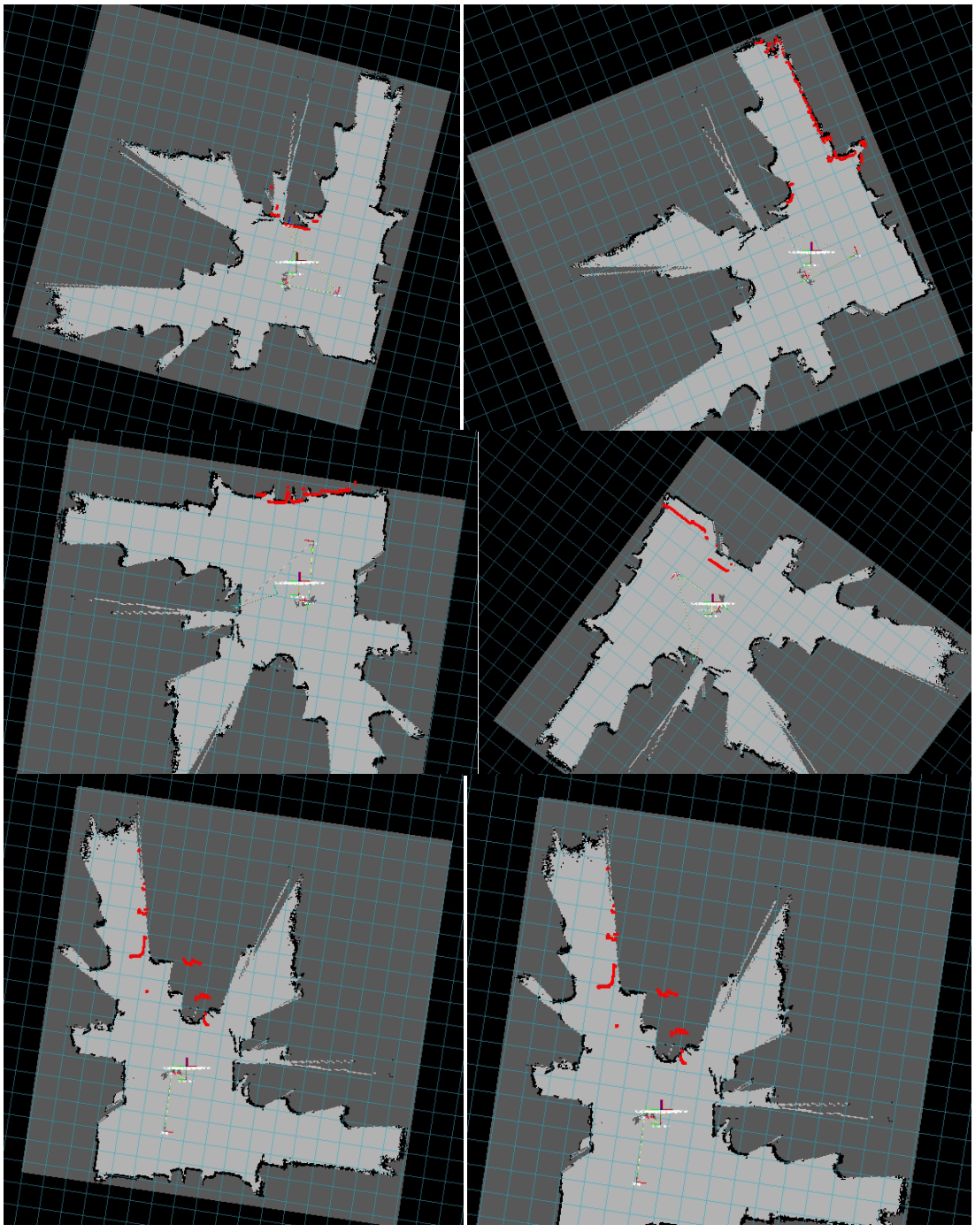


Figure 5.10: Sample set of online navigation scan data. The figure depicts the drift error associated with odometry measurements. These images correspond to the RGB images of figure 5.9.

5.4 Chapter Summary

This chapter described the process of integrating the designed fiducial marker system into an indoor localization methodology. Such a process requires a couple additional features to be implemented on the robotic platform. Floor elimination is used on 3D image data recovered from a depth sensor in order to remove the floor plane of the scene from being detected as an object. The results are then simplified into a fake laser scan such that a grid occupancy map can be created. The grid occupancy map provides key information on the location of objects within the scene. The first stage of localization was then outlined as an offline mapping stage, where multiple grid occupancy maps are combined using the rotation and translations computed from two wheel encoders on a differential drive robot. The global positions of markers are stored into a database which can be later used to recovery absolute position of the robotic platform upon marker detection. The marker structure implements a simple location synchronization technique that zeros out errors accumulated from relying on wheel odometry.

Chapter 6. Experimentation

This chapter provides an evaluation of the proposed fiducial marker system. The evaluation is based on the following three performance measures: marker detection rate as a function of distance, false negative detection rate, and inter marker confusion rate. The aim of any designed fiducial is to obtain the lowest of all three rates. Validation of the marker system is accomplished with the ultimate goal of integrating the marker into an indoor localization system. Hence recovered pose information from marker detection must yield results with an error that would be low enough for position recovery utilization.

6.1 Experimental Setup

The robotic platform used in the validation of the fiducial marker system has been custom made and is depicted in figure 5.1 (a). The platform is equipped with an Xbox Kinect sensor [**61**]. This is a motion sensing device that contains an RGB and range camera in order to capture 3D video data. The Kinect sensor is mounted on a tripod which is secured to the robotic platform. Images obtained from the sensor have resolution of 640 x 480 at 30 Hz. The robot has a differential drive system with additional encoder feedback. The robot is also equipped with an on board computer. The computer is an Acer Aspire 5745DG which contains a Intel core i5-460M processor, 4GB of DDR3 memory and a NVIDIA GeGorce GT 425M video card. All testing in this section was performance using a linux-based operating system, version kubuntu 11.04. Software packages that where utilized in marker detection and navigation

algorithms include OpenCV 2.3 [**27**], OpenGL and ROS version Diamondback [**68**].

The experiments conducted to evaluate the performance of the fiducial marker have the following setup. The robotic platform is fixed in the environment while the marker is set perpendicular to the camera's image plane. The marker can then be moved to various distance values in order to determine a relation between distance and bit errors. The marker can also be rotated to vary the angle the camera's optical axis makes with the marker. This will determine the maximum angle deviation the markers supports before detection fails. An example of the setup can be seen in the figure below.

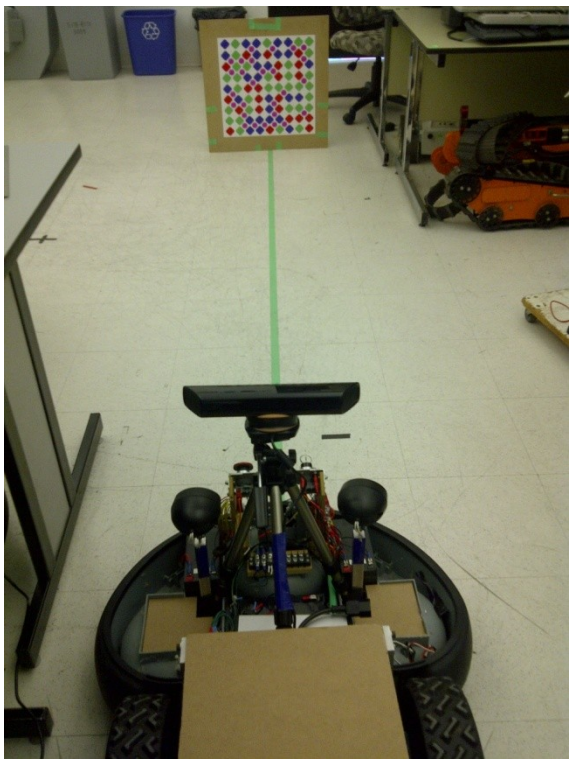


Figure 6.1: Experimental setup configuration

6.2 False Positive Detection

In [**11**] Fiala proposes a few performance measures for fiducial markers. One of these performance measures is the false positive detection rate. The false positive detection rate refers to the rate at which a marker is detected in a scene when no marker is physical present. This is proportional to the number of incorrect marker detections. In [**54**] it has been shown that implementing digital codes reduces the probability of false positives. Another factor that affects false positive detection is feature point uniqueness. Fiducials that have feature points similar to common object in the scene run a greater risk of detecting a marker when there is none present.

The proposed marker system utilizes a PRA as a unique feature space. Since this pattern is unlikely to be seen in nature indoor environments, such as office buildings, factories and plants, it is expected to have a low false positive detection rate. On top of the marker's uniquely defined feature space, it also implements digital codes which aim to reduce the probability of false detections through error correction. The implemented marker system uses both RS and CRC codewords to encode the marker's identification number. As stated in Section 3, this yields up to a 33.33% error correction capability. The ARTag fiducial marker system on the other hand is only able to correct 2 bit errors out of a total of 36 bits, yielding a 5.56% error correction capability [**11**]. From the error correction tolerance of both ARTag and the proposed marker, we would expect ARTag to have a higher false positive detection rate. Using a probabilistic

approach, one can compare the two markers by calculating the false positive detection probabilities.

The proposed marker system supports five symbol errors from a 15 symbol codeword. Since each symbol is 4 bits in length we have a total of 2^4 = 16 possible bit combinations per symbol. 15 out of the 16 combinations produce a single symbol error. This is due to the fact that there is only one valid symbol amongst the 16 combinations. A valid data sequence in the proposed system can be obtained from N_{valid} codes.

$$N_{valid} = B_0 || B_1 || B_2 || B_3 || B_4 || B_5 \tag{6.1}$$

where $B_0$is the number of codes with zero bit errors, $B_1$is the number of codes with one bit errors, $B_2$is the number of codes with two bit errors, $B_3$is the number of codes with three bit errors, $B_4$is the number of codes with four bit errors, $B_5$is the number of codes with five bit errors, and || represents the logical 'OR' operator.

Since there can only be one correct code, B_0 is equal to 1. For each symbol in the codeword a total of 15 bit combinations produce an error. Hence all possible single symbol errors, B_1, is equal to 15 (symbols in the codeword) x 15 (possible bit error combinations) = 225. B_2can then be calculated as 15 (symbols in the codeword) x 15 (possible bit error combinations) x 14 (remaining symbols in the codeword) x 15 (possible bit error combinations) = 47250. Similarly, one can obtain the values of B_3, B_4 , B_5.

$$B_0 = 1 \tag{6.2}$$

$$B_1 = 15 \times 15 \tag{6.3}$$

$$B_2 = (15 \times 15) \times (14 \times 15) \tag{6.4}$$

$$B_3 = (15 \times 15) \times (14 \times 15) \times (13 \times 15) \tag{6.5}$$

$$B_4 = (15 \times 15) \times (14 \times 15) \times (13 \times 15) \times (12 \times 15) \tag{6.6}$$

$$B_5 = (15 \times 15) \times (14 \times 15) \times (13 \times 15) \times (12 \times 15) \times (11 \times 15) \tag{6.7}$$

$$N_{valid} = B_0 + B_1 + B_2 + B_3 + B_4 + B_5 = 275316111226 \tag{6.8}$$

Each valid codeword can be obtained from a total of N_{valid} possible 60 bit sequences, thanks to the error correction capabilities of RS codes. Since the total number of markers in the library is 4096, the number of codewords that lead to any one marker being detected is

$$N_{valid} \times 4096 = 275\,316\,111\,226 \times 4096 = 1\,127\,694\,791\,581\,696 \tag{6.7}$$

The number of total possible 60 bit codewords can be expressed by 2^{60}. That is because each bit is binary and there are a total of 60 bits contained within the marker. Thus, randomly choosing and decoding a 60 bit sequence would have the following probability of yielding a valid marker identification number:

$$\frac{1127694791581696}{2^{60}} \approx 9.7811 \times 10^{-4} \tag{6.8}$$

Therefore if the predefined PRA is detected in the image when no marker is present, it has a 0.00097 percent probability of decoding a valid marker ID. The false positive probability of ARTag in comparison has been reported in [**11**] to be 0.0078%, which is greater than that of the designed marker.

