Självständigt arbete på avancerad nivå
Independent degree project – second cycle

Electronics Design
30 Credits

Calibration of prepared environment for optical navigation

Jinu Panilet Panipichai
Calibration of prepared environment for optical navigation
Jinu Panilet Panipichai

MID SWEDEN UNIVERSITY
Name of division
Examiner: Bengt Oelmann, Bengt.Oelmann@miun.se
Supervisor: Benny Thörnberg, Benny.Thornberg@miun.se
Author: Jinu Panilet Panipichai, jipa1000@student.miun.se
Degree programme: Masters Programme in Electronics Design, 120 credits
Main field of study: Machine Vision System (Image processing)
Semester, year: HT, 2014
Abstract

The main objective of this thesis is to evaluate accuracy and precision of the machine vision system used to calibrate a prepared environment for optical navigation. Rotationally independent optimized colour reference labels (symbols) creates an environment. Any number of symbols can be used. A symbol carries 8–bit (0 to 255) information, which can be designed for different values by using Matlab algorithms. An optical navigation system enters into the environment and captures the symbols. The symbols are then decoded to determine the geographical positions of the symbols from reference position of the system by using Matlab algorithms. Then, the system is moved to a known position and the same set of symbols are captured, decoded and located. The process is repeated for several positions of the system to find precision and accuracy. Finally, the results are analysed.

Keywords: Optimized colour symbols, Precision, Accuracy, SNR.
Foreword

I would to take this opportunity to extend my gratitude towards my thesis supervisor, Benny Thörnberg for his encouragement, support and guidance throughout my thesis work. I would like to thank Qaiser Anwar for helping me throughout my thesis. I would also like to thank my family and friends for their support, encouragement and motivation.
Table of Contents

Abstract .................................................................................................................. iii
Foreword ............................................................................................................... iv
Table of Contents .............................................................................................. v
Table of Figures ................................................................................................... vii
List of Tables ...................................................................................................... viii
Terminology / Notation ..................................................................................... ix
Acronyms ........................................................................................................... ix

1 Introduction ..................................................................................................... 3
1.1 Background .................................................................................................... 3
1.2 Overall aim ..................................................................................................... 3
1.3 Contributions ................................................................................................ 4
1.4 Verifiable goals .............................................................................................. 5
1.5 Outline ........................................................................................................... 5

2 Related work .................................................................................................. 7

3 Theory .............................................................................................................. 11
3.1 Colours in the symbol .................................................................................. 11
3.2 Imaging Geometry ....................................................................................... 12
3.2.1 Perspective projection ............................................................................ 12
3.2.2 Translation ............................................................................................. 14
3.2.3 Rotation ................................................................................................. 15
3.2.4 Environment calibration ........................................................................ 15

4 Methodology / Model ..................................................................................... 17
4.1 System used .................................................................................................. 17
4.2 Experimental Procedure ............................................................................. 18
4.3 Algorithm ..................................................................................................... 19

5 Implementation ............................................................................................... 21
5.1 Symbols design ............................................................................................ 21
5.2 Symbol detection ......................................................................................... 22
5.3 Camera rotation ........................................................................................... 23
5.4 Symbol decoding ........................................................................................ 23
5.5 From pixel co-ordinates to image plane co-ordinates ............................... 24
Calibration of prepared environment for optical navigation
Jinnu Panilet Panipichai

Table of Contents

5.6 Location of dots in world co-ordinates .................................................... 25
5.7 Illustration .................................................................................................. 28

6 Result ........................................................................................................... 33
6.1 Precision .................................................................................................... 33
6.2 Accuracy ..................................................................................................... 35
6.3 SNR ............................................................................................................ 37

7 Discussion .................................................................................................... 39
7.1 Disadvantages of the previous work eliminated ........................................ 39
7.2 Minimum and maximum $Z_0$ .................................................................. 39
7.3 Disadvantage ............................................................................................. 39
7.4 Comparison with VSLAM .......................................................................... 40
7.5 Future work ............................................................................................... 41
7.6 Effects of this project on ethics .................................................................. 41
7.7 Effects of this project on society ................................................................. 42

8 Conclusion ................................................................................................... 43

References ....................................................................................................... 45

Appendix: Documentation of own developed program code ......................... 47
### Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Steps involved in environment calibration</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>Mars Reconnaissance Orbiter</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Mars’ moons – Phobos and Deimos</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>Optical Navigation Demonstration near Mars</td>
<td>9</td>
</tr>
<tr>
<td>3.1</td>
<td>Perspective projection</td>
<td>13</td>
</tr>
<tr>
<td>3.2</td>
<td>Translation from real world to image plane coordinate system</td>
<td>14</td>
</tr>
<tr>
<td>4.1</td>
<td>Experimental setup</td>
<td>17</td>
</tr>
<tr>
<td>4.2</td>
<td>Camera used</td>
<td>18</td>
</tr>
<tr>
<td>4.3</td>
<td>Accuracy Measurement at constant $Z_0$</td>
<td>19</td>
</tr>
<tr>
<td>5.1</td>
<td>Symbol representing the code - 138</td>
<td>21</td>
</tr>
<tr>
<td>5.2</td>
<td>Background elimination</td>
<td>22</td>
</tr>
<tr>
<td>5.3</td>
<td>Symbol and dots detection</td>
<td>22</td>
</tr>
<tr>
<td>5.4</td>
<td>Big Dots</td>
<td>23</td>
</tr>
<tr>
<td>5.5</td>
<td>Image acquired at reference position</td>
<td>29</td>
</tr>
<tr>
<td>5.6</td>
<td>Output image showing location of dots</td>
<td>30</td>
</tr>
<tr>
<td>5.7</td>
<td>Matlab output - location of dots in real world coordinates</td>
<td>30</td>
</tr>
<tr>
<td>5.8</td>
<td>Input image for 2 symbols</td>
<td>31</td>
</tr>
<tr>
<td>5.9</td>
<td>Output for the Matlab script</td>
<td>31</td>
</tr>
<tr>
<td>6.1</td>
<td>Standard deviations in World coordinates (X-axis)</td>
<td>34</td>
</tr>
<tr>
<td>6.2</td>
<td>Standard deviations in World coordinates (Y-axis)</td>
<td>34</td>
</tr>
<tr>
<td>6.3</td>
<td>Standard deviations in Image plane coordinates (X-axis)</td>
<td>35</td>
</tr>
<tr>
<td>6.4</td>
<td>Standard deviations in Image plane coordinates (Y-axis)</td>
<td>35</td>
</tr>
<tr>
<td>6.5</td>
<td>Different camera positions at constant $Z_0$</td>
<td>36</td>
</tr>
<tr>
<td>6.6</td>
<td>Accuracy</td>
<td>36</td>
</tr>
<tr>
<td>6.7</td>
<td>Background areas of a symbol</td>
<td>37</td>
</tr>
<tr>
<td>6.8</td>
<td>SNR</td>
<td>38</td>
</tr>
<tr>
<td>7.1</td>
<td>VSLAM system</td>
<td>40</td>
</tr>
</tbody>
</table>
List of Tables

Table 2.1: MRO Mission Overview .................................................................7
Table 2.2: Quick Facts about Phobos and Deimos ...........................................10
Table 4.1: Sensor specification ........................................................................18
Table 6.1: Standard deviations of position of dots in grayscale images .33
Table 6.2: Standard deviations of position of dots in binary images............34
Table 6.3: Accuracy ..........................................................................................36
Table 6.4: SNR ..................................................................................................38
Terminology / Notation

**Acronyms**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOI</td>
<td>Area of Interest</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to Noise Ratio</td>
</tr>
<tr>
<td>HSI</td>
<td>Hue Saturation Intensity</td>
</tr>
<tr>
<td>RGB</td>
<td>Red Green Blue</td>
</tr>
<tr>
<td>Cr</td>
<td>Red component of YCbCr colour space</td>
</tr>
<tr>
<td>DOF</td>
<td>Dimensions of freedom</td>
</tr>
<tr>
<td>MRO</td>
<td>Mars Reconnaissance Orbiter</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics &amp; Space Administration</td>
</tr>
<tr>
<td>ONC</td>
<td>Optical Navigation Camera</td>
</tr>
</tbody>
</table>
Calibration of prepared environment for optical navigation
Jinnu Panilet Panipichai

Introduction
2015-12-14
1 Introduction

1.1 Background

A robot is a very good example for an optical navigation system. Lately, the number of robots that are being hijacked has increased. This can be controlled if a robot has the ability to calibrate the environment in which it has been placed. Therefore, calibrating an environment by a machine vision system has become very important in recent times.

When a machine vision system enters into an environment, the initial position is considered as the reference position. From reference position, the system detects and locates one or more symbols. The machine vision system is then moved to a different position. Displacement or the new position of the system can be calculated if the same set of symbols are captured. After determining the new position, the system will capture a new set of symbols. New set of symbols is located and thus the whole process is repeated until the environment is calibrated completely.

1.2 Overall aim

An environment is created with rotationally independent optimized colour reference labels (symbols) each carrying one byte data encoded in it. A circle enclosed with 10 dots creates a symbol. 2 bigger dots in the circle give the angular rotation of the camera in the system while the remaining 8 smaller dots contribute to the code in the symbol. Thus, it is possible to create as much as 256 different symbols whose codes range from 0 to 255. A machine vision system is used to calibrate this environment. The overall aim of this project is to evaluate the accuracy and precision of the system used to calibrate the prepared environment. In other words, this project aims at finding an answer to the overall research question, “What accuracy and precision can be achieved for a camera based optical position sensing system?”

