Vision Based Bayesian State Estimation of Unmanned Aerial Vehicles – A Preliminary Study

Pradeep Papanna Nanjappa

B.E., Mechanical Engineering. Bangalore University (1997)

A thesis (dissertation) presented to the faculty of the

OGI School of Science & Engineering

at Oregon Health & Science University

in partial fulfillment of the

requirements for the degree

Master of Science

in

Electrical and Computer Engineering

June 2003

Dedicated to my Parents

The dissertation "Vision Based Bayesian State Estimation of Unmanned Aerial Vehicles – A Preliminary Study" has been examined and approved by the following examination committee:

Dr. Xubo Song
Professor
Thesis Research Advisor

Dr. Robert Jaffe
Professor

Dr. John-Paul Hosom
Professor

Acknowledgements

I would like to extend my hearty thanks to my advisor, Dr.Xubo Song, for her support and guidance. Her encouragement and advice was key to seeing me through this work. She was most patient and enduring with my queries and doubts about the material. She was instrumental in my understanding of the background material and concepts. Thank you Dr.Song, for giving me this great opportunity where I have learnt much more than what has gone into this report.

Dr.Bob Jaffe has been more a mentor and a philosophical guide than my professor. I am deeply indebted to him for what I know today in many subjects. My gratitude to him is definitely beyond words.

My gratitude to Dr.Paul Hosom for being on the defense committee for this thesis and overseeing my work.

Finally and most importantly, I wish to thank my parents who have been the backbone of my ambition. Without their support I would not have been where I am.

Table of Contents

List of Tables

List of Figures

Abstract

Vision Based Bayesian State Estimation of Unmanned Aerial Vehicles – A Preliminary Study

Pradeep Papanna Nanjappa

M.S., Oregon Health & Science University

June 2003

Thesis Advisor: Dr. Xubo Song.

The goal of this study was to arrive at a method to estimate the physical state of a scaled helicopter model using images taken from a stationary camera on the ground. The state would consist of the [XYZ] coordinates and orientation of the helicopter in space. The end was achieved by using the Direct Linear Transform to calibrate the given camera and obtain the transformation matrices and camera calibration parameters. By using these results, the ideal projection is calculated. And by using the methods below, the actual projection was obtained. The difference between these projections was modeled as a Gaussian distribution and plugged into a Bayesian framework to estimate the actual state of the helicopter. The different schemes tried out for locating the actual projections of the model as mentioned below gave a range of possibilities and results. Color coded markers placed on the helicopter at strategic locations was one of the schemes. Pictures were taken of the thus prepared helicopter and the color information in the image was used to automatically recognize the markers as features on the object. These color images were processed by suitably thresholding the RGB values which make up the color information of the individual pixels to locate the positions of the markers in the images. Another scheme that was used was the application of the Harris Corner Detector to detect the

positions of a completely different set of markers. The third method tried out to locate the positions of the markers in the image was Template Matching. All three methods are for locating point features in the images. Although with a lesser degree of accuracy, these methods allow us the flexibility of estimating the ground truth of the helicopter visually without actually using any onboard instruments like the gyroscope or a GPS receiver to do so.

Chapter 1
Introduction

1.1 Introduction

A Bayesian approach to integrating estimation, image processing and control is attempted. The controller requires an estimate of the state of the system x_k, corresponding to the position, attitude, velocities as well as any physical model parameters. The basic idea is to propagate an estimate of the density of the state given a sequence of images $I_0^{\,k} = \{I_0, I_1, I_2, ..., I_k\}$,

$$P(x_k \mid I_0^{\,k}) = \frac{P(x_k \mid I_0^{\,k-1}) \cdot P(I_k \mid x_k)}{P(I_k \mid I_0^{\,k-1})}$$

where

$$P(x_k \mid I_0^{\,k-1}) = \int P(x_k \mid x_{k-1}) \cdot P(x_{k-1} \mid I_0^{\,k-1}) dx_{k-1}$$

and the normalizing constant $P(I_k \mid I_0^{\,k-1})$ is given by,

$$P(I_k \mid I_0^{\,k-1}) = \int P(x_k \mid I_0^{\,k-1}) \cdot P(I_k \mid x_k) dx_k$$

$P(x_k \mid I_0^{\,k})$ - Probability of the state, given image.