The false positive probability is a theoretical number that should represent what happens in practice; however this is not always the case. In order to test the

practical results of false positive detection rates, frames from various locations throughout the building used for localization were compiled into one video stream. This video stream was then tested for marker detection using the following fiducial systems: ARToolKit, ARTag, and the designed PRA marker. Figure 6.2 shows a few images that were contained in the video stream while table 6.1 presents the results of the experiment.

Figure 6.2: Sample set of RGB images used for false positive detection

Number of Frames	False Positive Detections					
	ARToolKit (10 Marker Library Size)				ARTag	Proposed Marker
	CF = 0.6	CF = 0.7	CF = 0.8	CF = 0.9		
180215	1987	330	50	15	0	0

Table 6.1: False positive detection results of ARTag, ARToolKit and the PRA marker. CF indicates the confidence factor used for ARToolKit marker detection with a marker library size of 10.

From the results of table 6.1 we see that both ARTag and the designed fiducial record zero false positive detections. This result corresponds with what would be expected to see through the use of digital codes. In fact, throughout all the testing performed on the designed fiducial marker there has not been a single reported false positive detection. ARToolKit on the other hand suffers from a fairly large false positive detection rate, a rate which varies according to the confidence factor used in marker detection. Important to note here is that the higher the confidence factor, the high the probability that marker in the scene will not be detected. In [**40**] it was confirmed that the false positive detection rates of ARToolKit dramatically increases as the marker library size increases. The library size dependence does not affect markers using digital coding techniques, such as ARTag and ARToolKit Plus.

6.3 Inter-Marker Confusion

Inter-marker confusion is another performance measure for fiducial markers which is related to the distinction of fiducials in the same marker system. In the case of the proposed fiducial, markers are differentiated via the inner 60 bit

data region. Accordingly we can minimize the inter-marker confusion rate by maximizing the hamming distance between the 60 bit codewords of fiducial markers. Hamming Distance is the best estimate for possible inter-marker confusion rates between any two markers in the library [**69**].This is one of the advantages for implementing digitally encoded markers. Correlation type markers such as ARToolKit on the other hand must rely on choosing data patterns that have low correlation between other markers in the database. To analyze the inter-marker confusion of the designed marker the statistical approach described in [**11**] will be taken.

The probability that one binary code can be mistaken for another can be defined by equation (6.9).

$$P_{xy} = p(n)q(n) \tag{6.9}$$

where $p(n)$ is the probability of n bits being falsely detected, $q(n)$ is the probably that n bits are correctly detected and n is the hamming distance between the two codes. If we make the assumption that bit errors are uncorrelated and independent, the probability $p(n)$ becomes p^n and $q(n)$ becomes $(1-p)^{N-n}$, where N is the total number of bits in the code and p is the probability of a bit error.

$$P_{xy} = p^n(1-p)^{N-n} \tag{6.10}$$

Assuming that we have a set of codes with varying hamming distances, the probability that any one code, X, is mistaken for another codeword in the set is given by equation 6.11 [**11**]. Notice that this probability relies on the hamming distance between all other codes in the set.

$$P(\neq X) = \sum_{n=1}^{N} HD(n) \cdot P_{xy} \tag{6.11}$$

where $HD(x)$ is the number of codes with hamming distance n.

For the proposed marker with a 60 bit codeword, equation 6.11 becomes:

$$P(\neq X) = \sum_{n=1}^{60} HD(n) \cdot p^n (1-p)^{60-n} \tag{6.12}$$

in view of the fact that the probability p cannot be easily changed, the marker must be designed such that the function $HD(n)$ is very small for low values of n. In order words the marker must maximize the hamming distances between codes. This is because the term $p^n(1-p)^{60-n}$ in equation 6.12 is fairly large at low values of n in comparison to the same term at higher values of n.

Referring back to the encoding structure of the fiducial marker, two approaches were used, mainly Reed Solomon and Cyclic Redundancy Check codes. The generator polynomials used for these coding schemes affect the bit separation between codewords. Altering the chosen polynomial will affect the hamming distance between codes. Hence a brute force method to maximize hamming distance by altering generation polynomials can be used to minimize equation 6.12. In effect this will result in decreasing the overall inter-marker confusion. Table 6.2 lists a few the generator polynomials for CRC-8 and RS(15, 5). Figure 6.3 illustrates the histogram of hamming distances corresponding to all combinations of generator polynomials seen in table 6.2. The numerical values used to plot figure 6.3 can be found in Appendix D.

Code	Polynomial	Hex
RS1	$x^4 + x + 1$	0x13
RS2	$x^4 + x^3 + 1$	0x19
CRC0	$x^8 + x^7 + x^4 + x^2 + x + 1$	0x97
CRC1	$x^8 + x^2 + x + 1$	0x07
CRC2	$x^8 + x^5 + x^4 + 1$	0x31
CRC3	$x^8 + x^7 + x^6 + x^4 + x^2 + 1$	0xD5
CRC4	$x^8 + x^4 + x^3 + x^2 + 1$	0x1D
CRC5	$x^8 + x^7 + x^4 + x^3 + x + 1$	0x9B

Table 6.2: Generator polynomials for RS(15, 5) and CRC-8

Referring to figure D2 in appendix D, or alternatively Figure 6.3, one can pin point the RS and CRC generator polynomial combination which yields the lowest hamming distance frequencies for the first 5 bins. These first 5 bins are the values that will dominate equation 6.12 and hence need to be minimized. The optimal generator polynomial combination for the lowest inter-marker confusion rates is RS2 and CRC1 as defined in table 6.2. The minimum hamming distance for this combination is 17. Therefore the chosen Reed Solomon primitive polynomial for the final marker library is RS2 and the chosen CRC generator polynomial is CRC1.

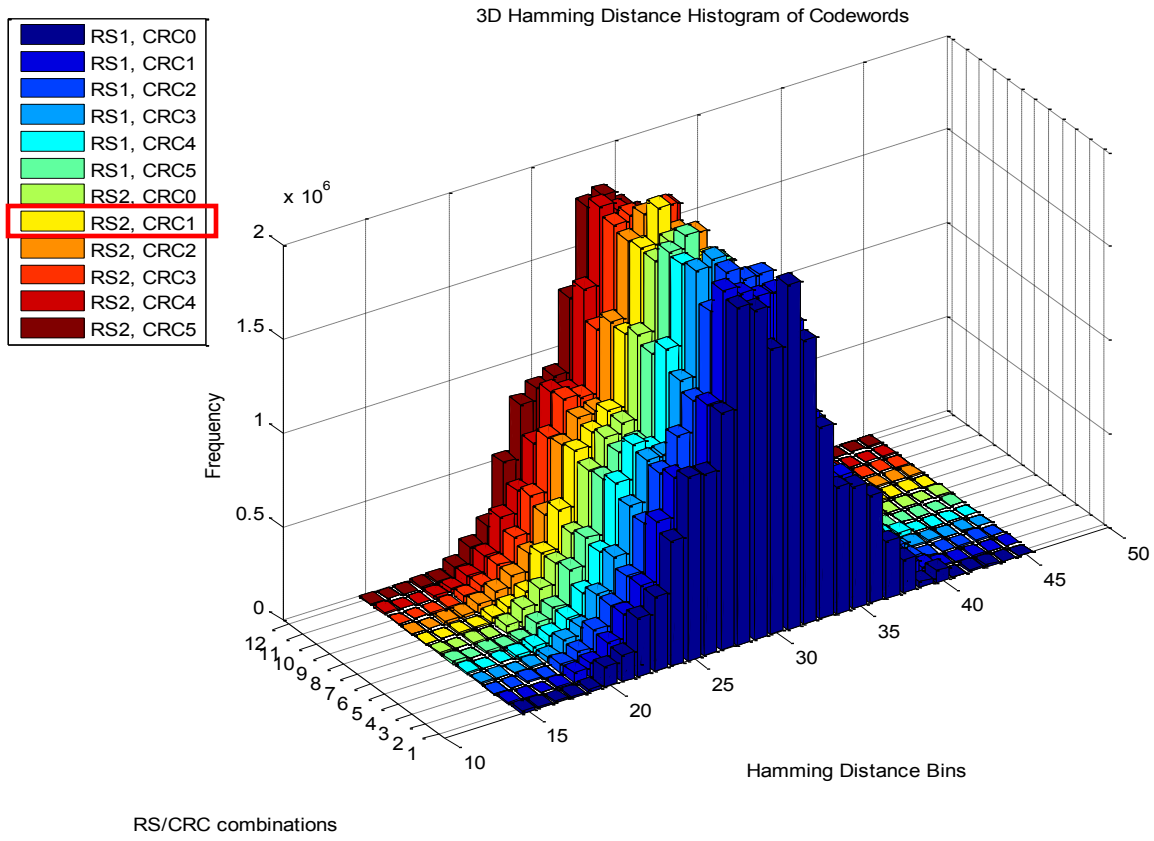


Figure 6.3: Inter-marker confusion analysis. A graph depicting the combined histogram of hamming distance frequencies while adjusting generation polynomials for different encoding methods. Note that bin value less than 15 where not considered since the minimum hamming distance for all codewords was above 14.

6.4 Occlusion Experimentation

Based on the number of correctible symbol errors of a fiducial marker, a theoretical maximum occlusion percentage can be obtained. For the designed fiducial marker a total of 5 symbol errors can be corrected. This yields a maximum occlusion percentage of 5/15 = 33.333%, which is fairly large in comparison to other fiducial markers, as reported by Kohler *et al.* in [**40**]. ARTag

supports occlusion of 5% while ARToolKit, Intersense, and TriCode [**22**] [**12**] [**40**] do not support occlusion at all. ARToolKitPlus in [**38**] makes use of CRC codes, however an ability to recover occluded bits is not indicated since CRC codes are used only for error detection.

In order to test the partial occlusion properties of the purposed marker within a dynamic indoor environment, a simple occlusion validation was performed. This validation consisted of fixing both a marker and the robotic platform used for testing into the environment. The marker was placed roughly 2.7 meters perpendicular to the camera's optical axis. Then a dynamic object, in this case a person, would occlude the marker by passing directly in front of it. The physical occlusion of the marker by the object and the estimated occlusion, using percentage bit errors, were graphed to visually analyze their correlation. A sample of images taken from the test can be seen in figure 6.5, while a graph of the physical occlusion and percentage bit error is depicted in figure 6.4.

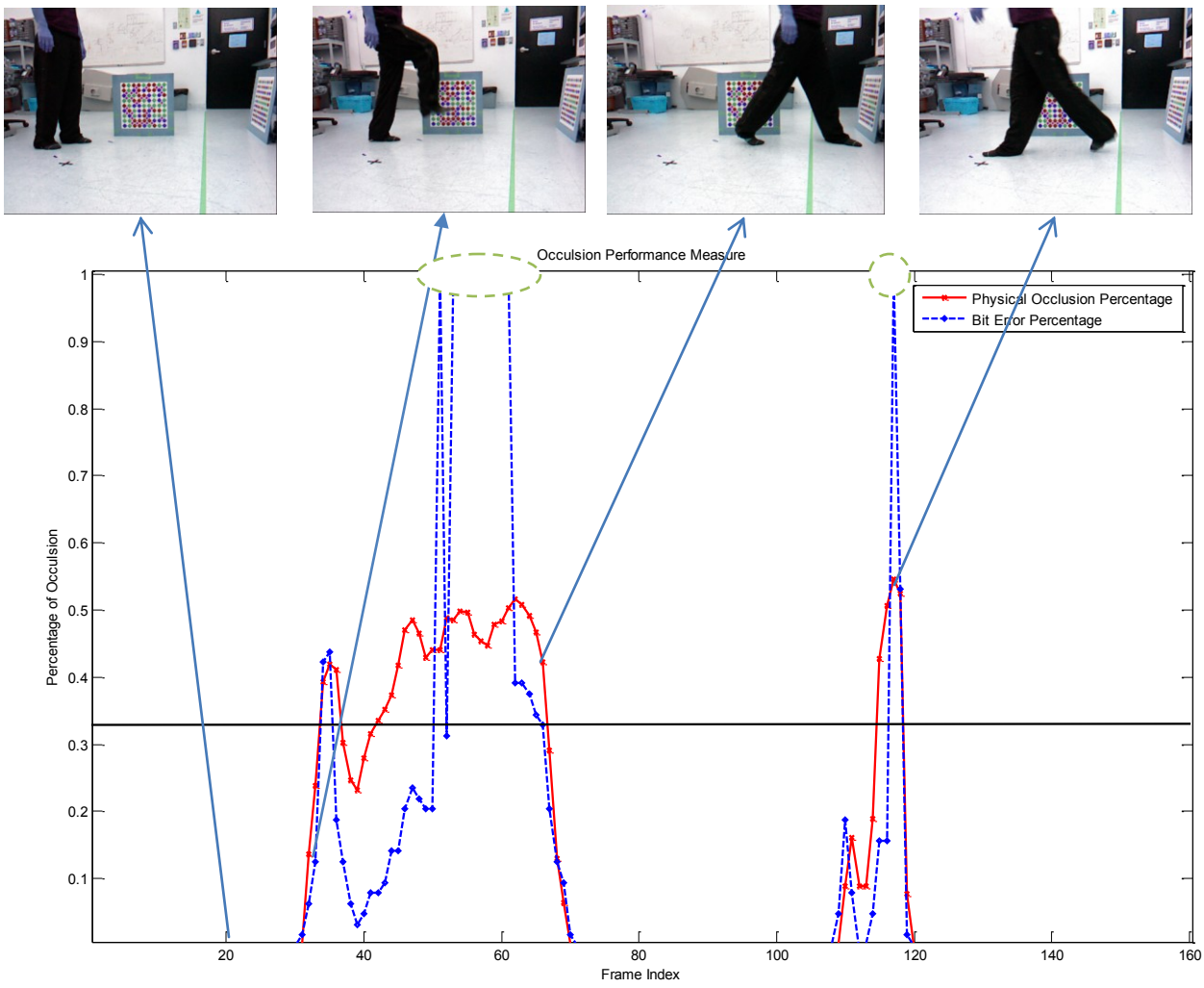


Figure 6.4: Performance graph for occlusion test case. The red line indicates the physical occlusion percentage of the marker while the blue dotted line represents the percentage of bit errors.