The initial position of the machine vision system in the environment is taken as the reference position. A Matlab algorithm is developed to detect and decode one or more symbols. The dots in the symbols are located in world co-ordinate system. When the system is moved from the reference position, the symbols detected act as references to find the new position of the system. After calculating the new position, the
Calibration of prepared environment for optical navigation
Jinnu Panilet Panipichai

System detects and decodes another set of symbols and the process continued until the entire environment is calibrated. The steps involved in calibrating the entire environment is shown in Figure 1.1.

![Diagram of steps in environment calibration]

- Qaiser Anwar did the steps written in italics characters
- This project covers the steps in bold characters
- Steps yet to be done are written in normal characters

1.3 Contributions

Qaiser Anwar has completed the initial part of this project as a 30 hp master thesis in 2013 under the topic, “Optical Navigation by recognition of reference labels using 3D calibration of camera” [1]. He managed to find the position of the machine vision system (camera) assuming that the positions of the symbols are known. Anwar’s part in the project is highlighted in italic characters in Figure 1.1.
This project is an extension of Qaiser’s work in which, the positions of the dots in the symbols are calculated assuming that the system (camera) is at the reference position initially. Further displacements of the system are physically measured and positions of the dots in the symbols are found. The parts covered in this project are highlighted in bold characters in Figure 1.1. System accuracy, precision and SNR of the images captured at different positions are also investigated in this project.

1.4 Verifiable goals

A machine vision system (camera) is placed in a prepared environment. The symbols in the environment are captured. The symbols are decoded and the dots in the symbols are located. Results expected which could be verified in section 6 are as follows.

a) Precision, accuracy and SNR are inversely proportional to the distance between the system (camera) and the symbol.

b) Grayscale images give better precision than binary images.

1.5 Outline

Chapter 2 describes about a related work to this project.

Chapter 3 describes theory behind the symbol design and the derivation of formulae to calibrate the environment.

Chapter 4 describes the system used in this project, experimental setup and the reasons behind choosing Matlab as the programming tool.

Chapter 5 describes implementation of methods discussed in Chapter 4.

Chapter 6 gives the explanation of results produced by Matlab scripts.

Chapter 7 discusses the advantages and disadvantages in the project.

Chapter 8 gives the conclusion to the project.
Calibration of prepared environment for optical navigation

Jinru Panilet Panipichai
2 Related work

Optical Navigation attracts numerous researchers recently. One of them is NASA (National Aeronautics and Space Administration) from the United States of America. NASA launched Mars Reconnaissance Orbiter (MRO) [2] on August 12, 2005 to search for evidence that water persisted on the surface of Mars for a long period of time. The previous Mars missions have shown that water flowed across in Mars’ history but researches are going on to conclude whether water was ever around long enough to provide habitat for life. MRO looking for clues about where watery Martian habitats may lie [2] is shown in Figure 2.1.

Mission overview on MRO [2] is given in Table 2.1.

![Mars Reconnaissance Orbiter](image1.png)

Figure 2.1: Mars Reconnaissance Orbiter [2]

<table>
<thead>
<tr>
<th>Mission Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Launch:</strong> August 12, 2005</td>
</tr>
<tr>
<td><strong>Launch location:</strong> Cape Canaveral Air Force Station, Florida</td>
</tr>
<tr>
<td><strong>Arrival:</strong> March 10, 2006</td>
</tr>
<tr>
<td><strong>Weight:</strong> 2,180 kilograms at launch, including fuel</td>
</tr>
<tr>
<td><strong>Electrical Power:</strong> Solar panels</td>
</tr>
<tr>
<td><strong>Rocket:</strong> Atlas V</td>
</tr>
<tr>
<td><strong>Mission Duration:</strong> 2006 - On-going</td>
</tr>
</tbody>
</table>

Table 2.1: MRO Mission Overview [2]
MRO carries 3 engineering instruments [3] that assist with spacecraft navigation and communications and one of them is Optical Navigation Camera (ONC). Optical Navigation Camera [4] was tested for improved navigation capability for future missions. Similar cameras on orbiters of the future will be able to serve as high precision interplanetary “eyes” to guide incoming spacecraft as they near Mars.

From 30 days to 2 days [4] prior to Mars Orbit Insertion, the spacecraft collected a series of images of Mars’ moons Phobos and Deimos. By comparing the observed position of the moons to their predicted positions relative to the background stars, the mission team was able to accurately determine the position of the orbiter in relation to Mars. Figure 2.2 shows a sample image of ONC’s series of images [5] capturing Mars’ moons Phobos and Deimos.

![Image of Mars' moons Phobos and Deimos with Aldebaran and The Pleiades]

Figure 2.2: Mars’ moons – Phobos and Deimos [5]
Figure 2.3 shows the position of the Martian moon Deimos against a background of stars, [6] which is a part of a successful technology demonstration completed by NASA’s MRO before arrival at Mars. Deimos is the smaller of the two Mars’ moons. This example image was taken on March 6, 2006, at a distance of 1.08 million kilometres from Deimos. Deimos has a diameter of 15 kilometres and orbits 23,459 kilometres above the planet’s surface.

While not needed by MRO [4] to navigate to Mars, the data from this experiment demonstrated that a future spacecraft could use this technique to ensure their accurate arrival in Mars. Accuracy will be important to future landers and rovers in need of extremely precise navigation to safely reach their landing sites. Some quick facts about Phobos and Deimos [5] are given in Table 2.2.
<table>
<thead>
<tr>
<th></th>
<th>Phobos</th>
<th>Deimos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean distance from Mars (km)</td>
<td>9377</td>
<td>23436</td>
</tr>
<tr>
<td>Orbital period (Mars days)</td>
<td>0.31891</td>
<td>1.26244</td>
</tr>
<tr>
<td>Major axis (km)</td>
<td>26</td>
<td>16</td>
</tr>
<tr>
<td>Minor axis (km)</td>
<td>18</td>
<td>10</td>
</tr>
<tr>
<td>Mass (x (10^{15}) kg)</td>
<td>10.8</td>
<td>1.8</td>
</tr>
<tr>
<td>Mean density (kg/m(^3))</td>
<td>1900</td>
<td>1750</td>
</tr>
</tbody>
</table>

Table 2.2: Quick Facts about Phobos and Deimos [5]

Mars mission can be related to this project. In this project, Optical Navigation system acts as MRO and the symbols act as Mars' moons, Phobos and Deimos. Mars mission located the moons revolving around Mars while this project aims at locating the symbols in the prepared environment.
3 Theory

This chapter explains why the colours used in the symbol are chosen and the approach to locate the dots in the symbols.

3.1 Colours in the symbol

Two colours are used in the symbol, foreground colour and background colour. When a camera captures a combination of colours, care should be taken so as to achieve the best possible SNR [7]. High SNR in turn gives a high sub-pixel precision. Therefore, the colour with the lowest mean value is selected as the background colour. The colour with the highest SNR with respect to the background colour is selected as the foreground colour. The calculated values for the foreground colour for a HSI (Hue Saturation Intensity) model are $H = 210$, $S = 0.996$, $I = 0.5$. The calculated values for the background colour are $H = 359$, $S = 0.57$, $I = 0.5$.

To encode information in the symbols, the HSI model is converted into a RGB model using Matlab. The computed hue of the HSI colour space for the foreground colour is 210, which is in between 120 and 240 degrees. Therefore the conversion formulae [8] for foreground colour is given by,

$$R = I \times (1 - S) \quad \text{.....Eq (3.1)}$$

$$G = I \left[ 1 + S \left\{ \frac{\cos(H - 120)}{\cos(180 - H)} \right\} \right] \quad \text{.....Eq (3.2)}$$

$$B = I \left[ 1 + S \left\{ 1 - \frac{\cos(H - 120)}{\cos(180 - H)} \right\} \right] \quad \text{.....Eq (3.3)}$$

where, $R$, $G$ and $B$ stand for red, green and blue colour components of the RGB colour space respectively. The computed hue of the HSI colour space for the background colour is 359, which lies in between 240 and 360 degrees. So, the conversions to [8] RGB colour space are given by,

$$R = I \left[ 1 + S \left\{ 1 - \frac{\cos(H - 240)}{\cos(300 - H)} \right\} \right] \quad \text{.....Eq (3.4)}$$

$$G = I \times (1 - S) \quad \text{.....Eq (3.5)}$$
Calibration of prepared environment for optical navigation
Jinnu Panilet Panipichai

Theory 2015-12-14

\[ B = I \left[ 1 + S \left( \frac{\cos(H - 240)}{\cos(300 - H)} \right) \right] \]  

.....Eq (3.6)

RGB images are then converted into YCbCr images because YCbCr colour space has less redundancy compared to RGB colour space. RGB to YCbCr colour space conversion [9] is given by,

\[ Y = 0.229R + 0.587G + 0.114B \]  

.....Eq (3.7)

\[ Cb = 128 - 0.1687R - 0.3313G + 0.5B \]  

.....Eq (3.8)

\[ Cr = 128 + 0.5R - 0.4187G - 0.081B \]  

.....Eq (3.9)

where, \( Y \) is the intensity and \( Cb \) and \( Cr \) are the blue and red Chroma components. In the experiments \( Cr \) components are used because they generate fewer image components after segmentation when compared to that of the components produced by \( Y \) components.

3.2 Imaging Geometry

Position of an object in any space is described by using Cartesian coordinate system to map between three dimensional world plane and two dimensional image plane. The position of an object in world plane is denoted by \((X, Y, Z)\) and in image plane, it is denoted by \((x, y)\).