Important regions in the graph include: the horizontal (black) line representing y = 0.3333 and the data points contained within the two dotted ellipses (green). The horizontal line is used to represent the maximum amount of bit errors that the marker can withstand before decoding fails. The data points within the dotted ellipses indicate frames at which the pseudo random array of

the marker was not detected. Thus, data decoding was not performed, leading to detection failure. Upon first glance of the graph, it is evident that the two occlusion measures are highly correlated since the plot contours are closely matched. There also exits data points in the graph with have a high physical occlusion but low bit errors, such as, frame number 47 (x = 47). The cause of this can be attributed to one of two things:

1) Physical occlusion was concentrated in areas of the marker which do not contain data points, ie. any of the four corners, or the area between PRA elements and data points.
2) When sampling occluded data pixels of the marker, their image intensity values matched those of the occluded data bits. In order words, the object occluding the marker is represented in the grayscale image by intensity values similar to those of the pink data circles included in the marker.

Overall a total of 160 images where analyzed, leading to a fairly high detection rate. More than 50% of the time when the marker was occluded, detection still occurred. This experiment indicates a typical situation for marker use in dynamic indoor environments. It is expected that large objects such as people walking about the environment would frequently cause marker occlusion. It is then necessary for a localization system to estimate the robot's position under partial occlusion. Providing a maximum occlusion rate of 33.33% the designed marker satisfies the detection under partial occlusion requirement, which makes the marker an excellent candidate for use in localization systems.

This experiment was repeated several times, verifying that marker detection occurs more than 50 percent of the time when the marker was occluded, due to someone walking in front of it.

Figure 6.5: A sample image sequence from the 160 images used to graph figure 6.4.

6.5 3D Pose Estimation Experimentation

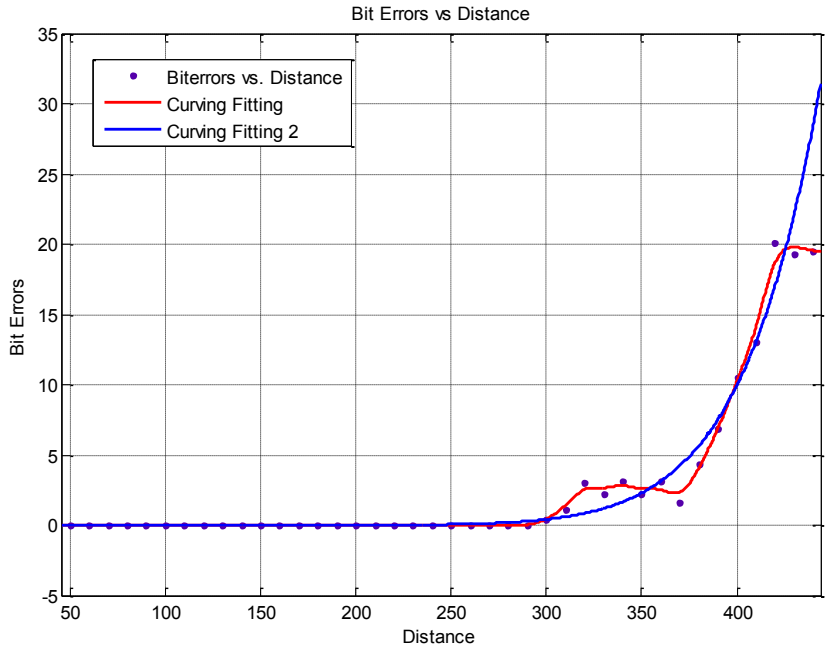
In order to determine the performance of pose estimation a series of three tests were conducted. The first two are used to determine the limitations of marker detection, while the third one is used evaluate the stability associated in the calculation of 3D position. These three tests are perhaps the most important

in determining the feasibility of the designed fiducial marker for use in an indoor localization system. The calculated pose between the marker and the fiducial is the only component used to zero out any error incurred from the dead reckoning system, ie. wheel encoders. Any calculated pose needs to be fairly accurate with high stability.

6.5.1 Distance Limitation

The first experiment consists of measuring the runtime and number of bit errors as a function of distance. Runtime in this case has the following meaning: the time it takes for marker detection algorithm to complete, from image acquisition until decoding the marker ID. In order to facilitate measurement of these parameters, a simple yet effective setup up was constructed. The Kinect sensor was physical mount atop the mobile platform. For the duration of the test, the mobile platform was static and fixed to one location in the environment. A line was then estimated from the optical center of the device and extended outward a minimum of 5 meters. Figure 6.1 illustrates the 2D projection of this line onto the floor plane through the use of a green marker. The fiducial marker is then placed on this line, such that the plane of the marker is roughly perpendicular to the image plane of the camera. This is repeated a number of times, varying the distance between the marker and the camera. At each new distance measurement 200 video frames are taken. For each frame the marker detection algorithm is executed, calculating the total run time and number of bit errors. Figure 6.6 a) and b) depict the results of this test.

(a)



(b)

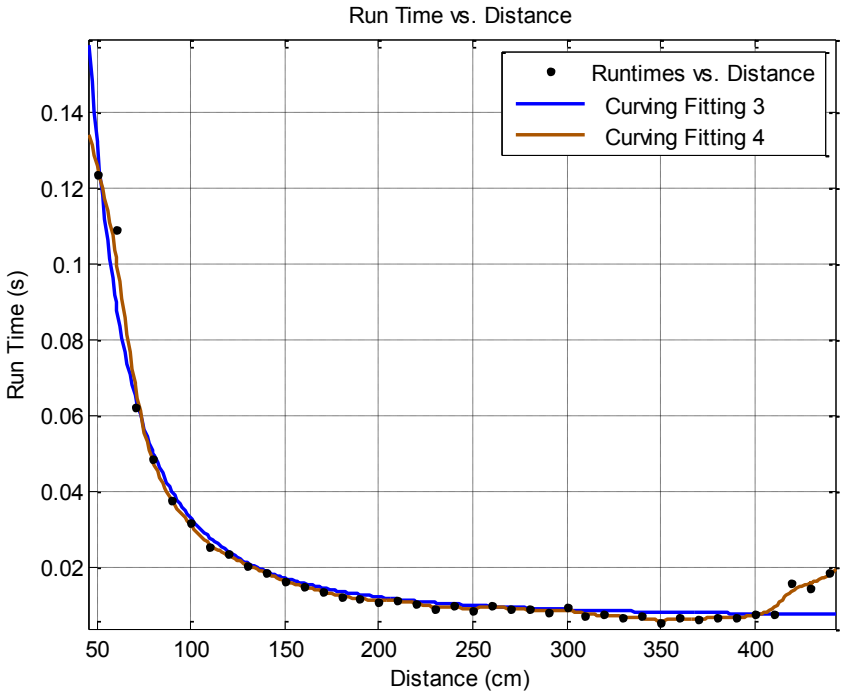


Figure 6.6: Results for test case one. (a) A graph of the number of bit errors detected as a function of distance between the marker and the camera (b) A graph showing how run time of the marker is affected by distance

For each of the graphs in figure 6.6, two curves were fit to the data points. The first of the two simply follows the data points such that a better visual of the information can be constructed, while the second (blue) aims to estimate the function that best fits the data. In the case of bit errors versus distance, we see with distances less than 3 meters, marker detection performs very well with little or no errors. However after the 3 meter marker, bit errors start to rapidly increase relying on error correction techniques to properly decode the marker. Since the maker system is able to correct for up to 20 bit errors, marker detection succeeds for over a 95% detection rate until a 4.3 meter separation. From various curve fitting techniques we can best model the relation of bit errors versus distance using the following equation:

$$f_{be}(d) = Ad^b + c \tag{6.13}$$

$$A = 1.831\ x\ 10^{-28}, b = 11.04\ and\ c = 0$$

Where A, b, c are constants and d is the separation between the camera and the marker in meters.

Turning to the second graph of figure 6.6, run time versus distance, we see an opposite trend as that of bit errors. At close distance values the marker detection algorithm is fairly lengthy. Distances of greater than one meter yields run times less than 0.033 seconds in duration. The greater the distance between the marker and the camera, the lower the run time becomes. We can attribute this quality to the decrease in pixel size of the marker as it is moved further from the sensing device. Thus the bottle neck in run time of the detection algorithm is

strongly related to the pixel size of marker feature points. The model of the blue curve in figure 6.6 (b) is described by the following formula:

$$f_{rt}(d) = Ed^{g} + h \tag{6.14}$$

$$E = 870.7, g = -2.261 \; and \; c = 0.006598$$

Where E, g, h are constants and d is the separation between the camera and the marker in meters.

6.5.2 Angle Deviation Limitation

A second validation test, very similar to the first, was conducted to determine the angle of incidence at which marker detection begins to fail. The test is arranged in the same way as the distance validation test, in which the sensing device is fixed in the environment and the marker is moved at various distances that are perpendicular to the optical axis of the camera. At each marker placement we repeatedly rotate the marker 10 degrees, for a maximum of 9 iterations in the clockwise direction, recording the number of bit errors incurred. This process is also repeated for the counter clockwise direction before re-positioning the marker at a new depth point. The experiment was conducted for 8 different depth values and the results are illustrated in figure 6.7. An important factor to consider

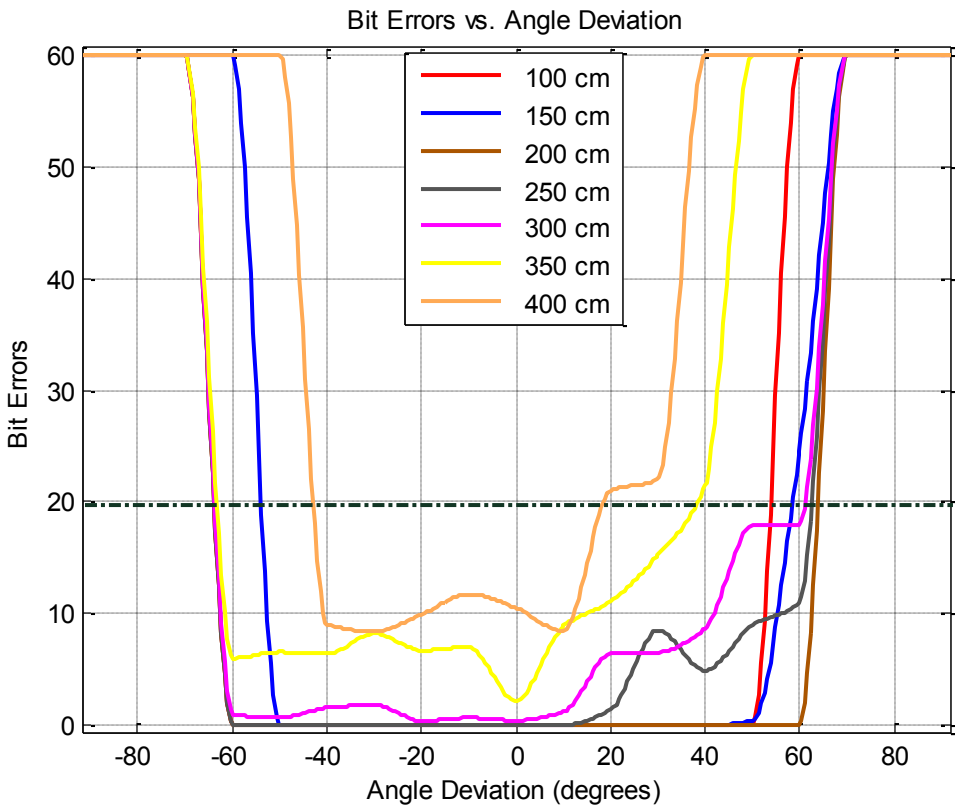


Figure 6.7: Results for angle of incidence test case.