3.2.1 Perspective projection

Distant objects appear smaller than the original size for a human eye as shown in Figure 3.1. This is called perspective. Orthographic projection ignores the effect of distance to enable high accuracy during measurements. Therefore, perspective projection provides additive realism [10] by making distant objects smaller. If \( \lambda \) is the focal length of the camera used, perspective projection matrix, \( P \) can be written as,

\[
P = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & -\frac{1}{\lambda} & 1
\end{bmatrix}
\]  

.....Eq (3.10)

Let \((X, Y, Z)\) be the co-ordinates of a point in 3D world coordinate system and \((x, y)\) is the projection of the point in 2D image plane. The camera coordinate system \((x, y, z)\) has an image plane coincident with the \((xy)\) plane and the optical axis (established by the centre of the lens)
along the z-axis. Thus the centre of the image plane is at the origin and the centre of the lens is at coordinates (0, 0, λ) [11].

![Figure 3.1: Perspective projection](image)

Perspective projection is implemented using homogenous coordinate system because homogenous coordinate system can represent points even at infinity by using finite coordinates. A point in 3D Cartesian world coordinate system can be shown as,

$$W = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$  \hspace{1cm} .....Eq (3.11)

The homogenous form of a point \((X, Y, Z)\) in the world coordinate system can be given as

$$W_h = \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix}$$  \hspace{1cm} .....Eq (3.12)

The homogenous form of the camera coordinates, \(C_h\) can be given as

$$C_h = P \ast W_h$$  \hspace{1cm} .....Eq (3.13)

Substituting equations, Eq (3.10) and Eq (3.12) in Eq (3.13),
Therefore, homogenous form of camera coordinates are given by,

\[
\begin{bmatrix}
  ch_1 \\
  ch_2 \\
  ch_3 \\
  ch_4
\end{bmatrix}
= \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & -\frac{1}{\lambda} & 1
\end{bmatrix}
\begin{bmatrix}
  kX \\
  kY \\
  kZ \\
  k
\end{bmatrix}
\]

Cartesian form of camera coordinates, 'c' can be obtained by dividing all components of homogenous form by its fourth (last) component [11].

\[
c = \begin{bmatrix}
  x \\
  y \\
  z
\end{bmatrix} = \begin{bmatrix}
  \frac{\lambda X}{\lambda - Z} \\
  \frac{\lambda Y}{\lambda - Z} \\
  \frac{\lambda Z}{\lambda - Z}
\end{bmatrix}
\]

In the above equation, z acts a free variable, which is of no interest in terms of camera model. x and y are the image plane coordinates showing the projection of a 3D point (X, Y, Z) in world coordinates.

### 3.2.2 Translation

(a) Real world coordinate system  
(b) Image plane coordinate system

Figure 3.2: Translation from real world to image plane coordinate system
The process of moving all the points in an object or an image from one plane to another towards the same direction at a constant distance is called translation. Figure 3.2 shows translation from real world coordinates to image plane coordinates. If the position of the camera in world plane co-ordinates is denoted by \((X_0, Y_0, Z_0)\), the transformation matrix \(T\) can be used to locate the origin of the world at the centre of the camera and is given by,

\[
T = \begin{bmatrix}
1 & 0 & 0 & -X_0 \\
0 & 1 & 0 & -Y_0 \\
0 & 0 & 1 & -Z_0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  ........Eq (3.16)

3.2.3 Rotation
The camera used can be rotated along z-axis and so the angle between x and X axes gives the angle of rotation, \(\theta\). The rotation transformation matrix is given by,

\[
R_\theta = \begin{bmatrix}
\cos \theta & \sin \theta & 0 & 0 \\
-\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  ........Eq (3.17)

3.2.4 Environment calibration
Environment calibration is the process of determining the location of the symbol dots in world coordinate system. From Qaiser Anwar’s work [1], the relationship between the image plane and the world coordinate system is given by,

\[
C_h = A \times W_h 
\]  ........Eq (3.18)

\[
\begin{bmatrix}
\text{ch}_1 \\
\text{ch}_2 \\
\text{ch}_3 \\
\text{ch}_4 
\end{bmatrix} = \begin{bmatrix}
A_{11} & A_{12} & A_{13} & A_{14} \\
A_{21} & A_{22} & A_{23} & A_{24} \\
A_{31} & A_{32} & A_{33} & A_{34} \\
A_{41} & A_{42} & A_{43} & A_{44}
\end{bmatrix}\begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}
\]  ........Eq (3.19)

Therefore, a point in world coordinate system \(W_h\) is given by,

\[
W_h = A^{-1} \times C_h
\]  ........Eq (3.20)

So, equation Eq (3.20) becomes,
Calibration of prepared environment for optical navigation
Jinnu Panilet Panipichai

\[
\begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix} =
\begin{bmatrix}
S_{11} & S_{12} & S_{13} & S_{14} \\
S_{21} & S_{22} & S_{23} & S_{24} \\
S_{31} & S_{32} & S_{33} & S_{34} \\
S_{41} & S_{42} & S_{43} & S_{44}
\end{bmatrix}
\begin{bmatrix}
ch_1 \\
ch_2 \\
ch_3 \\
ch_4
\end{bmatrix}
\]

\[.....\text{Eq (3.21)}\]

where,

\[
A^{-1} =
\begin{bmatrix}
S_{11} & S_{12} & S_{13} & S_{14} \\
S_{21} & S_{22} & S_{23} & S_{24} \\
S_{31} & S_{32} & S_{33} & S_{34} \\
S_{41} & S_{42} & S_{43} & S_{44}
\end{bmatrix}
\]

In the experiments, Z-axis is of no interest and so, if Z-axis is ignored,

\[
A^{-1} =
\begin{bmatrix}
S_{11} & S_{12} & S_{14} \\
S_{21} & S_{22} & S_{24} \\
S_{41} & S_{42} & S_{44}
\end{bmatrix}
\]
4 **Methodology / Model**

This chapter deals with the machine vision system involved to capture symbols and the platform used to develop algorithms to work on the captured images. This chapter also gives an idea about the approach on algorithms developed and the outputs expected from the algorithms.

4.1 **System used**

The machine vision system used in this project has an USB camera (5mm focal length), a computer running on Windows or Linux operating systems that has the software to run the camera (ueye software for the camera used) and a tripod to mount the camera. Figure 4.1 shows the experimental setup.

![Experimental setup](image)

**Figure 4.1: Experimental setup**

Camera used by the machine vision system is in this project is an USB 2 interfaced C-Mount lens camera (UI146xSE-C) with 1/2” Aptina CMOS sensor having 3 mega-pixel resolution. Figure 4.2 [12] shows the camera used in this project. Sensor specification [12] is shown in Table 4.1.
4.2 Experimental Procedure

The proposed optical navigation system enters into a prepared environment to capture the symbols attached on the wall. Any number of symbols can be used though only one symbol has been used for the experiments in this project. The initial position of the camera becomes the reference position. Keeping the distance between the camera and the symbols \((Z_0)\) constant, the camera is moved to various positions. To make the displacement measurements easier, the position of the camera along X-axis \((X_0)\) is also kept constant. The camera is displaced only in Y-axis. The maximum displacement was 33.5 cm. In other words, images are captured by continuously changing \(Y_0\). This experiment is done to estimate the accuracy of the system at \(Z_0\). This is shown in Figure 4.3. To find the accuracy of the system at different distances between camera and symbol, the entire experiment is repeated for different \(Z_0\) values varying from 20 cm to 200 cm.

To estimate the precision, the system is kept at constant position and 250 images are taken repeatedly. Then, the distance between the camera and
the symbol ($Z_0$) is changed to take another set of 250 images. In the experiments here $Z_0$ is changed from 20 cm to 200 cm.

**Figure 4.3: Accuracy Measurement at constant $Z_0$**

### 4.3 Algorithm

Matlab is the tool used to develop algorithms and work on the captured images. Matlab is a script-based language in an interactive environment for numerical computation, visualization and programming. Matlab environment is very easy to develop algorithms. An algorithm can be tested with ease without any recompilation. These are the reasons why Matlab had been chosen as the platform to develop the algorithms.

The inputs of the algorithms would be the position of the system in X and Y – axes and the distance between the camera and the symbols on the wall. The algorithms are developed to find the number of symbols in the captured images and then decode them. The algorithms also give the angle at which the camera is tilted and the positions of all ten dots in the symbols. All the positions found are in respect to the initial position of the camera (reference position of the camera).
5 Implementation

This chapter deals with the design of the symbols and the implementation of the proposed method in the previous chapter to identify the symbols in the images, decode them and find the position of the symbol in the real world coordinates.

5.1 Symbols design

The symbols are designed to have a circle containing 10 dots inside. Out of the 10 dots, 2 dots are relatively bigger than the other 8 dots. These two dots remain in a constant position for all symbols, one at the centre of the circle and the other near to surface at lower half of the circle at an angle of -90 degrees or +270 degrees between them. The remaining eight dots represent eight-bit code of the symbol. The most significant bit lies in between the 2 bigger dots and for every 45 degrees in the clockwise direction from the centre of the circle, one dot is plotted to represent the respective bit in the eight-bit code represented by the symbol. The distance between an individual dot and the centre dot of the symbol is one-third of the distance between the 2 big dots (ref_dist) if the corresponding bit represents ‘0’. If the bit represents ‘1’, then distance between the corresponding dot and centre dot of the symbol will be two-third of ref_dist. An example of the symbols used is shown in Figure 5.1.

![Figure 5.1: Symbol representing the code - 138](image)

After the symbols have been designed, the symbols are printed using a coloured laser printer. The printed symbols are attached on the wall of the prepared environment and the images are taken using the available machine vision system.
5.2 Symbol detection

The captured images are then converted into YCbCr colour space because RGB images are highly redundant. Red component (Cr) of YCbCr images are used because they have the least number of segmented components in binary images when compared to the other components. To reduce the noise of the Cr image, an 11 X 11 mean value filter is used. When Cr image and 11 X 11 kernel of mean value filter are convolved, a background is computed from the image [13]. The resulted background image is then subtracted from the original image and then a threshold value is applied. This threshold value depends on the distance between the camera and the object. The higher the distance between the camera and the object, the smaller is the threshold value. This enables to detect the symbol at larger distances.

![Figure 5.2: Background elimination](image)

In every image, area of interest (AOI) is found which covers the symbols in the image. The outer circles of the symbols are most likely to have a large eccentricity with a big area. There might be other objects in the image with similar eccentricity and area as well. To eliminate those objects from AOI, a condition is inserted such that the AOIs have 11 objects. These 11 objects include the outer circle and 10 dots within them that eventually form a symbol. The 10 dots are then located in pixel coordinates.

![Figure 5.3: Symbol and dots detection](image)

X represents small dots, X represents centre dot, X represents corner dot
5.3 Camera rotation

Out of the 10 dots that are located in pixel co-ordinates, the location of 2 bigger dots can give the angle at which the camera is been rotated. The angle between the 2 dots ($\varphi$) can be given by,

$$\varphi = \tan^{-1}\left(\frac{by - ay}{bx - ax}\right)$$

.....Eq (5.1)

where, (ax, ay) represent the pixel co-ordinates of the center dot and (bx, by) represent the pixel co-ordinates of the dot that is at the surface of the lower half of the circle as shown in Figure 5.4.

![Figure 5.4: Big Dots](image)

The angle between the 2 dots ($\varphi$) should be 270 degrees if the camera is not rotated. When $\varphi$ is not equal to 270 degrees (when the camera is rotated), the angle at which the camera is rotated ($\theta$) is given by,

$$\theta = \varphi - 270$$

.....Eq (5.2)

5.4 Symbol decoding

The 8 small dots in the symbol represent the code in the symbol. To decode the code, it is necessary to find the dots that represent the first and last bits of the code. To enable this, the angles ($\alpha_i$) between the individual dots and the centre dot of the symbol are found with respect to the corner dot. For example, if small dots are represented by the pixel co-ordinates $(cx_i, cy_i)$ then the angles ($\alpha_i$) can be given by,

$$\alpha_i = \theta - \tan^{-1}\left(\frac{cy_i - ay}{cx_i - ax}\right)$$

.....Eq (5.3)

where, i = 1 to 8. After all the angles are found out, the dots are arranged in the ascending order of their corresponding angles. The dot with the
least angle gives the MSB of the code while the dot with the highest angle gives the LSB of the code. To estimate whether the dot represents ‘1’ or ‘0’ bit, the distances between individual dots and the centre dot of the symbol (dist) are calculated and compared with ‘ref_dist’. If ‘dist’ is more than half of ‘ref_dist’, then the dot represents ‘1’ else ‘0’. Formulae for finding ‘dist’ and ‘ref_dist’ are given by,

\[
\text{ref\_dist} = \sqrt{[(bx - ax)^2 + (by - ay)^2]} \quad \text{.....Eq (5.4)}
\]

\[
\text{dist}_i = \sqrt{[(cx_i - ax)^2 + (cy_i - ay)^2]} \quad \text{.....Eq (5.5)}
\]

### 5.5 From pixel co-ordinates to image plane co-ordinates

The dots in the symbol are located in pixel co-ordinates from section 5.2. To convert these locations in to the image plane co-ordinates, pixel width and pixel height of the camera should be calculated.

- Pixel width \( (p_x) = \text{width of sensor/width of the image} \)
- Pixel height \( (p_y) = \text{width of sensor/height of the image} \)

Width of sensor used is 6.554 mm and height of sensor used is 4.915 mm respectively. Resolution of camera images is 2048×1536 [12]. Therefore,

\[
p_x = \frac{6.554 \text{ mm}}{2048} = 3.199 \times 10^{-4} \text{ cm} \quad \text{.....Eq (5.6)}
\]

\[
p_x = \frac{4.915 \text{ mm}}{1536} = 3.2 \times 10^{-4} \text{ cm} \quad \text{.....Eq (5.7)}
\]

The position of a point in an image plane can be calculated using,

\[
x = (o_x - x_{im}) p_x
\]

\[
y = (o_y - y_{im}) p_y
\]

where \((o_x,o_y)\) are co-ordinates of the principal point in pixels. Generally, centre of the image acts as the principal point in pixels. Therefore,

\[
o_x = \text{width of the image/2} = 2048/2 = 1024
\]

\[
o_y = \text{height of the image/2} = 1536/2 = 768
\]
5.6 Location of dots in world co-ordinates

The perspective transformation matrix \( P \), rotation matrix \( R_\theta \) and the translation matrix \( T \) from equations, Eqn (3.10), Eqn (3.17) and Eqn (3.16) are given as

\[
P = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & -\frac{1}{\lambda} & 1
\end{bmatrix}
\]

\[
R_\theta = \begin{bmatrix}
\cos \theta & \sin \theta & 0 & 0 \\
-\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
T = \begin{bmatrix}
1 & 0 & 0 & -X_0 \\
0 & 1 & 0 & -Y_0 \\
0 & 0 & 1 & -Z_0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

where, \( \lambda \)-focal length of camera, \( \theta \)-angle at which camera is rotated and \((X_0, Y_0, Z_0)\) is the real world position of the camera.

So, the calibration matrix \( A \) can be written as

\[
A = P \times R_\theta \times T \quad \text{.....Eq (5.8)}
\]

\[
A = \begin{bmatrix}
\cos \theta & \sin \theta & 0 & -X_0 \cos \theta - Y_0 \sin \theta \\
-\sin \theta & \cos \theta & 0 & X_0 \sin \theta - Y_0 \cos \theta \\
0 & 0 & 1 & -Z_0 \\
0 & 0 & -\frac{1}{\lambda} & \left(\frac{Z_0}{\lambda}\right) + 1
\end{bmatrix} \quad \text{.....Eq (5.9)}
\]

The relationship between the homogenous form of camera co-ordinates and a point in the world co-ordinate system can be written as,

\[
C_h = A \times W_h \quad \text{.....Eq (5.10)}
\]

\[
\begin{bmatrix}
\text{ch}_1 \\
\text{ch}_2 \\
\text{ch}_3 \\
\text{ch}_4
\end{bmatrix} = \begin{bmatrix}
\cos \theta & \sin \theta & 0 & -X_0 \cos \theta - Y_0 \sin \theta \\
-\sin \theta & \cos \theta & 0 & X_0 \sin \theta - Y_0 \cos \theta \\
0 & 0 & 1 & -Z_0 \\
0 & 0 & -\frac{1}{\lambda} & \left(\frac{Z_0}{\lambda}\right) + 1
\end{bmatrix} \begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix} \quad \text{.....Eq (5.11)}
\]
If Z-axis is ignored i.e., if $Z = 0$ in equation Eq (5.11), then,

$$\begin{bmatrix}
ch_1 \\
ch_2 \\
ch_4
\end{bmatrix} = \begin{bmatrix}
\cos \theta & \sin \theta & -X_0 \cos \theta - Y_0 \sin \theta \\
-\sin \theta & \cos \theta & X_0 \sin \theta - Y_0 \cos \theta \\
0 & 0 & \left(\frac{Z_0}{\lambda}\right) + 1
\end{bmatrix}\begin{bmatrix}
X \\
Y \\
1
\end{bmatrix} \quad \text{.....Eq (5.12)}$$

Therefore, the calibration matrix (A) can be written as

$$A = \begin{bmatrix}
\cos \theta & \sin \theta & -X_0 \cos \theta - Y_0 \sin \theta \\
-\sin \theta & \cos \theta & X_0 \sin \theta - Y_0 \cos \theta \\
0 & 0 & \left(\frac{Z_0}{\lambda}\right) + 1
\end{bmatrix} \quad \text{.....Eq (5.13)}$$

To get world co-ordinates ($W_h$) from equations, Eq (5.10) and Eq (5.12),

$$W_h = A^{-1}\times C_h \quad \text{.....Eq (5.14)}$$

Therefore, from equation, Eq (5.13), inverse of $A$ [14] is calculated as,

$$A^{-1} = \begin{bmatrix}
\cos \theta & \sin \theta & \frac{X_0}{1 + \left(\frac{Z_0}{\lambda}\right)} \\
-\sin \theta & \cos \theta & \frac{Y_0}{1 + \left(\frac{Z_0}{\lambda}\right)} \\
0 & 0 & \frac{1}{1 + \left(\frac{Z_0}{\lambda}\right)}
\end{bmatrix} \quad \text{.....Eq (5.15)}$$