Determining the maximum angle of incidence for the fiducial marker is important in analyzing it's feasibility for indoor localization. The angle of incidence in this situation can be defined as the angle the optical axis of the camera makes when it intersects the surface of the marker. Decreasing the maximum detectible angle of incidence limits the area in the environment at which the marker can be detected. Figure 6.7 illustrates the relation of bit errors as a function angle deviation for multiple distance poses. The black horizontal dotted line indicates the maximum number of bit errors that can be incurred before marker detection fails. From this plot it can be concluded that the maximum angle of incidence is roughly 45 degree for distances up to 350 cm. It is important to note however this

values changes as a function of distance. At lower distance values a maximum angle deviation of 60 degrees is obtained, while at large distance values, such as 4 meters, the maximum incidence angle is about 25 degrees. A small sample set of images taken from the validation test case is shown in figure 6.8.

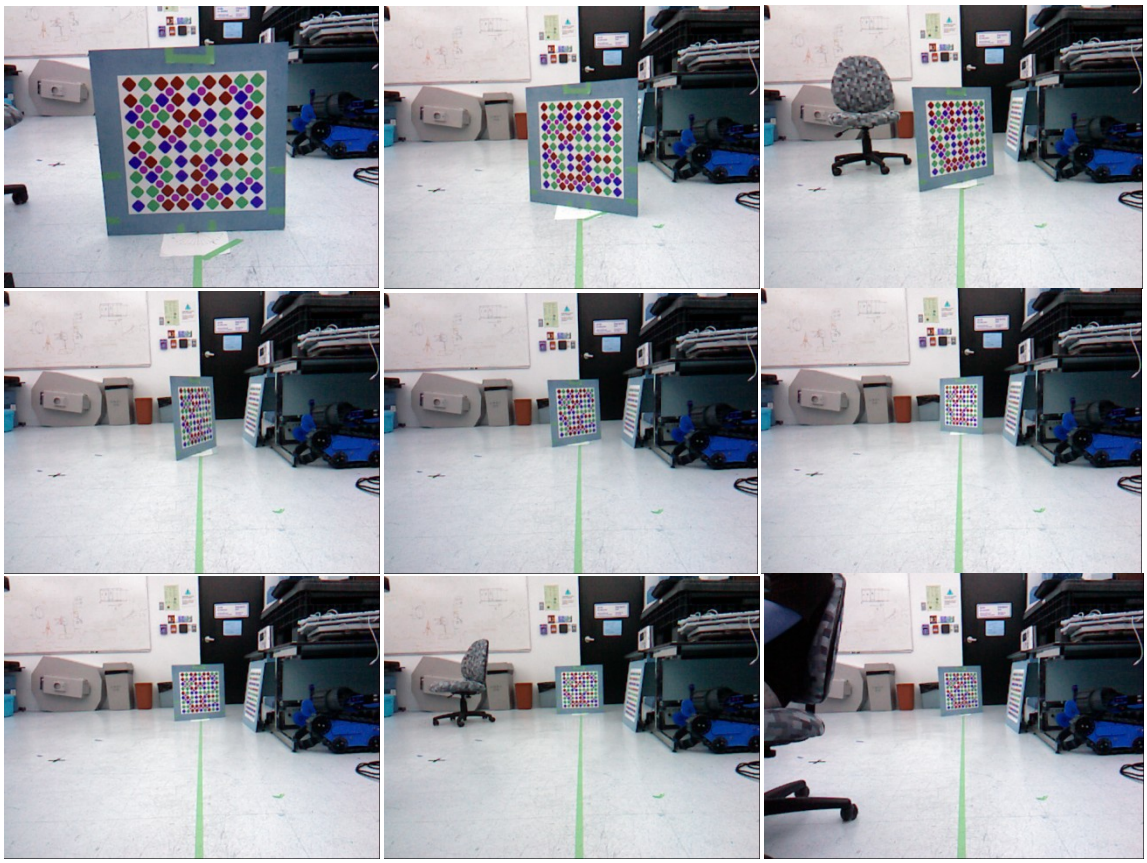


Figure 6.8: Sample set of images used to validate maximum angle of incidence.

6.5.3 Pose Estimation Stability

The last of the three validation tests aims to classify the marker system's stability in pose estimation. That is, we wish to determine how much the calculated rotation and translation parameters change upon successive marker detections in a static setup. Recall that the distance and angle deviation test cases involved recording detection data as the distance between the marker and the camera was varied. At each distance value the rotation and translation of 200 frames were recorded. Using those results we are able to analyze the marker's stability by calculating the standard deviation of the 200 frames. The same 200 frames were also used to determine error in the calculated translation pose data. For the sake of simplicity this thesis will only be looking at the error in z-axis of translation. Error analysis for other parameters such as rotation about x, y, and z axis' and translation about the x and y axis' would be difficult to determine. This comes about from the very difficult task of making sure the marker is translated perfectly along the optical axis of the camera. And since the reference frame of the camera is unknown with respect to the environment, we are unable to determine an exact ground truth of where the marker is with respect to the camera. The reference frame of the camera can only be determined with respect to another object, and this is accomplished using image processing techniques.

Figure 6.9 shows a plot of the estimated translation about the z axis as a function of ground truth distance between the camera and the marker. In an ideal situation this plot would yield a line (f(x) = x), where the translation about the z-axis is equal to the separation between the marker and the camera. The ideal

situation is expressed when there is no error present in the system and the detection algorithm is able to perfectly calculate the marker's pose. However, since we are in no way dealing with an ideal situation, we expect to see a bit of error in estimated pose data. The data of figure 6.9 can be modeled using a linear equation and then compared to the ideal solution of f(x) = x.

$$Model:\ f(x) = \ 1.024x - 5.129 \qquad (6.15)$$

Equation 6.15 models the experimental result of figure 6.9, where x is ground truth distance between the marker and the camera and f(x) is the estimated distance between the marker and the camera. Equation 6.15 is very close to the ideal case of f(x) = x. Notice the slopes are very closely match, a slope of 1 as compared to the experimental slope of 1.024. The two line equations intersect at an x-value of 213.708 which corresponds to optimal distance in which pose estimation would be most accurate, ie. translation error will be at a minimum. At distance separations of 4 meters, the amount of error that can be experienced is up to 5 cm in depth accuracy. This error can be attributed to error incurred from the Kinect sensor, as well as noisy image intensities causing miss classification of colored pixels.

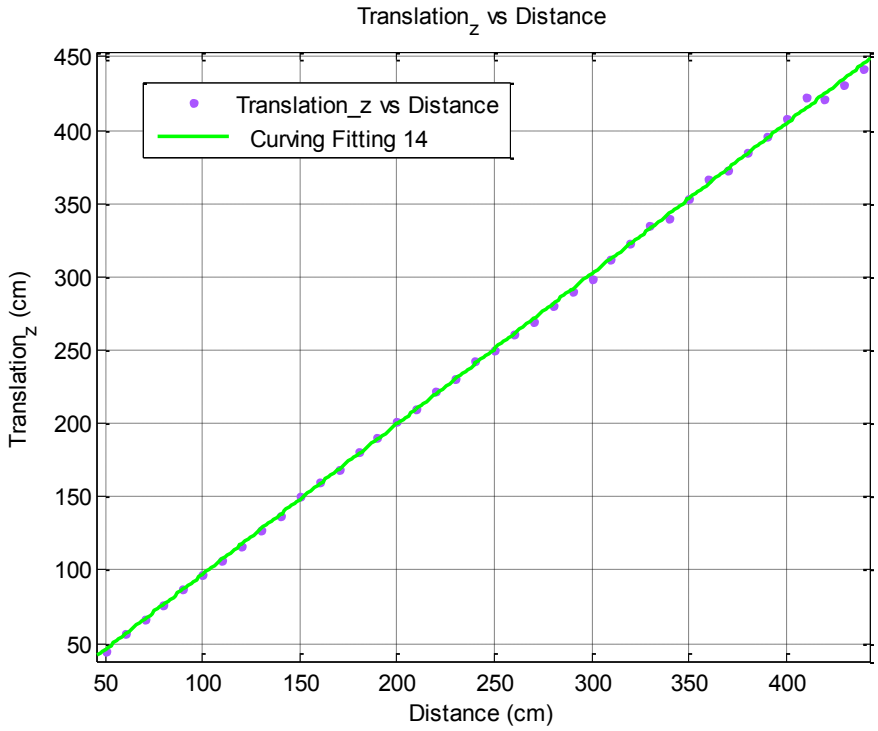


Figure 6.9: Estimated camera-marker separation versus ground truth distance.

A second measurement of error and marker stability is standard deviation. The standard deviation of marker rotation and translation was calculated for various depth values and angle changes. The results can be seen in figures 6.10. For good results we expect to see very low standard deviation values, indicating that pose estimation converges to similar values at multiple detections for a static marker.

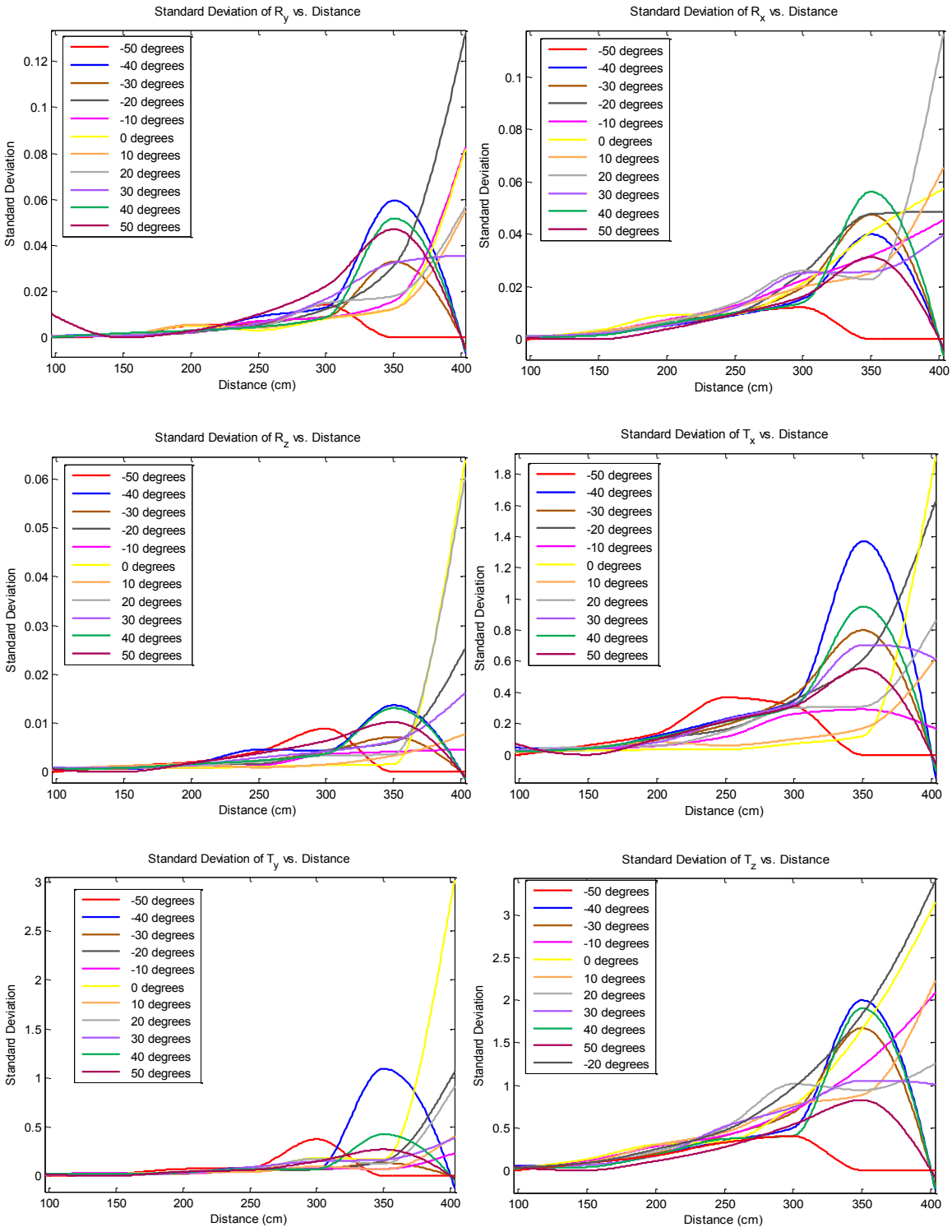


Figure 6.10: Standard deviations for pose estimation. Note that due to curve estimation techniques the sharp decreases in the curves indicate marker detection failure.