Substituting the equation, Eq (5.15) in equation, Eq (5.14) the real world co-ordinates ($W_h$) of a point can be written as,

$$\begin{bmatrix}
X \\
Y \\
1
\end{bmatrix} = \begin{bmatrix}
\cos \theta & \sin \theta & \frac{X_0}{1 + \left(\frac{Z_0}{\lambda}\right)} \\
-\sin \theta & \cos \theta & \frac{Y_0}{1 + \left(\frac{Z_0}{\lambda}\right)} \\
0 & 0 & \frac{1}{1 + \left(\frac{Z_0}{\lambda}\right)}
\end{bmatrix}\begin{bmatrix}
ch_1 \\
ch_2 \\
ch_4
\end{bmatrix} \quad \text{.....Eq (5.16)}$$

$$\begin{bmatrix}
X \\
Y \\
1
\end{bmatrix} = \begin{bmatrix}
S_{11} & S_{12} & S_{14} \\
S_{21} & S_{22} & S_{24} \\
S_{41} & S_{42} & S_{44}
\end{bmatrix}\begin{bmatrix}
\chi_1 \\
\chi_2 \\
\chi_4
\end{bmatrix} \quad \text{.....Eq (5.17)}$$
From equations Eq (5.14) and Eq (5.17),

\[
A^{-1} = \begin{bmatrix}
S_{11} & S_{12} & S_{14} \\
S_{21} & S_{22} & S_{24} \\
S_{41} & S_{42} & S_{44}
\end{bmatrix}  \quad .....\text{Eq (5.18)}
\]

From equation Eq (5.17),

\[
X = S_{11}ch_1 + S_{12}ch_2 + S_{14}ch_4  \quad .....\text{Eq (5.19)}
\]

\[
Y = S_{21}ch_1 + S_{22}ch_2 + S_{24}ch_4  \quad .....\text{Eq (5.20)}
\]

\[
1 = S_{41}ch_1 + S_{42}ch_2 + S_{44}ch_4  \quad .....\text{Eq (5.21)}
\]

The co-efficient of image co-ordinates from homogenous co-ordinates are computed as,

\[
x = \frac{ch_1}{ch_4}  \quad .....\text{Eq (5.22)}
\]

\[
y = \frac{ch_2}{ch_4}  \quad .....\text{Eq (5.23)}
\]

Dividing equations Eq (5.19), Eq (5.20) and Eq (5.21) by \( ch_4 \),

\[
X\left(\frac{1}{ch_4}\right) = \left[\left(S_{11}\left(\frac{ch_1}{ch_4}\right)\right) + \left(S_{12}\left(\frac{ch_2}{ch_4}\right)\right) + \left(S_{14}\left(\frac{ch_4}{ch_4}\right)\right)\right]  \quad .....\text{Eq (5.24)}
\]

\[
Y\left(\frac{1}{ch_4}\right) = \left[\left(S_{21}\left(\frac{ch_1}{ch_4}\right)\right) + \left(S_{22}\left(\frac{ch_2}{ch_4}\right)\right) + \left(S_{24}\left(\frac{ch_4}{ch_4}\right)\right)\right]  \quad .....\text{Eq (5.25)}
\]

\[
\left(\frac{1}{ch_4}\right) = \left[\left(S_{41}\left(\frac{ch_1}{ch_4}\right)\right) + \left(S_{42}\left(\frac{ch_2}{ch_4}\right)\right) + \left(S_{44}\left(\frac{ch_4}{ch_4}\right)\right)\right]  \quad .....\text{Eq (5.26)}
\]

Substituting the values of \( (1/ch_4) \), \( (ch_1/ch_4) \) and \( (ch_2/ch_4) \) from equations Eq (5.26), Eq (5.22), and Eq (5.23) respectively in equations Eq (5.24) and Eq (5.25),

\[
S_{41}XX + S_{42}YX + S_{44}X = S_{11}x + S_{12}y + S_{14}  \quad .....\text{Eq (5.27)}
\]

\[
S_{41}YX + S_{42}YY + S_{44}Y = S_{21}x + S_{22}y + S_{24}  \quad .....\text{Eq (5.28)}
\]
From equations Eq (5.16) and Eq (5.17),

\[ S_{41} = 0 \]
\[ S_{42} = 0 \]

Therefore equations Eq (5.27) and Eq (5.28) changes to,

\[ S_{44}X = S_{11}x + S_{12}y + S_{14} \quad \text{.....Eq (5.29)} \]
\[ S_{44}Y = S_{21}x + S_{22}y + S_{24} \quad \text{.....Eq (5.30)} \]

Rewriting the equations, Eq (5.29) and Eq (5.30) in matrix form and extending it to ‘n’ number of values,

\[
\begin{bmatrix}
X_1 \\
Y_1 \\
X_2 \\
Y_2 \\
\vdots \\
X_n \\
Y_n \\
\end{bmatrix}
= \begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_1 & y_1 & 1 \\
x_2 & y_2 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_2 & y_2 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
x_n & y_n & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_n & y_n & 1 \\
\end{bmatrix}
\begin{bmatrix}
S_{11} \\
S_{12} \\
S_{14} \\
S_{21} \\
S_{22} \\
S_{24} \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
X_1 \\
Y_1 \\
X_2 \\
Y_2 \\
\vdots \\
X_n \\
Y_n \\
\end{bmatrix}
= \frac{1}{S_{44}} \begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_1 & y_1 & 1 \\
x_2 & y_2 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_2 & y_2 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
x_n & y_n & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_n & y_n & 1 \\
\end{bmatrix}
\begin{bmatrix}
S_{11} \\
S_{12} \\
S_{14} \\
S_{21} \\
S_{22} \\
S_{24} \\
\end{bmatrix}
\]

The above equation, Eq (5.32) gives the positions of the dots in a symbol in real world co-ordinates with \((x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\) are the positions of the dots in symbol in image plane co-ordinates and \((X_1, Y_1), (X_2, Y_2), \ldots (X_n, Y_n)\) giving the positions of the dots in real world co-ordinates. If the position of only one dot in the symbol is needed, equations Eq (5.29) and Eq (5.30) can be used.

### 5.7 Illustration

Let the system (camera) at the reference position i.e., \(X_0 = 0\) and \(Y_0 = 0\). Let the distance between the camera and the symbol, \(Z_0 = 50\ \text{cm.}\) Focal length of the camera, \(\lambda = 5\ \text{mm.}\) Image obtained is shown in Figure 5.5.
Initially, the angle ($\theta$) at which the camera is rotated is found out. In this case, it is found out to be, $\theta = -0.04 \approx 0$. The coefficients of transformation matrix ($A^{-1}$) from equations Eq (5.17) and Eq (5.16),

$$\begin{bmatrix} S_{11} & S_{12} & S_{14} \\ S_{21} & S_{22} & S_{24} \\ S_{41} & S_{42} & S_{44} \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta & X_0 \\ -\sin \theta & \cos \theta & Y_0 \\ 0 & 0 & 1 + \left(\frac{Z_0}{\lambda}\right) \end{bmatrix}$$

Substituting the values of $X_0$, $Y_0$, $Z_0$, $\lambda$ and $\theta$ in equation Eq (5.33),

$$\begin{bmatrix} S_{11} & S_{12} & S_{14} \\ S_{21} & S_{22} & S_{24} \\ S_{41} & S_{42} & S_{44} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 + \left(\frac{Z_0}{\lambda}\right) \end{bmatrix}$$

Therefore, the positions of centre and corner dots in the symbol in real world co-ordinates can be found out using the equations, Eq (5.34), Eq (5.29) and Eq (5.30). Remaining 8 dots in the symbol can be found using the equations Eq (5.34) and Eq (5.32). The output image showing the location of the dots is shown in Figure 5.6.
The position of the centre dot, the corner dot and the 8 small dots in the order of their corresponding bits starting from MSB at first and LSB at the last are obtained as shown in Figure 5.7. The code carried by the symbol is also displayed.

This approach can also be followed for capturing more than one symbol. The input image is shown in Figure 5.8 and the output of the Matlab script is shown in Figure 5.9 when 2 symbols are used.
Figure 5.8: Input image for 2 symbols

![Input image for 2 symbols]

Figure 5.9: Output for the Matlab script

The output gives the 2 codes represented by the symbols. The first two columns in “Center_XY”, “Corner_XY” and “Small_dots_XY” in Figure 5.9 represent X-axis real world co-ordinates while the last two columns represent the Y-axis real world co-ordinates.
<table>
<thead>
<tr>
<th>Calibration of prepared environment for optical navigation</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jinnu Panilet Panipichai</td>
<td>2015-12-14</td>
</tr>
</tbody>
</table>
6 Result

This chapter deals with precision, accuracy and SNR (Signal to Noise Ratio) of camera at different distances between camera and symbol.

6.1 Precision

A number of images are taken keeping the camera in a fixed position (250 images are taken here). The positions of dots in the symbol are calculated in image plane and in real world co-ordinates both in binary images and grayscale images. The standard deviations of individual dots in the symbol are calculated. The mean of all standard deviations are found. The entire process is repeated for different camera distances from the symbol ($Z_0$).