From the standard deviations of figure 6.10, a noticeable increase in error is present as the distance between the marker and the camera increases. This error growth is caused from the quantization steps of the low resolution digital image. When calculating the center coordinates of feature point elements to be used in the 3D to 3D correspondence for pose estimation, the shape of these elements are estimated only by a number colored pixels. As the marker is moved further away from camera the original shape of the elements are distorted by the perimeter pixels of detected blobs, thus cause a small amount of error in the center of mass calculation. As a result, both rotation and translation estimates yield low standard deviations and high stability for distance values up to 350 cm. In addition, it is evident that as the marker is rotated away from a zero degrees angle of incidence (where marker is perpendicular to the optical axis) the stability of pose estimation decreases.

6.6 Localization Verification

The ultimate goal of the designed fiducial marker has been stated many times throughout this thesis; it is to be implemented as a synchronization device in an odometry based localization system. This work provides two main components:

1) Pose Estimation: use of a multi-colored fiducial marker with visual image processing techniques to obtain the global position of a robotic platform in a known environment.

2) Odometry Correction: use of transform data obtained from registration of 3D images between frames to correct errors in the dead reckoning system of the device under test.

The prototype for the designed system has been constructed and tested to determine its feasibility for implementation. A sample set of the results from an executed test can be seen in figures 6.11, 6.12, and 6.13.

The validation test for localization was implemented in an environment which closely resembles an office or research laboratory. For simplicity of the test, as well as emphasizing the pose correction capability of the system, only one fiducial marker was used. This marker was placed in the scene at a height relative to the ground plane of the room. The robotic platform used for testing was then activated to autonomous wonder the room. As the platform accumulated travel distance so did the errors associated with the poor odometry system (wheel encoders). With an increase in odometry error, the precise location of the robot within the environment would be no longer known. Small adjustments to its global position were attempted to be continuously corrected for using image registration techniques between captured depth frames. However, these corrections are limited by speed of the robot and the uniqueness of feature points in depth images. The detection of the fiducial marker was used to override any global position with the one inferred from the pose estimation technique involved in the marker detection process.

Figure 6.11 depicts a plot of the ground truth position of the robot versus the estimated position, using odometry readings and position corrections for a small segment of autonomous navigation. The green circle indicated the start position of this data segment and the purple square represents the end position. The segment under analysis is composed of 4200 image frames, some of which contain a fiducial marker while others do not. The red curve of the plot represents the estimated odometry of the robot using wheel encoders, while the blue curve represents the ground truth position of the robot in the environment. Notice that the estimated position of robot tends to drift. This drift is caused from the accumulation of errors in the wheel encoders. Discontinuities in the red curve indicate position corrections incurred from marker detection.

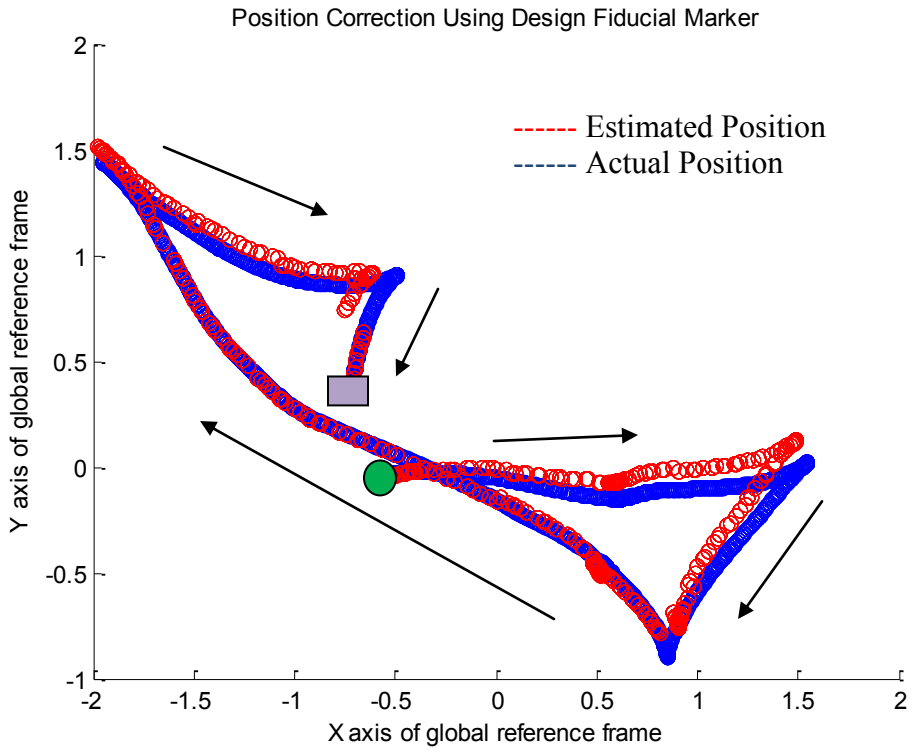


Figure 6.11: Sample localization position graph

The corresponding error analysis of figure 6.11 is displayed in chart form in figure 6.12. Because the position data used for this test has been taken from a small sample of concurrent images in a larger test case, the initial error present between the estimated position and actual position is not equal to zero. Upon first glance of the error curve, two characteristics are visible. The first being the high frequency variations in amplitude, and the second, the large amplitude drops towards zero. For this particular test the robot is configured to accumulate errors very rapidly, thus relying on both marker detection and emulated laser scan registration for position corrections. The high frequency variations are due to position corrections incurred from fake laser scan matching, as explained in chapter 5 of this work. The large amplitude drops on the other hand are due to position synchronization from marker detection, where the global position of the robot is replaced with the one obtained from the pose estimation algorithm.

A visual representation of the robot's position in the environment can be seen in figure 6.13, where the map of the environment is represented by a black and grey grid. The red points in the map indicate an emulated laser scan of what the camera is seeing, and the yellow circle outlines the robot's position. From the images of figure 6.13 and the graphs of figure 6.12 it is evident that the fiducial marker system does indeed work for indoor localization of small robotic platforms. As the error from odometry increases, marker detection and simple depth registration techniques are used to correction global position.

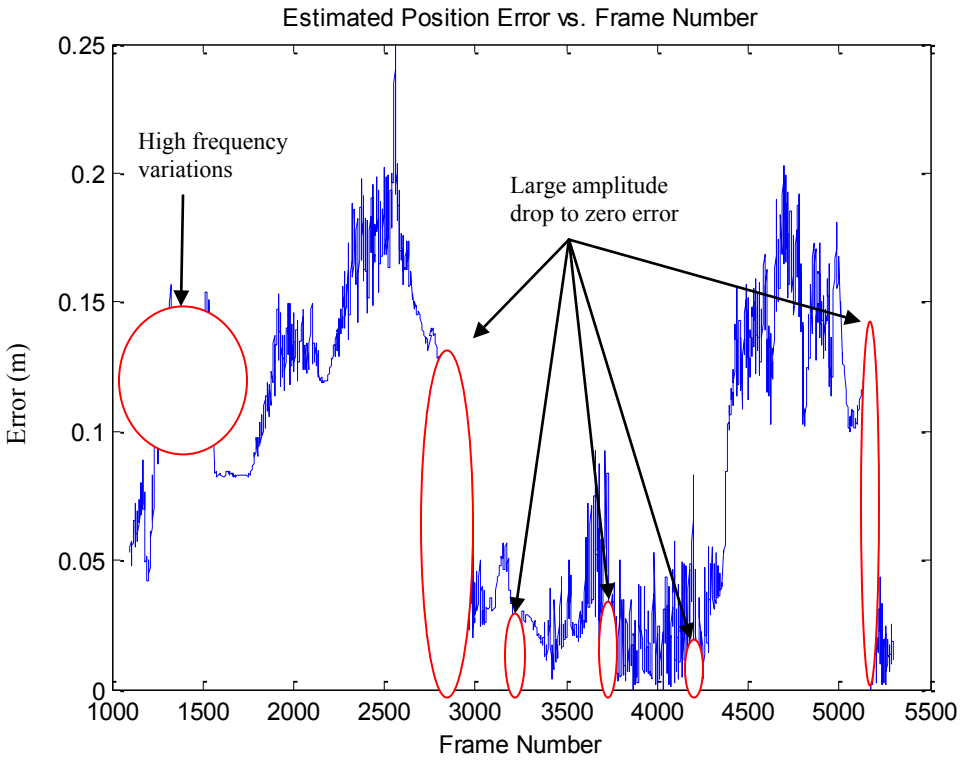


Figure 6.12: Error analysis for localization validation test

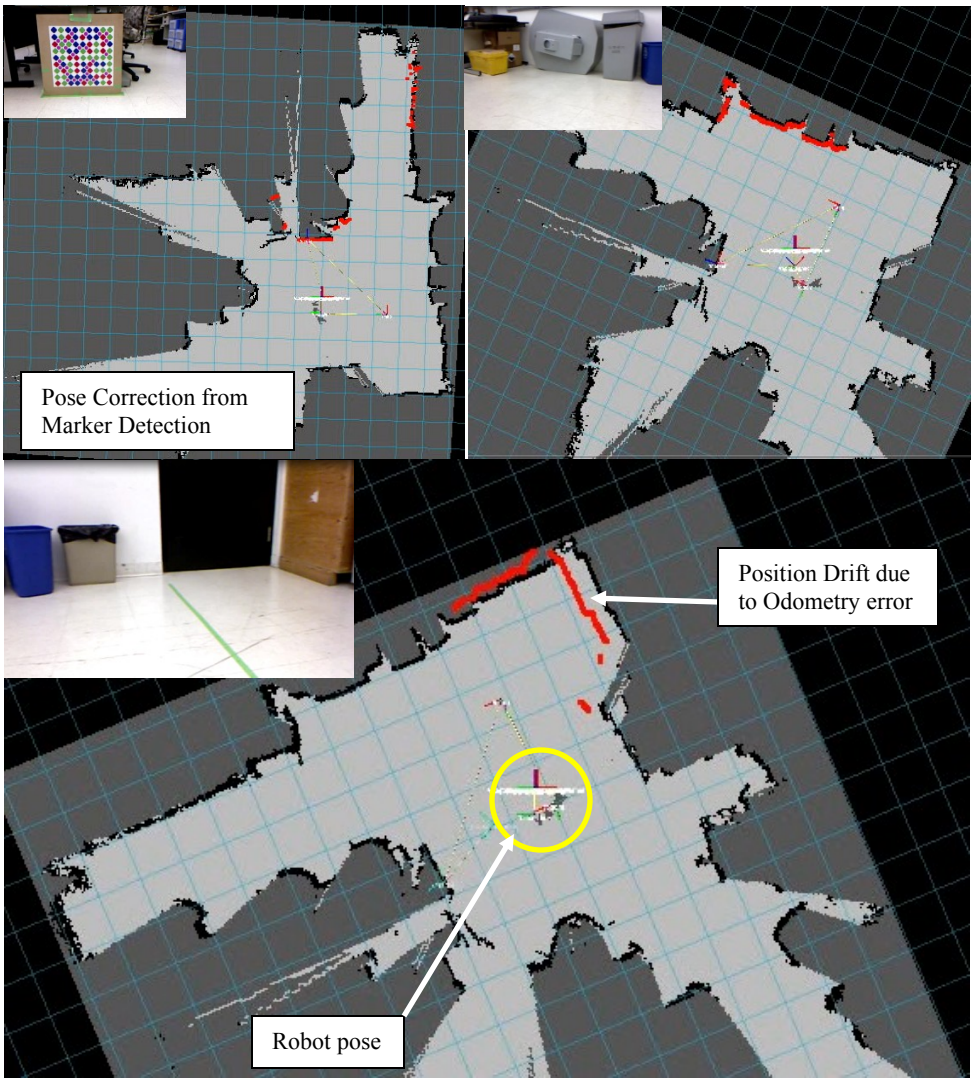


Figure 6.13: Sample images from localization test

6.7 Chapter Summary

This chapter describes a series of validation tests that are used to analyze the feasibility of using the designed fiducial marker in an odometry based indoor localization system. First, a few key attributes of the marker were examined. The false positive detection rate of the fiducial was determined to be 0.00097% by calculating the probability of randomly choosing 60 bits and having the combined

bit sequence decoded as one of the 4096 possible markers. Inter-marker confusion rates were found to be minimized by altering the generator polynomials involved in the Reed Solomon and Cyclic Redundancy Check encoding schemers for data restoration. The occlusion properties of the marker were then verified through examination of a realistic occlusion figures and a third intensive set of data was used to investigate the error and stability of pose estimation when a marker was detected. Lastly, a sample of autonomous navigation data in an indoor environment was presented, illustrating the operation of the fiducial system for localization. The series of presented validation tests proves that the designed fiducial marker system has the potential to be implemented as a position synchronization scheme for known indoor environments.

Chapter 7. Conclusion

This chapter has been divided into three sections. The first section summarizes the bulk of the material described throughout this thesis. The second section aims to emphasize the contributions resulting from the research completed is this work while the last section details future work and improvements.

7.1 Summary

The thesis begin with a review of literature that contained the full description and detection methodology for square and circular fiducial marker systems, such as ARTag [**11**], and Cantag [**21**], as well as many others. Both the advantages and disadvantages of the marker systems were identified. The techniques used to recognize these popularized fiducial markers were shown to rely on simplistic feature point detection. The inherit drawbacks associated with poor feature point selection included: high inter-marker confusion rates, high false positive detection rates, and low tolerance towards partial occlusion.

The ARToolKit [**12**] fiducial system, famed for its quick setup of time and open source coding, relied on a detection scheme based on the correlation of an inner data region. This scheme was characterized to have relatively poor detection rates with respect to inter-marker confusion and false positive detection. In the case of digital encoded fiducials, such as ARTag, marker detection rates have documented to be much higher, however, still being limited in their ability to compensate for partial occlusion. Circular fiducials on the other

hand provide accurate pose estimation, but due an inherit rotation invariance solutions are not unique. The motivation of this thesis is to adopt many techniques used in currently implement marker systems to design a robust solution towards low cost indoor localization.

An overview of the design process for a new robust pseudo-random fiducial marker is presented in chapter 3. The chapter proposed a marker that utilizes a square grid of color elements composing a pseudo-random array. This array defines 49 unique feature points at that are responsible for identifying the fiducial within a scene. It then goes on to introduce a complex and integrated data region that is used to distinguish between markers within the same library. The data region incorporates both Cyclic Redundancy Check and Reed Solomon error correcting codes to provide a maximum occlusion tolerance of 33.33% of the marker, while offering a large library size of 4096.

Chapter 4 provides an overview of the detection process for the designed fiducial marker of chapter 3. The detection algorithm made used of an optimized local adaptive thresholding technique to provide a slight robustness to illumination variations in the environment, while identifying key objects within the scene to be using in color classification. A k-means clustering method in conjunction with an HSV converted image was then employed to identify the color of these key objects. Subsequently, four k-dimensional search trees identified elements of the marker and compared to them to the pre-defined pattern of the known pseudo-random array in order to identify the marker. The

detection algorithm went even further to describe a least square solution to pose estimation which was followed by decoding of the marker's data payload.

Integration of the fiducial marker into an indoor localization system is presented in chapter 5. The marker detection and pose estimation algorithms of the fiducial alone are not enough to localize a mobile platform. Other algorithms such as floor elimination, grid occupancy map generation and basic dead reckoning are detailed. The localization scheme was described as a two step process. The first stage being an offline procedure with the intent of mapping out all static objects within the scene, as while as the locations of all fiducial markers. The second stage on the other hand, was described as a position synchronization method which used fiducial markers as a means to zero out accumulated errors from the dead reckoning system.

The experimental validation of this work is presented chapter 6, as well as the experimental setup used to collect data for this thesis. The results showed that the false positive detection probability of the designed fiducial was smaller than that of both ARTag and ARToolKit. It was also shown that by varying the generator polynomial used for Reed Solomon and Cyclic Redundancy Check encoding, lower inter-marker confusion probabilities could be obtained. The feasibility of the fiducial's use in an indoor localization system was also analyzed. The marker showed excellent results under realistic partial occlusion. It provided a maximum detection distance of about 3.5 meters with angle deviation tolerance of ± 45 degrees. Stability of the marker was also investigated illustrating standard deviations within a sub-centimeter range for distances of up to 3.5 meters.

7.2 Contributions

This thesis proposed a new form of fiducial marker that is intended to be used for integration into an indoor localization system. The marker provides 49 unique feature points that encode their absolute position within the marker's coordinate system. These feature points are then utilized, upon marker detection, to extract reliable positioning information of the imaging system with reference to the fiducial. The thesis also provides advanced algorithms for the robust detection of this marker, which are fully detailed in chapter 4.

The fiducial marker offers two digital encoding techniques, mainly Reed Solomon and Cyclic Redundancy Check, to provide a false positive probability of 0.00097% and a 33.33% tolerance to partial occlusion. The marker surpasses the designed specification of other famed fiducial markers in the research field, such as ARToolKit and ARTag, in regards to data restoration ability, inter-marker confusion rates and false position detection rates. The marker performance has been illustrated throughout a set of integrated validation tests performed in chapter 6 as well as in the already published paper [**70**] that describes the quality of the system.

The ability of a marker to be detected in environments with large dynamic objects is crucial towards artificial land-mark based localization. The proposed fiducial marker implements a pseudo-random pattern that is unlikely to be confused with random objects from within a scene, which makes it an ideal candidate as an artificial landmark for position synchronize in indoor environments. Chapter 5 establishes additional algorithms that are required to

fully integrate the proposed fiducial marker into a dead reckoning indoor localization system. Both an offline mapping stage and an online position synchronization stage are offered. Thus, this thesis provides a unique new fiducial marker and the corresponding detection algorithms such that robust and reliable position information can be recovered in less than optimal working environments.

7.3 Future Work

The algorithms presented in this thesis, although successfully providing the fundamentals for a vision-based indoor localization system, have yet to be fully tested and developed for large scale use of indoor localization. To complete this work, the system needs to be deployed on a larger scale while expanding on concepts of marker placement, and accurate map generation. Additionally, pose estimation could be improved upon by eliminating the dependence on color image processing. That is to say, future work should consider altering the design of the marker such that it is bi-tonal, black and white. This would eliminate the need for color classification and could possible aid the fiducial marker's ability to be detected in various lighting conditions. Finally, the analysis of multiple markers in the same image also needs to be address, as this could affect run-times, pose estimation and/or marker detection.

References

[1] A. Efrat and C. Gotsman, "Subpixel Image Registration Using Circular Fiducials," *Proceedings of the 2nd Israel Symposium on the Theory and Computing Systems, 1993*, pp. 49-58, June 1993.

[2] M. Kanbara, N. Yokoya, and H. Takemura, "Registration for stereo vision-based augmented reality based on extendible tracking of marker and natural features," *Proceedings. 16th International Conference on Pattern Recognition, 2002.*, vol. 2, pp. 1045-4651, December 2002.

[3] Jun. J., Q. Yue, and Z. Qing, "An Extended Marker-based Tracking System for Augmented Reality," *Second Internation Conference on Modeling, Simulation and Visualization Methods (WMSVM), 2010*, pp. 94-97, May 2010.

[4] R. Li, Z. Fang, B. Hao, and F. Yang, "Research on Indoor Wireless Localization System for Radioactive Sources Based on ZigBee," *International Conference on Computing, Control and Industrial Engineering (CCIE), 2010*, vol. 2, pp. 359-362, June 2010.

[5] P. Bahl and V. Padmanabhan, "Radar: An in building RF based user location and tracking system," *Nineteenth Annual Joint Conference on the IEEE Computer and Communixations Societies (INFOCOM), 2000*, vol. 2, pp. 775-784, April 200.

[6] J. Hightower and G. Borriello, "Particle filters for location estimation in ubiquitous computing: A case study," *Proceeding of Internation Conference on Ubiquitous Computing (UBICOMP)*, pp. 88-106, September 2004.

[7] N. Priyantha, A. Charraborty, and H. Balakrishnan, "The cricket location-support system," *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pp. 32-43, July 2000.

[8] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Transaction on Information Systems*, vol. 40, no. 1, pp. 91-102, January 1992.

[9] N. Petrellis, N. Konofaos, and G. Alexiou, "Target localization utilizing the success rate in infrared pattern recognition," *IEEE Sensors Journal*, vol. 6, no. 5, pp. 1355-1364, October 2006.

[10] F. J. MacWilliams and N. J. A. Sloane, "Pseudo-Random Sequences and Arrays," *Proceeding of the IEEE*, vol. 64, no. 12, pp. 1715-1729, December 1976.

[11] M. Fiala, "ARTag, a fiducial marker system using digital techniques," *Computer Vision and Pattern Recognition*, vol. 2, p. 590, June 2005.

[12] H. Kato, M. Billinghurst, and I. Poupyrev, "ARToolKit," Human Interface Technology Lab, University of Washington, 2000.

[13] H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," *Augmented Reality, 1999. (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop*, pp. 85-94, Oct. 1999.

[14] M. Fiala, "ARTag, an improved marker system based on ARToolkit," *NRC Institute for Information Technology*, vol. NRC: 47419, 2004.

[15] J. Sattar, E. Bourque, P. Giguere, and G. Dudek, "Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction," *Computer and Robotic Vision*, pp. 165-174, June 2007.

[16] C.B. Owen, Fan Xiao, and P. Middlin, "What is the best fiducial?," *Augmented Reality Toolkit, The First IEEE International Workshop*, p. 8, January 2003.

[17] X. Zhang, S. Fronz, and N. Navab, "Visual Marker Detection and Decoding in AR Systems: A Comparative Study," *Mixed and Augmented Reality, 2002. ISMAR 2002*, pp. 97-106, January 2003.

[18] E. Olson, "AprilTag: A robust and flexible visual fiducial system," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2011*, May 2011.

[19] J. Rekimoto and Y. Ayatsuka, "CyberCode: Designing Augmented Reality Environments with Visual Tags," *Proceedings of DARE 2000 on Designing augmented reality environments*, pp. 1-10, 2000.

[20] D. Varodayan and K. Ghosh, "Visual Code Marker Detection," Stanford Univeristy, 2006.

[21] A.C. Rice, A.R. Beresford, and R.K. Harle, "Cantag: an open source software toolkit for designing and deploying marker-based vision systems," *Fourth Annual IEEE International Conference on Pervasive Computing and Communications, 2006*, pp. 10-21, March 2006.

[22] L. Naimark and E. Foxlin, "Circular Data Matrix Fiducial System and Robust Image Processing for a Wearable Vision-Inertial Self Tracker," *IEEE International Sympsium on Mixed and Augmented Reality (ISMAR 2002)*, pp. 27-36, 2002.

[23] S. Shih and T. Yu, "On Designing an Isotropic Fiducial Mark," *IEEE Transactions on Image Processing*, vol. 12, no. 9, pp. 1054-1066, September 2003.

[24] Y. Cho and U. Neumann, "Multi-Ring Color Fiducial Systems for Scalable Fiducial Tracking Augmented Reality," *Virtual Reality Annual International Symposium*, p. 212, March 1998.

[25] F. Bergamasco, A. Albarelli, E. Rodola, and A. Torsello, "RUNE-Tag: a High Accuracy Fiducial Marker with Strong Occlusion Resilience," *IEEE International Conference on Computer Vision and Pattern Recognigion (CVPR2011)*, to appear in 2011.