<table>
<thead>
<tr>
<th>Distance (cm)</th>
<th>Standard deviations X coordinates</th>
<th>Standard deviations Y coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>World coordinates (cm)</td>
<td>Image plane coordinates (pixels)</td>
</tr>
<tr>
<td>200</td>
<td>0.2670</td>
<td>0.0339</td>
</tr>
<tr>
<td>100</td>
<td>0.1364</td>
<td>0.0088</td>
</tr>
<tr>
<td>90</td>
<td>0.0944</td>
<td>0.0051</td>
</tr>
<tr>
<td>80</td>
<td>0.0978</td>
<td>0.0048</td>
</tr>
<tr>
<td>70</td>
<td>0.1880</td>
<td>0.0086</td>
</tr>
<tr>
<td>60</td>
<td>0.1054</td>
<td>0.0041</td>
</tr>
<tr>
<td>50</td>
<td>0.1376</td>
<td>0.0045</td>
</tr>
<tr>
<td>20</td>
<td>0.1379</td>
<td>0.0021</td>
</tr>
</tbody>
</table>

Table 6.1: Standard deviations of position of dots in grayscale images

Precision increases while the standard deviation decreases. Standard deviations of positions of the dots at different distances between the camera and symbol that are calculated in grayscale images are given in Table 6.1. Standard deviations of positions of the dots at different distances between the camera and symbol that are calculated in binary images are given in Table 6.2. From Table 6.1 and Table 6.2, it can be clearly seen that the standard deviations increases with the distance between the camera and symbol ($Z_0$). Precision is slightly better when the dots are located in grayscale images compared to binary images. This is clearly shown in Figure 6.1, Figure 6.2, Figure 6.3 and Figure 6.4.
Calibration of prepared environment for optical navigation
Jinnu Panilet Panipichai

<table>
<thead>
<tr>
<th>Distance (cm)</th>
<th>World coordinates X</th>
<th>Image plane coordinates</th>
<th>World coordinates Y</th>
<th>Image plane coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Standard deviations</td>
<td></td>
<td>Standard deviations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>X coordinates</td>
<td></td>
<td>Y coordinates</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(cm)</td>
<td>(pixels)</td>
<td>(cm)</td>
<td>(pixels)</td>
</tr>
<tr>
<td>200</td>
<td>0.2688</td>
<td>0.0342</td>
<td>0.2120</td>
<td>0.0269</td>
</tr>
<tr>
<td>100</td>
<td>0.1401</td>
<td>0.0090</td>
<td>0.1289</td>
<td>0.0079</td>
</tr>
<tr>
<td>90</td>
<td>0.0977</td>
<td>0.0053</td>
<td>0.0983</td>
<td>0.0057</td>
</tr>
<tr>
<td>80</td>
<td>0.1025</td>
<td>0.0050</td>
<td>0.1139</td>
<td>0.0058</td>
</tr>
<tr>
<td>70</td>
<td>0.1902</td>
<td>0.0087</td>
<td>0.1823</td>
<td>0.0085</td>
</tr>
<tr>
<td>60</td>
<td>0.1113</td>
<td>0.0043</td>
<td>0.1134</td>
<td>0.0043</td>
</tr>
<tr>
<td>50</td>
<td>0.1443</td>
<td>0.0047</td>
<td>0.1311</td>
<td>0.0040</td>
</tr>
<tr>
<td>20</td>
<td>0.1428</td>
<td>0.0022</td>
<td>0.1460</td>
<td>0.0016</td>
</tr>
</tbody>
</table>

Table 6.2: Standard deviations of position of dots in binary images

Figure 6.1: Standard deviations in World coordinates (X-axis)

Figure 6.2: Standard deviations in World coordinates (Y-axis)
6.2 Accuracy

Keeping the distance between the camera and the symbols (Z₀) constant, the camera is moved to various positions. To make the displacement measurements easier, the position of the camera along X-axis (X₀) is also kept constant as explained in Section 4.2. The camera displacement along Y-axis (ΔY₀) is measured. The average difference in positions of all the dots (ΔY) in the symbol is calculated. Accuracy is given by,

\[ \text{Accuracy} = \frac{\Delta X_0}{\Delta X} \]  

.....Eq (6.1)

The above procedure is repeated for many camera displacements and the corresponding accuracies are found. The average accuracy is then recorded as the accuracy of the camera at Z₀. Images taken at the same distance (Z₀ = 20 cm) between camera and symbol, but at different positions, ((X₀, Y₀) = (0, 2 cm)) and ((X₀, Y₀) = (0, 12 cm)) are shown in Figure 6.5,
This experiment is repeated for different distances between the camera and symbol ($Z_0$). $Z_0$ is varied from 20 cm to 200 cm and the accuracies of the camera are tabulated in Table 6.3. The graph showing the accuracy of the camera at different $Z_0$ is shown in Figure 6.6.

### Table 6.3: Accuracy

<table>
<thead>
<tr>
<th>Distance (cm)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>93.19</td>
</tr>
<tr>
<td>100</td>
<td>97.76</td>
</tr>
<tr>
<td>90</td>
<td>98.24</td>
</tr>
<tr>
<td>80</td>
<td>97.61</td>
</tr>
<tr>
<td>70</td>
<td>96.59</td>
</tr>
<tr>
<td>60</td>
<td>98.37</td>
</tr>
<tr>
<td>50</td>
<td>94.57</td>
</tr>
<tr>
<td>20</td>
<td>98.53</td>
</tr>
</tbody>
</table>

![Figure 6.5: Different camera positions at constant $Z_0$](image1)

![Figure 6.6: Accuracy](image2)
6.3 SNR

Signal to Noise Ratio of an image is given by the ratio of mean value of signal in the image and standard deviation of noise in the image. In an image, it is very difficult to differentiate noise from the original signal since there is no ideal image. So, the area of interest in an image or the symbol in an image (in this case) is concentrated here. SNR is given by,

\[
\text{SNR} = \frac{\mu_{\text{symbol}}}{\sigma_{\text{symbol}}} \quad \text{.....Eq (6.2)}
\]

where,
\[
\mu_{\text{symbol}} \quad \text{– mean of the pixel values covered by the symbol},
\]
\[
\sigma_{\text{symbol}} \quad \text{– standard deviation of the pixels covered by the symbol}
\]

To get SNR in decibels, Eqn (6.2) changes as,

\[
\text{SNR} = 20 \times \log_{10} \left( \frac{\mu_{\text{symbol}}}{\sigma_{\text{symbol}}} \right) \quad \text{dB} \quad \text{.....Eq (6.3)}
\]

To calculate mean and standard deviation of only the pixels covered by the symbol, it is necessary to consider the background of the symbol. Figure 6.7 shows the background areas analysed in a symbol image.

![Figure 6.7: Background areas of a symbol](image)

Considering the background, mean and standard deviation of symbol are given by,

\[
\mu_{\text{symbol}} = \mu_{\text{gray}} - \mu_{\text{bg}} \quad \text{.....Eq (6.4)}
\]

\[
\sigma_{\text{symbol}} = \sqrt{ \left( \sigma_{\text{gray}} \right)^2 + \left( \sigma_{\text{bg}} \right)^2 } \quad \text{.....Eq (6.5)}
\]

where,
\[
\mu_{\text{gray}}, \mu_{\text{bg}} \quad \text{are mean of grayscale and background images respectively},
\]
\[
\sigma_{\text{gray}}, \sigma_{\text{bg}} \quad \text{are standard deviations of grayscale and background images.}
\]
Table 6.4 shows SNR for the camera at different distances between the camera and the symbol. Image frames used in section 6.1 are used here too. It can be observed that SNR tends to decrease with the increase in the distance between camera and symbol. This can also be seen from the graph in Figure 6.8.

Precision in section 6.1 and SNR in section 6.3 use same sets of images or image frames. Both precision and SNR increases with the decrease in distance between camera and symbol. Therefore, SNR and precision of the camera are directly proportional to each other.

Minimum and maximum distances at which the camera used are 20 cm and 200 cm from the symbol. So, if ‘x’ is the minimum distance at which a symbol could be detected by the given camera, the same symbol can be detected at least from a distance of ‘10x’ from the symbol.
7 Discussion

This chapter deals with the advantages and disadvantages of the project.

7.1 Disadvantages of the previous work eliminated

This project is a continuation of Qaiser Anwar’s work as discussed in section 1.3. Qaiser managed to find the position of the system and he needed at least 6 symbols. He found the position of the system by considering the centroid of only one dot that is in the centre of a symbol. He captured the images, which had a default resolution of 2048×1536. But then, he resized those images into 840×480. The reason mentioned was the increase in processing time if the images are processed with the default resolution.

In this project, the centroids of all the dots in a symbol are taken into account. This also helped in the fact that only one symbol can be used to find the position of a system. The images are processed with their default resolution, which is 2048×1536. Resizing is not done in the captured images before processing. The minimum accuracy or the highest error in finding the positions of the dots in the symbol was ±2.3 cm. The highest error was found when the system is placed at the maximum distance, 200 cm from the symbol. The least error, ±0.29 cm was found when the system was placed at the closest to the symbol, which is 20 cm from the symbol.

7.2 Minimum and maximum Z₀

If the system used needs a minimum distance of ‘D’ away from the symbol (Z₀) to detect the symbol, then the system can detect the symbol at a distance of 10D away from the symbol. This can be seen from the experiments as the system used detected symbol from both 20 cm and 200 cm away from the symbol.

7.3 Disadvantage

The main disadvantage in this project is the coding used in this project. The coding used in the project is not robust. The model parameters used to detect the symbol has to be changed manually whenever the distance between the system and the symbol (Z₀) is changed. The reason behind the coding being not robust might be because of the default changes in the environment especially light intensity. The changes in light intensity
in turn create an inconsistency in noise levels and so the parameters needed to be changed for every change in $Z_0$ (distance between camera and symbol).

### 7.4 Comparison with VSLAM

VSLAM (Visual Simultaneous Localization And Mapping) is the process of constructing or updating a map of an unknown environment starting from an unknown position. VSLAM system [16] is shown in Figure 7.1. Figure 7.1 and Figure 1.1 can be compared and found similar to each other if the system and symbol dots in Figure 1.1 are respectively taken as robot and landmarks in Figure 7.1.

![VSLAM system](image.png)

*Figure 7.1: VSLAM system [16]*

‘New Image Input’ block in Figure 7.1 has landmarks embedded in it. The landmarks are extracted from the image input. This block can be compared to a combination of first 3 blocks in Figure 1.1 where symbol and symbol dots are identified and located at different system positions.

According to Qaiser Anwar [1], when positions of the symbols are known, the pose of the system can be estimated. Work done by Qaiser
Anwar is marked in italic characters in Figure 1.1. This can be compared to the block, ‘Calculate the pose of robot’ in Figure 7.1.

VSLAM calculates the pose of robot if the landmarks are recognized in previous databases. If landmarks of the image input are not recognized, VSLAM creates new landmarks and updates the database. This part can be explained by the block, ‘Check if the entire environment is calibrated’ in Figure 1.1. This block is written in normal characters to indicate that the part is not completed. So, if a database can be created and updated simultaneously, this project in combination with Qaiser Anwar’s work will be a solution for the SLAM problem. SLAM concentrates on finding a solution to make an autonomous vehicle to start in an unknown location with an unknown environment [15]. The vehicle then builds up a map of the environment and finds its own location from the map simultaneously. This is called as SLAM problem.

7.5 Future work

This project combined with Qaiser Anwar’s work with some more work can be a solution for the SLAM problem. The future work of this project should be concentrated on creating a database, which is capable to get the updates as well. If a database is created and then updated for every new symbol, a machine vision system will be able to start in an unknown location and in an unknown environment and then identify the symbols. After a symbol is identified, the database can be checked to find whether the symbol’s locality is already found. If the symbol’s location is not found in the database, the database can be updated for the new symbol. This process should be continued for the entire environment.

7.6 Effects of this project on ethics

Optical position sensing can improve robots for elderly care. For instance, these types of robots can reduce the burden of driving for the elderly and disabled persons. But, the problem here is about who takes up the responsibility if and when an accident occurs and the debate just goes on. Whether a robot is liable or in fact, whether it is ethical to surrender the responsibilities of human lives to a robot remains a question mark.

SLAM robots can also be used for surgeries based on image guidance systems (IGS). Researchers say that SLAM robots improve accuracy and
efficiency of IGS based surgeries [17]. The problem really starts when the robots have access to the medical records. There is a fear of compromised privacy of medical records of patients.

7.7 Effects of this project on society

Clearly, the technology has some ethical issues but talking about social effects, demographic problem stands out especially in caregivers’ area. So, many countries are trying to develop care robots to help the elderly and disabled [18]. There are two social issues observed in this context. The first one being isolation from human beings. Disabled and elderly might feel more isolated than they actually are because there would be reduced or no time spent with human beings. The second issue is their behaviour. The robots behave or act the same way regardless of the situation or environment that might cause embarrassment to the care receiver.

In the medical field, SLAM robots are designed so as to interact with the patients like the human doctors do. In this situation, the patients have to delude themselves that the robot has inner states and feelings [18]. Many patients would not buy into deceiving themselves and so there lays the ultimate problem.
8 Conclusion

A machine vision system is used to calibrate the prepared environment. The main aim of this project to evaluate accuracy and precision of the system used has been achieved. The symbols in the images are decoded and the geographical positions of the dots in the symbols are located when the system is placed at different positions. If the locations of symbols are known, the location of the system can also be determined. Therefore, it is possible to calibrate the prepared environment entirely if both these works are combined. Precisions and accuracies of the system at different distances from the symbol are compared. Precision and accuracy are inversely proportional to the distance between the system and the symbol. SNR for the same sets of image frames are calculated and found to be either directly proportional to the precision of the camera or inversely proportional to the distance between the system and the symbol.
References


Calibration of prepared environment for optical navigation
Jinnu Panilet Panipichai


Appendix: Documentation of own developed program code

clear all; clc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% POSITION OF CAMERA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
X0 = 0; % All measurements
Y0 = 0; % are taken
Z0 = 20; % in centimeters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CAMERA PARAMETERS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fo = 0.5; % focal length(cm)
sw = 0.6554; % sensor width(cm)
sh = 0.4915; % sensor height(cm)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MODEL PARAMETERS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dS = 0.1; % threshold on dynamicSegment function when Z0 = 20
% dS = 0.17; % threshold on dynamicSegment when Z0 is not equal to 20
bwL = round(2100/Z0); % threshold on noise reduction to find dots in symbols when Z0 is not equal to 200
% bwL = round(600/Z0); % threshold on noise reduction to find dots in symbols when Z0 = 200
bwL2 = bwL*10; % threshold on noise reduction to find symbols
strL = round(200/Z0); % threshold on structuring element
saL = 16500; % threshold on area of symbol when Z0 = 20
% saL = 4000; % threshold on area of symbol when Z0 = 50
% saL = 3750; % threshold on area of symbol when Z0 = 60
% saL = 3000; % threshold on area of symbol when Z0 = 70
% saL = 2400; % threshold on area of symbol when Z0 = 80
% saL = 1950; % threshold on area of symbol when Z0 = 90
% saL = 1650; % threshold on area of symbol when Z0 = 100
% saL = 540; % threshold on area of symbol when Z0 = 200
sdL = saL/30; % threshold on area of small dots
bdL = saL/15; % threshold on area of big dots
SymCodes = []; % codes encoded in symbols
sn = 1; % number of symbols
dn = 1; % number of small dots in the symbol

%Image Acquisition
%Preprocessing
in = imread('final/20/12.bmp');%input image
du = double(in);
cr = 128 + (0.5 * du(:,:,1) - 0.4187 * du(:,:,2) - 0.0813 * du(:,:,3));%red component of YCbCr plane
gr = cr/max(max(cr));%conversion into grayscale image
[inr, inc] = size(gr);%finising the number of rows and columns
pur_sym = zeros(inr, inc);%to seperate symbols from background to find SNR

%Segmentation
[bi, f] = dynamicSegment(gr,dS);%11X11 Convolving mean value filter and image subraction
bw = bwareaopen(bi, bwL);%removing noise to find dots in symbol
imc = imclose(bw,strel('square',strL));%recovering lost pixels due to noise
bw2 = bwareaopen(imc, bwL2);%removing noise to find the symbol

%Labeling
%Feature extraction
[B, L, N, A] = bwboundaries(bw2);%computing the boundries of objects in the binary image
props_bw2 = regionprops(L,'BoundingBox','Area');%computing area and bounding box for objects to detect symbols
figure,imshow(imc);
hold on

%Classification
for l = 1 : N
    if(props_bw2(l).Area > saL)
        %checking if major axis and minor axis of object is almost equal
        if(((props_bw2(l).BoundingBox(3))+10)>(props_bw2(l).BoundingBox(4))
            &&(((props_bw2(l).BoundingBox(3))-10)<(props_bw2(l).BoundingBox(4)))
    }
Calibration of prepared environment for optical navigation

Appendix: Documentation of own developed program code

Jinnu Panilet Panipichai

2015-12-14

%obtaining bounding box for the symbol
h1(sn) = round(props_bw2(l).BoundingBox(1,1));
h2(sn) = round(props_bw2(l).BoundingBox(1,2));
h3(sn) = round(props_bw2(l).BoundingBox(1,3));
h4(sn) = round(props_bw2(l).BoundingBox(1,4));
temp_sym = imc(h2(sn):(h2(sn)+h4(sn)),h1(sn):(h1(sn)+h3(sn))); %copping the symbol
sym = bwareaopen(temp_sym, bwL); %removing noise inside bounding box but outside the actual symbol
pur_sym(h2(sn):(h2(sn)+h4(sn)),h1(sn):(h1(sn)+h3(sn))) = sym; %background is eliminated to calculate SNR
symG{sn} = gr(h2(sn):(h2(sn)+h4(sn)),h1(sn):(h1(sn)+h3(sn))); %symbol is cropped in grayscale images
[ROI, ob] = bwlabel(sym); %labeling the dots in the symbols

if(ob == 11) %checking if there are 11 objects in the symbol including the outer circle
rectangle('Position',[h1(sn),h2(sn),h3(sn),h4(sn)],'EdgeColor','r'); %drawing bounding box around the symbol
props_ROI = regionprops(ROI, 'Centroid', 'Area'); %extracting properties of objects in the symbol
props_SP = regionprops(sym, symG{sn}, {'Centroid', 'WeightedCentroid', 'PixelValues'}); %extracting prop from grayscale image to increase subpixel precision
for obj = 1 : ob
   da(obj) = props_ROI(obj).Area; %finding the area of dots
   GrMean(sn, obj) = mean(cat(1, props_SP(obj).PixelValues)); %mean of pixels covered by dots to calculate SNR
   GrStd(sn, obj) = std(double(cat(1, props_SP(obj).PixelValues))); %standard deviation of pixels covered by dots to calculate SNR
   %finding the two big dots in the symbol
   if(da(obj)>sdL) && (da(obj)<bdL)
      cx(obj) = props_SP(obj).WeightedCentroid(1); %x coordinate of centroid of big dots
      cy(obj) = props_SP(obj).WeightedCentroid(2); %y coordinate of centroid of big dots
   end
end
coordinate of centroid of big dots
% finding the center dot
if(cy(obj)>=(h4(sn)/2)-10) && (cy(obj)<=(h4(sn)/2)+10))
cnty(sn) = cy(obj) + h2(sn); % y coordinate of centroid of
dots in the image
plot(cntx(sn),cnty(sn),'x','color','g'); % marking green
colour 'x' over the center dot
end
% finding the corner dot
else
corx(sn) = cx(obj) + h1(sn); % x coordinate of centroid of
dots in the image
cory(sn) = cy(obj) + h2(sn); % y coordinate of centroid of
dots in the image
plot(corx(sn),cory(sn),'x','color','b'); % marking blue
colour 'x' over the corner dot
end
% finding the small dots in the symbol
elseif(da(obj)<=sdL)
scx(sn,dn) = props_SP(obj).WeightedCentroid(1) + h1(sn); % x coordinate of centroid of dots in the image
scy(sn,dn) = props_SP(obj).WeightedCentroid(2) + h2(sn); % y coordinate of centroid of dots in the image
plot(scx(sn,dn),scy(sn,dn),'x','color','r'); % marking red
colour 'x' over the small sots
dn = dn + 1;
end
end
% arrange the small dots and get the code in the symbol
[asx(sn,:), asy(sn,:), tmp_code] = decod(cntx(sn), cnty(sn),
corx(sn), cory(sn), scx(sn,:), scy(sn,:));
SymCodes = [SymCodes tmp_code]; % if there are multiple
symbols, it gets stored here
end
sn = sn+1; % if there are more than one symbol,
dn = 1;
end
end
end
%SNR calculation
pur_sym = uint8(pur_sym);
snr = SNR_calc(pur_sym, gr, GrMean, GrStd, sn);