[26] M. Hirzer. (2008) Marker Detection for Augmented Reality Application. [Online]. http://studierstube.icg.tugraz.at/thesis/marker_detection.pdf

[27] G. Bradski and A. Kaehler, *Learning OpenCV*, 1st ed., Mike Loukides, Ed. Gravenstein Highway North, Sebastopol, CA, United States: O'Reilly Media, 2008.

[28] P. Payeur, "Coordinate Systems and Transformations", 2010, Class Notes: ELG5163 - University of Ottawa.

[29] R.I. Hartley, "In defence of the 8-point algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 580-593, 1997.

[30] P. Rousseeuw, "Least Median of Squares Regression," *Journal of the American Statistical Association*, vol. 79, no. 388, p. 871, December 1984.

[31] M.P. Kumar, S. Kuthirumunal, C.V. Jawahar, and P.J. Narayanan, "Planar

homography from fourier domain representation," *International Conference on Signal Processing and Communications, 2004. SPCOM '04*, pp. 560-564, December 2004.

[32] M. Lourakis. (2010, July) homest: A C/C++ Library for Robust, Non-linear Homography Estimation. [Online]. http://www.ics.forth.gr/~lourakis/homest/

[33] Y. Chen, J. Sun, and G. Wang, "Minimizing Geometric Distance by Iterative Linear Optimization," *20th International Conference on Pattern Recognition (ICPR)*, pp. 1-4, August 2010.

[34] Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330-1334, November 2000.

[35] H. Lim and Y. Lee, "Real-Time Single Camera SLAM Using Fiducial Markers," *ICROS-SICE International Joint Conference*, pp. 177-182, August 2009.

[36] H. Lim, "Simultaneous Localization and Mapping (SLAM) of Mobile Robot Using a Single Camera," Department of Electrical Engineering, Inha University, Master's Thesis 2010.

[37] A. Mulloni, D. Wagner, I. Barakonyi, and D. Schmalstieg, "Indoor Positioning and Navigation with Camera Phones," *IEEE Pervasive Computing*, vol. 8, no. 2, pp. 22-31, April 2009.

[38] D. Wagner and D. Schmalstieg, "ARToolKitPlus for Pose Tracking on Mobile Devices," *Proc. 12th Computer Vision Winter Workshop*, February 2007.

[39] M. Kaltenbrunner and R. Bencina, "reacTIVision: a computer-vision framework for table-based tangible interaction," *Proc. of the 1st International Conference on Tangible and Embedded Interaction*, pp. 69-74, 2007.

[40] J. Kohler, A. Pagani, and D. Stricker, "Detection and Identification Techniques for Markers Used in Computer Vision," *Visualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling and Engineering (IRTG 1131 Workshop)*, vol. 19, pp. 36-44, 2011.

[41] G. Schweighofer and A. Pinz, "Robust Pose Estimation from a Planar Target," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 2024-2030, December 2006.

[42] E. M. Petriu, T. Bieseman, N. Trif, W. S. McMath, and S. K. Yeung, "Visual Object Recognition Using Pseudo-Random Grid Encoding," *Proceedings of the 1992 International Conference on Intelligent Robots and Systems*, vol. 3, pp. 1617-1624, July 1992.

[43] J. W. Spoelder, F. M. Vos, E. M. Petriu, and F. C. A. Croen, "A study of the robustness of pseudorandom binary-array-based surface characterization," *IEEE Transactions on Instrumentation and Measurement*, vol. 47, no. 4, pp. 833-838, August 1998.

[44] E. M. Petriu and J. S. Basran, "On the position measurement of automated guided vehicles using pseudorandom encoding," *Instrumentation and Measurement, IEEE Transactions*, vol. 38, no. 3, pp. 799-803, June 1989.

[45] E. M. Petriu, "Absolute position recovery for automated path-guided vehicles,"

presented at ROBOTS 12 and VISION '88 Conf. Exibit, June 1988.

[46] E. M. Petriu, "Absolute-type position transducers using a pseudorandom encoding," *IEEE Trans. Instrumentation Measurements*, vol. 1, no. 36, pp. 950-955, Dec. 1987.

[47] E. M. Petriu, W. S. McMath, S. K. Yeung, N. Trif, and T. Bieseman, "Two-dimensional position recovery for a free-ranging automated guided vehicle," *IEEE Trans. Instrumentation and Measurement*, vol. 42, no. 3, pp. 701-706, 1993.

[48] A. Kazantsev and E. M. Petriu, "Robust pseudo-random coded colored structured light technique for 3D object model recovery," *IEEE Int'l Workshop on Robotic and Sensors Environments*, pp. 150-155, 2008.

[49] P. Payeur and D. Desjardins, "Structured light stereoscopic imaging with dynamic pseudo-random patterns," *Proc. of the International Conference on Image Analysis and Recognition*, pp. 687-696, 2009.

[50] R.A. Morano, C. Ozturk, R. Conn, S. Dubin, S. Zietz and J. Nissano, "Structured light using pseudorandom codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 322-327, March 1998.

[51] J. Salvi, J. Pages, and J. Batlle, "Pattern codification strategies in structured light systems," *Pattern Recognition*, vol. 37, pp. 827-849, 2004.

[52] I. Kim, S. Kim, S. Yu, and S. Lee, "Robust 3D Face Data Acquisition Using a Sequential Color-Coded Pattern and Stereo Camera System," *Computational Science and Its Application (ICCSA 2006)*, vol. 3981, pp. 87-95, 2006.

[53] G Kim, "Designing Robust Fiduciary Markers for Indoor Localization," University of Ottawa, Ottawa, Unpublished Master's Thesis 2010.

[54] M. Fiala, "Comparing ARTag and ARToolkit Plus Fiducial Marker Systems," *IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, pp. 148-153, October 2005.

[55] P. Sutheebanjard and W. Premchaiswadi, "QR-code generator," *8th Internation Conference of ICT and Knowledge Engineering*, pp. 89-92, november 2010.

[56] M. Querini, A. Grillo, A. Lentini, and G. F. Italiano, "2D Color Barcodes for Mobile Phones," *International Journal of Computer Science and Applications*, vol. 8, no. 1, pp. 136-155, 2011.

[57] P. Koopman and T. Chakravvarty, "Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks," *Internation Conference on Dependable Systems and Networks*, p. 145, 2004.

[58] B. Sklar, *Digital Communications: Fundamentals and Applications 2ed*. Upper Saddle River, United States: Prentice Hall, 2001.

[59] R. G. Gallager, *Information Theory and Reliable Communication*. New York, United States: John Wiley and Sons, 1968.

[60] R. E. Blahut, *Theory and Practice of Error Control Codes*. Reading, United States: Addison-Wesley, 1983.

[61] T. Layvand, C. Meekhof, Y. Wei, J. Sun, and B. Guo, "Kinect Identity: Technology and Experience," *Computer*, vol. 44, no. 4, pp. 94-96, April 2011.

[62] F. Shafait, D. Keysers, and T. M. Breuel, "Efficient Implementation of Local Adaptive Thresholding," *Document Recognition and Retrieval XV*, vol. 6815,

January 2008.

[63] J. S. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," *Conference on Computer Vision and Pattern Recognition*, pp. 1000-1006, June 1997.

[64] M. Muja and D. Lowe, "Fast approximate nearest neighbor with automatic algorithm configuration," *In VISAPP International Conference on Computer Vision Theory and Applications*, pp. 331-340, 2009.

[65] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-Squares Fitting of Two 3-D Point Sets," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-9, no. 5, pp. 698-700, September 1987.

[66] T. S. Huang, S. D. Blostein, and E. A. Margerum, "Least-squares estimation of motion parameters from 3-D point correspondences," *Proc. IEEE Conference in Computer Vision and Pattern Recognition*, June 1986.

[67] UCSB Robotics Lab. (2011, October) UC Santa Barbara Repository for ROS packages. [Online]. http://code.google.com/p/ucsb-ros-pkg/

[68] K. Conley. (2011, September) ROS.org. [Online]. http://www.ros.org/wiki/

[69] M. Fiala, "Designing Highly Reliable Fiducial Markers," *IEEE Transcations on pattern analysis and machine intelligence*, vol. 32, no. 7, p. 1317, July 2010.

[70] A. Stathakis and E. M. Petriu, "Robust Pseudo-Random Fiducial marker for Indoor Localization," *IEEE Int. Symp. on Robotic and Sensor Environments, in Proc. ROSE 2011*, pp. 19-24, September 2011.

[71] J. J. Rushanan. (1996, December) Northeastern University. [Online]. www.ccs.neu.edu/home/jjr/tutor.ps

[72] Dr. Thamer. (2011, October) University of Technology - IRAQ. [Online]. http://www.uotechnology.edu.iq/dep-eee/lectures/4th/Communication/Information%20theory/8.pdf

[73] J. Rekimoto, "Matrix: a realtime object identification and registration method for augmented reality," *Computer Human Interaction, 1998. Proceedings. 3rd Asia Pacific* , pp. 63-68, July 1998.

[74] D. Flohr and J. Fischer, "A Lightweight ID-Based Extension for Marker Tracking Systems," *Proc. Eurographics Symp. Virtual Environments*, pp. 147-152, May 2007.

[75] X. Zhong, P. Liu, N. D. Georganas, and P. Boulanger, "Designing a Vision-based Collaborative Augmented Reality Application for Industrial Training," *it - Information Technology*, pp. 7-19, 2003.

Appendix A. Singular Value Decomposition

Singular Value Decomposition (SVD) is a way to factorize matrices, where any real or complex $m \times n$ matrix $\boldsymbol{A}$ can be decomposed as such:

$$\boldsymbol{A} = U\Sigma V^T$$

U is a $m \times n$ orthogonal matrix ($U^T U = \boldsymbol{I}$). The columns of U are eigenvectors of $\boldsymbol{AA^T}$.

V is a $n \times n$ orthogonal matrix ($V^T V = \boldsymbol{I}$). The columns of V are eigenvectors of $\boldsymbol{A^T A}$.

D is a $n \times n$ diagonal matrix. Any non-negative real values are called singular values.

$$D = diag(\sigma_1, \sigma_2, \dots, \sigma_n)$$

SVD is used when a matrix, A, has a matrix of eigenvectors that is not invertible. That is to say that A does not have a eigen decomposition. There are many applications towards using SVD, some of which are: calculating the pseudo-inverse of a matrix, solving linear equations, and computing a total least squares minimization.

Appendix B. Homography Estimation

The homography matrix, $\boldsymbol{H}$, provides a direct mapping between points in two planes. Let $\boldsymbol{x}_2$be a point in the second plane while $\boldsymbol{x}_2$be a point on the first plane. Thus a relation between the two points can be expressed as follows:

$$\boldsymbol{x_2} = H\boldsymbol{x_1} \tag{B1}$$

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = H \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \tag{B2}$$

Converting the equation to homogeneous coordinates we obtain:

$$x'_2 = \frac{x_2}{z_2}, \quad y'_2 = \frac{y_2}{z_2}, \quad z_1 = 1 \tag{B3}$$

$$\begin{bmatrix} x'_2 \\ y'_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{31} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{B4}$$

Expansion of A4 using A3 yields:

$$x'_2 = \frac{h_{11}x_1 + h_{12}y_1 + h_{13}z_1}{h_{31}x_1 + h_{32}y_1 + h_{33}z_1} \tag{B5}$$

$$y'_2 = \frac{h_{21}x_1 + h_{22}y_1 + h_{23}z_1}{h_{31}x_1 + h_{32}y_1 + h_{33}z_1} \tag{B6}$$

Rearranging A5 and A6 into the following form $\boldsymbol{a_x^T h} = 0, \boldsymbol{a_x^T h} = 0$.

$$(-x_1, -y_1, -1, 0, 0, 0, x'_2 x_1, x'_2 y_1, x'_2)\boldsymbol{h} = 0 \tag{B7}$$

$$(0, 0, -1, -x_1, -y_1, -1, y'_2 x_1, y'_2 y_1, y'_2)\boldsymbol{h} = 0 \tag{B8}$$

where,

$$\boldsymbol{h} = (h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33})^T \tag{B9}$$

$$\boldsymbol{a_x^T} = (-x_1, -y_1, -1, 0, 0, 0, x'_2 x_1, x'_2 y_1, x'_2)^T \tag{B10}$$

$$\boldsymbol{a_y^T} = (0, 0, -1, -x_1, -y_1, -1, y'_2 x_1, y'_2 y_1, y'_2)^T \tag{B11}$$

A system of linear equations is formed:

$$\boldsymbol{Ah} = 0 \qquad (B12)$$

$$\boldsymbol{A} = \left(a_{x1}^T, a_{y1}^T \ldots\ a_{xN}^T, a_{yN}^T\right) \qquad (B13)$$

This system is solved using homogeneous linear least squares. More specifically a solution is obtained by computing the Singular Value Decomposition (SVD) of $\boldsymbol{A}$.

$$A = U\Sigma V^T = \sum_{i=1}^{9} \sigma_i u_i {v_i}^T \qquad (B14)$$

The smallest singular value, σ_9, will identify if the homography is exactly determined, underdetermined or over determined. There are three cases for the value of σ_9.