theta = (atand((corx(1)-cntx(1))/(cory(1)-cnty(1))))%camera rotation

pwd = sw/inc;%X axis distance represented by 1 pixel
pht = sh/inr;%Y axis distance represented by 1 pixel

%S matrix multiplied by (1/S44)
S = [cosd(theta); -sind(theta); (X0/((Z0/fo)+1)); sind(theta); cosd(theta);
(Y0/((Z0/fo)+1))];

%finding the geographical location of center dots of symbols
center_x = (cntx-(inc/2)) * pwd;
center_y = (cnty-(inr/2)) * pht;
%H matrix or A inverse matrix for Center dots of the symbols
for i = 1 : 2*(sn-1)
    j = int8(i/2);
    if(rem(i,2)==1)
        H(i,1) = center_x(j);  %  H = [x1 y1 1 0 0 0
        H(i,2) = center_y(j);  %  0 0 0 x1 y1 1
        H(i,3) = 1;            %  x2 y2 1 0 0 0
        H(i,4) = 0;            %  0 0 0 x2 y2 1
        H(i,5) = 0;            %  . . .
        H(i,6) = 0;            %  . . .
    elseif(rem(i,2)==0)
        H(i,1) = 0;            %  xn yn 1 0 0 0
        H(i,2) = 0;            %  0 0 0 xn yn 1
        H(i,3) = 0;
        H(i,4) = center_x(j);
        H(i,5) = center_y(j);
        H(i,6) = 1;
    end
end
Center_XY = (((Z0/fo)+1)*(H * S));%(1/S44)*(H*S)
%X and Y coordinates of center dots are separated
for p = 1 : 2*(sn-1)
    if(rem(p,2)==1)
Calibration of prepared environment for optical navigation

Jinnu Panilet Panipichai

Appendix: Documentation of own developed program code

2015-12-14

Center_X(round(p/2)) = Center_XY(p);
elseif(rem(p,2)==0)
    Center_Y(round(p/2)) = Center_XY(p);
end
end

Center_XY = [Center_X Center_Y]

% finding the geographical location of corner dots of symbols
corner_x = (corx-(inc/2)) * pwd;
corner_y = (cory-(inr/2)) * pht;
%H matrix or A inverse matrix for Corner dots of the symbols
for i = 1 : 2*(sn-1)
    j = int8(i/2);
    if(rem(i,2)==1)
        HC(i,1) = corner_x(j);
        HC(i,2) = corner_y(j);
        HC(i,3) = 1;
        HC(i,4) = 0;
        HC(i,5) = 0;
        HC(i,6) = 0;
    elseif(rem(i,2)==0)
        HC(i,1) = 0;
        HC(i,2) = 0;
        HC(i,3) = 0;
        HC(i,4) = corner_x(j);
        HC(i,5) = corner_y(j);
        HC(i,6) = 1;
    end
end

Corner_XY = (((Z0/fo)+1)*(HC * S));%(1/S44)*(H*S)
%X and Y coordinates of corner dots are separated
for p = 1 : 2*(sn-1)
    if(rem(p,2)==1)
        Corner_X(round(p/2)) = Corner_XY(p);
    elseif(rem(p,2)==0)
        Corner_Y(round(p/2)) = Corner_XY(p);
    end
end
Corner_XY = [Corner_X Corner_Y]

% finding the geographical location of small dots of symbols
%H matrix or A inverse matrix for Small dots of the symbols
for r = 1 : (sn-1)
    for i = 1 : 16
        j = int8(i/2);
        if(rem(i,2)==1)
            CH(i,1) = (asx(r,j)-(inc/2)) * pwd;
            CH(i,2) = (asy(r,j)-(inr/2)) * pht;
            CH(i,3) = 1;
            CH(i,4) = 0;
            CH(i,5) = 0;
            CH(i,6) = 0;
        elseif(rem(i,2)==0)
            CH(i,1) = 0;
            CH(i,2) = 0;
            CH(i,3) = 0;
            CH(i,4) = (asx(r,j)-(inc/2)) * pwd;
            CH(i,5) = (asy(r,j)-(inr/2)) * pht;
            CH(i,6) = 1;
        end
    end
end
Small_dots_XY(:,r) = (((Z0/fo)+1)*(CH * S));%(1/S44)*(H*S)
end

%X and Y coordinates of small dots are separated
for rr = 1 : (sn-1)
    for p = 1 : 16
        if(rem(p,2)==1)
            Small_dots_X((round(p/2)),rr) = Small_dots_XY(p,rr);
        elseif(rem(p,2)==0)
            Small_dots_Y((round(p/2)),rr) = Small_dots_XY(p,rr);
        end
    end
end
Small_dots_XY = [Small_dots_X Small_dots_Y]
function [snr] = SNR_calc(objects, gray_img, gray_mean, gray_std, symb_num)
    % calculates the SNR of the image
    objects = double(objects); % individual dots in the symbol
    bin_img = xor(imdilate(objects,strel('disk',7)),
                imdilate(objects,strel('disk',20))); % foreground image
    bg_img = gray_img .* bin_img; % background image
    [rows, cols, values] = find(bg_img);
    bg_mean = mean(values);
    bg_Std = std(values);
    signal = gray_mean - bg_mean;
    noise = sqrt(power(gray_std,2) + power(bg_Std,2));
    snrlist = 20*log10(signal./noise); % SNR of individual dots
    for i = 1 : (symb_num - 1)
        snrl(i) = sum(snrlist(i,:)); % sum of SNR of dots
    end
    snr = (sum(snr))/((11*(symb_num - 1)); % mean of SNR of dots

function [arr_x, arr_y, Code] = decod(x0, y0, xc, yc, x, y)
    % This function arranges the small dots in descending order of their angles
    % from the center dot of the symbol and calculates the code encoded in the symbol
    Code = 0; % Symbol code initialization

    % finding the distance and angle between center and corner dots------
    ref_dx = xc - x0;
    ref_dy = y0 - yc;
    ref_dist = sqrt(((ref_dx)^2) + ((ref_dy)^2));
    ref_angl = atand(ref_dy/ref_dx);
    ref_ang = angl(ref_angl, ref_dx, ref_dy);
    % ---------------------------------------------------------------

    % finding angle between center dot and small dots--------------
    for a = 1 : 8
        dx(a) = x(a) - x0;
    end
dy(a) = y0 - y(a);
dot_ang(a) = round(atand(dy(a)/dx(a)));
ang(a) = angl(dot_ang(a), dx(a), dy(a));
end
% ---------------------------------------------------

% ---finding angles with reference to corner dot to find the symbol code--
for b = 1:8
    if ang(b) <= ref_ang
        fnl_ang(b) = ref_ang - ang(b);
    else
        fnl_ang(b) = ref_ang - ang(b) + 360;
    end
end
temp = fnl_ang;
% ---------------------------------------------------

% ----------arranging the symbols in descending order of angles-----------
for c = 1:8
    k = find(temp==max(temp));
    arr_x(c) = x(k);
    arr_y(c) = y(k);
    temp(:,k) = -1;
end
% ---------------------------------------------------

% finding the distance between small dots and center dots to know whether
% the small dots represent '0' bit or '1' bit in the symbol code
% ---------------------------------------------------
for l = 1:8
    dist(l) = sqrt(((x0-arr_x(l))^2) + (y0-arr_y(l))^2);
    if dist(l) >= 0.5 * ref_dist
        Code = Code + (1 * 2^(l-1));%code encoded in the symbol
    end
end
% ---------------------------------------------------
function [ang] = angl(fyi, dx, dy)
% finds the angle between small dots and the center dot
% We know that the angles must be in the multiples of 45. So, the positions
% of small dot and center dot are compared to each other. The calculated
% angle between the center and small dots are also used to find the
% original angle between the center dot and small dots in one direction
if (fyi >= -10 && fyi <= 10)
    if dx > 0
        ang = 0;
    else
        ang = 180;
    end
elseif (abs(fyi) >= 80 && abs(fyi) <= 100)
    if dy > 0
        ang = 90;
    else
        ang = 270;
    end
else
    if (dx > 0)
        if (dy > 0)
            ang = 45;
        else
            ang = 315;
        end
    else
        if (dy > 0)
            ang = 135;
        else
            ang = 225;
        end
    end
end