Case 1: $\sigma_9 = 0$, then there exists a homography that fit the points exactly.

Case 2: $\sigma_9 \geq 0$, then the homography is over determined and the singular value represents a goodness of fit.

Case 3: the homography is undetermined.

From the SVD the smallest singular vector, right column of V, corresponds to σ_9. This column is the solution for $\boldsymbol{h}$**,** which contains the elements of the homography matrix that best suits the points.

Appendix C. Finite Fields

A finite field is a set of numbers that are closed under the following operations:

- Addition
- Subtraction
- Multiplication
- Division by non-zero elements

Mainly, a field has two binary operations, '+' (addition) and '·' (multiplication) that are both associative and commutative. These two operations have inverse operations '-' (subtraction) and '/' (division). Lastly, there exists a distributive law that relates the addition and multiplication operations:

$$a(b + c) = ab + ac \qquad (C1)$$

A few examples of fields are the rational set of numbers **Q**, the real set of numbers **R,** and the complex set of numbers **C**. Fields may also have subfields which are subsets of a field. Subfields must also obey the general properties of a field. Some subfields may be finite in size, unlike **Q**, **R** and **C**.

From [**71**] any finite field, $\mathbf{Z_p}$, must contain the integers set, **I,** modulo a prime number, *p.* A finite field implies that there must be some number *p* such that adding one to itself *p* time will result in zero. The term *p* is denoted the characteristic of the field. Any finite field of size *q* is label a Galois Field (GF(*q*)). A Galois Field exits if and only if *q* is a positive integer of a prime number, in other words a prime power. Equivalently, GF(q) is can be expressed as GF(p^n). It

is very common use a value of *p* equal to 2, as in the construction of Reed Solomon codes, although other values can be chosen.

The field GF(2), or $\mathbf{Z_2}$, is known as the binary field. The binary field is a subfield of the Galois field, **K** = GF(2^m). Galois Fields not only contain elements 0 and 1, but a unique element denoted by the symbol β. Every element of the field **K** can be expressed as a power of the symbol β, with the exception of the element zero. The range of elements associate with **K** can be written as follows:

$$\boldsymbol{K} = GF(2^m) = \left\{0, \beta^0, \beta^1, \dots, \beta^{2^m-2}\right\} \qquad (C2)$$

The field **K** can also be written in polynomial form with degree *m* – 1. Every non-zero element that lies in **K** is a polynomial, $\beta^i = b_i(x)$. Thus for $i = 0, 1, \dots, 2^m - 2$,

$$\beta^i = b_i(x) = b_{i,0} + b_{i,1}x + b_{i,2}x^2 + \dots + b_{i,m-1}x^{m-1} \qquad (C3)$$

Each of the b_i coefficients in C3 is assigned one of two values, 0 or 1. This comes from the fact that we have set the parameter *p* equal to 2. It is often convenient to express any field element β^i as a binary sequence by taking only the coefficient terms of the polynomial.

Once in polynomial form, addition of two elements of the field is simply defined as the modulo-2 sum, or more commonly known as the 'XOR' operation when dealing with binary sequences.

$$\beta^i + \beta^j = \left(b_{i,0} \oplus b_{j,0}\right) + \left(b_{i,1} \oplus b_{j,1}\right)x + \cdots + \left(b_{i,m-1} \oplus b_{j,m-1}\right)x^{m-1} \qquad (C4)$$

where $\oplus$ indicates an XOR operations between the two operands. For example say we wish to add the two polynomials $x^4 + x^2 + 1$ and $x^4 + x^1 + 1$ in GF(2^5). A simply way of doing so would be to first convert them to a binary sequence,

10101 and 10011, respectively. Then perform a modulo-2 sum of the two sequences and convert them back to polynomial form:

$$10101 \oplus 10011 = 00110 \qquad binary\ conversion \qquad (C5)$$

$$(x^3 + x^1 + 1) \oplus (x^3 + x^0 + 1) = x^1 + x^0 + 1 \qquad polynomial\ addition \quad (C6)$$

Finite fields can be defined using primitive polynomials. Consider an irreducible polynomial ***g(x)***, which is of degree m, and an integer value $n = 2^m - 1$. As stated in [**72**], ***g(x)*** is a primitive polynomial if and only if *n* is the smallest positive integer value for which ***g(x)*** evenly divides (non-zero quotient and zero remainder) $x^n + 1$. A few primitive polynomials are defined in Table C1.

***m* value**	**Primitive Polynomial**
3	$x^3 + x^1 + 1$
4	$x^4 + x^1 + 1$
5	$x^5 + x^2 + 1$
6	$x^6 + x^1 + 1$
7	$x^7 + x^3 + 1$
8	$x^8 + x^4 + x^3 + x^2 + 1$
9	$x^9 + x^4 + 1$

Table C1: Examples of various primitive polynomials

Non-zero elements of finite fields are defined using these primitive polynomials. One of the *m* roots of the primitive polynomial defines at least one element contained within $GF(2^m)$. The using this first element we can define the remaining elements. Let us consider the following example using the field $GF(2^m)$ where m = 4, and the primitive polynomial $\pi(x) = x^4 + x^1 + 1$.

1) The first step is to set *π(β) = 0.* This is brought by assuming the primitive element, β, of the field is a root of the primitive polynomial.

$$\pi(\beta) = \beta^4 + \beta + 1 = 0 \quad (C7)$$

$$\beta^4 = \beta - 1 = \beta + 1 \quad (C8)$$

Equation C8 comes from the fact addition and subtraction are equivalent when using modulo-2 base addition.

2) Next the identity of equation C8 is used repeatedly to form the remaining polynomials of GF(2^4).

$$\beta^5 = \beta^4\beta = (\beta + 1)\beta = \beta + \beta^2 \quad (C9)$$

$$\beta^6 = \beta^4\beta^2 = (\beta + 1)\beta^2 = \beta^3 + \beta^2 \quad (C10)$$

$$\beta^7 = \beta^4\beta^3 = (\beta + 1)\beta^3 = \beta^4 + \beta^3 = 1 + \beta + \beta^3 \quad (C11)$$

$$\beta^8 = \beta^6\beta^2 = (\beta^2 + \beta^3)(\beta^2) = \beta^5 + \beta^4 = 1 + \beta + \beta + \beta^2 = 1 + \beta^2 \quad (C12)$$

$$\beta^9 = \beta^8\beta^1 = (\beta^2 + 1)(\beta) = \beta^3 + \beta \quad (C13)$$

$$\beta^{10} = \beta^9\beta^1 = (\beta^3 + \beta)(\beta) = \beta^2 + \beta^4 = 1 + \beta + \beta^2 \quad (C14)$$

$$\beta^{11} = \beta^{10}\beta^1 = (1 + \beta + \beta^2)(\beta) = \beta^3 + \beta^2 + \beta \quad (C15)$$

$$\beta^{12} = \beta^{11}\beta^2 = (\beta + \beta^2 + \beta^3)(\beta) = \beta^5 + \beta^4 + \beta^3 = 1 + \beta + \beta^2 + \beta^3 \quad (C16)$$

$$\beta^{13} = \beta^{12}\beta^1 = (1 + \beta + \beta^2 + \beta^3)(\beta) = \beta^3 + \beta^2 + 1 \quad (C17)$$

$$\beta^{14} = \beta^{13}\beta^1 = (\beta^3 + \beta^2 + 1\,)(\beta) = \beta^3 + 1 \quad (C18)$$

To multiply two element in the field, it is as simple as adding their exponents and using the fact that $\beta^{15} = 1$ (ie. $\beta^3\beta^{14} = \beta^{17} = \beta^2$). Division is reduced to multiplication of the multiplicative inverse, β^{15-j} of β^j (ie. $\beta^3 / \beta^{14} = \beta^3\,\beta^1 = \beta^4$).

Appendix D. Hamming Distances of Codewords

HD	RS1/CRC0	RS1/CRC1	RS1/CRC2	RS1/CRC3	RS1/CRC4	RS1/CRC5
15	12288	8192	0	4096	8192	12288
16	8192	0	0	4096	0	0
17	8192	0	12288	0	4096	12288
18	4096	8192	0	8192	16384	4096
19	16384	49152	45056	28672	32768	12288
20	106496	73728	81920	90112	65536	77824
21	143360	155648	172032	131072	126976	131072
22	311296	282624	274432	278528	237568	307200
23	397312	405504	401408	450560	425984	446464
24	663552	716800	655360	692224	765952	733184
25	974848	925696	897024	909312	937984	847872
26	966656	839680	1060864	913408	892928	868352
27	1261568	1257472	1224704	1277952	1392640	1318912
28	1798144	1835008	1671168	1835008	1818624	1826816
29	1753088	1789952	1851392	1867776	1703936	1888256
30	1515520	1744896	1785856	1560576	1630208	1589248
31	1835008	1744896	1785856	1683456	1662976	1691648
32	1507328	1544192	1314816	1609728	1593344	1560576
33	1015808	1032192	987136	1015808	1069056	1036288
34	671744	622592	933888	724992	696320	700416
35	647168	655360	626688	659456	606208	634880
36	569344	516096	434176	446464	442368	487424
37	299008	278528	270336	262144	331776	270336
38	176128	167936	135168	176128	196608	200704
39	28672	69632	106496	90112	65536	77824
40	61440	28672	24576	32768	28672	28672
41	0	8192	4096	8192	20480	4096
42	12288	4096	4096	8192	0	0
43	8192	4096	4096	0	0	0
44	0	0	8192	4096	0	0
45	0	4096	0	0	0	4096

Figure D1: Frequency of Hamming Distance values between marker codeword pairs. The values in the chart express hamming distance frequencies when keeping the Reed Solomon generator polynomial the same and altering the CRC generator polynomial.

HD	RS2/CRC0	RS2/CRC1	RS2/CRC2	RS2/CRC3	RS2/CRC4	RS2/CRC5
15	4096	0	0	0	8192	4096
16	0	0	0	0	0	0
17	4096	4096	12288	4096	0	0
18	0	4096	16384	8192	12288	8192
19	40960	32768	49152	36864	24576	12288
20	102400	57344	90112	65536	65536	61440
21	180224	110592	143360	147456	147456	139264
22	286720	286720	233472	266240	323584	225280
23	417792	491520	413696	462848	434176	532480
24	729088	778240	753664	745472	659456	811008
25	872448	847872	872448	921600	905216	872448
26	921600	933888	884736	901120	876544	909312
27	1363968	1314816	1327104	1236992	1396736	1298432
28	1732608	1744896	1732608	1761280	1806336	1757184
29	1802240	1929216	1843200	1785856	1761280	1785856
30	1576960	1609728	1662976	1708032	1642496	1699840
31	1581056	1609728	1691648	1777664	1601536	1638400
32	1605632	1572864	1658880	1589248	1695744	1548288
33	1056768	1015808	1015808	995328	1060864	1085440
34	733184	679936	737280	643072	663552	667648
35	675840	667648	634880	593920	659456	638976
36	507904	503808	434176	512000	438272	491520
37	258048	270336	286720	327680	299008	282624
38	151552	151552	135168	139264	147456	155648
39	110592	77824	73728	86016	69632	69632
40	36864	53248	40960	36864	49152	45056
41	16384	16384	16384	12288	16384	24576
42	0	4096	0	4096	4096	4096
43	0	0	4096	0	0	0
44	0	4096	4096	4096	0	0
45	4096	0	4096	0	4096	4096

Figure D2: Frequency of Hamming Distance values between marker codeword pairs. The values in the chart express hamming distance frequencies when keeping the Reed Solomon generator polynomial the same and altering the CRC generator polynomial. The highlighted region indicates the lowest hamming distance values of any other codeword combination.