$P(x_k \mid I_0^{\,k-1})$ - Probability of the state, given previous image.

$P(I_k \mid x_k)$ - Probability of the image, given the state.

$P(I_k \mid I_0^{\,k-1})$ - Probability of the image, given the previous image.

This recursion specifies the current state density as a function of the previous density and the most recent (observed) data. The observation density $P(I_k \mid x_k)$ represents the 'image

likelihood' and describes the probability of observing the image given the current state. This relates to the geometry of the vehicle projected onto the image plane. The probability of the state given the image is what is achieved in this work.

This is done in the following sections.

1. Preparing the model with suitable markers for detection.
2. Coding the Direct Linear Transform in Matlab.
3. Calibrating the camera used in the experiment.
4. Processing the images to locate the markers.

The following equation gives the observation density modeled as a Gaussian distribution.

$$P(I \mid x) = \exp\left(-\frac{1}{\sigma^2} \sum_{i=0}^{n-1} \left[p(i) - p'(i) \right]^2 \right)$$

Where,

$p'(i)$ is the detected image coordinate.

$P(i)$ is the actual image coordinate.

So $P(I \mid x)$ will be the probabilistic deviation between the following –

1. Ideal Observation given the state.
2. Actual Observation.

The procedure of calibrating the camera, deriving the transformation matrices and applying them to get the projection of the point features will give the Ideal Observation. And the process of detecting the markers will give the Actual Observation.

All the coding for this work was done using Matlab. Thus only the development of the schemes was in focus. The real time implementation of these procedures is an area to be explored.

1.2 Background

The material presented in this section is the brief summary of the study undertaken to arrive at the procedures covered in the following sections.

When first the outline of the problem was presented by my advisor, it seemed to be a very interesting project to work on. Starting to work on it, one of the first tasks was to cut down the problem into as many subsections as possible and deal with the subsections individually. It was decided to first break up the task into two major portions. The first work needed to be done was to calibrate the camera which was available for the purpose, and the next task was to somehow get an estimate of the observed position of the flying helicopter. To do this it was necessary to arrive at some scheme to represent the contours of the helicopter in a mathematical way so as to facilitate obtaining the ground truth. It was clear that if some prominent contours of the helicopter like, for example, the cockpit outline were discernable enough, then such surface contours could be represented mathematically using piecewise cubic polynomials or B-Splines or some active contour models like Snakes. And once the contours were parameterized in this way, then to obtain the respective projections on any plane in 3D space was not a difficult problem. The projected polynomial curves were meant to be matched with the new polynomial curves obtained from the latest image frame and to find a match between them.

These ideas were investigated and a lot of practical limitations were encountered. The task of preparing the piecewise polynomials for representation of the complex contours of the helicopter cockpit and other surface curves and to make the process automated was beyond the scope of this project time frame. The computational complexities would not have permitted the procedure to be anywhere near 'Real-Time' capable. Most of the time, the helicopter was flown without the cockpit casing without which there was practically no regular surface contour which could be represented tractably by mathematical means.

So it was necessary to adopt a different approach to recognize the helicopter in any given image. Methods in literature (Gatrell, L., Hoff, W. and Sklair, C. (1991), Cho, Y., Lee, W.J. and Neumann, U. (1998), Sklair, C., Hoff, W. and Gatrell, L. (1991), Leonid Naimark & Eric Foxlin (2002)) use fiducial markers for similar purposes. This idea was adapted to the present problem.

1.3 Preparation of the model

The scaled model of the helicopter was prepared to enable the images to be processed to obtain the location of certain points on the helicopter. These strategic points, for reasons pertaining to the DLT method [Shapiro, R. (1978)], were chosen to be non-collinear and so as to form a volume on the model. Six such control points were marked using Red colored tape. A color picture is shown below.

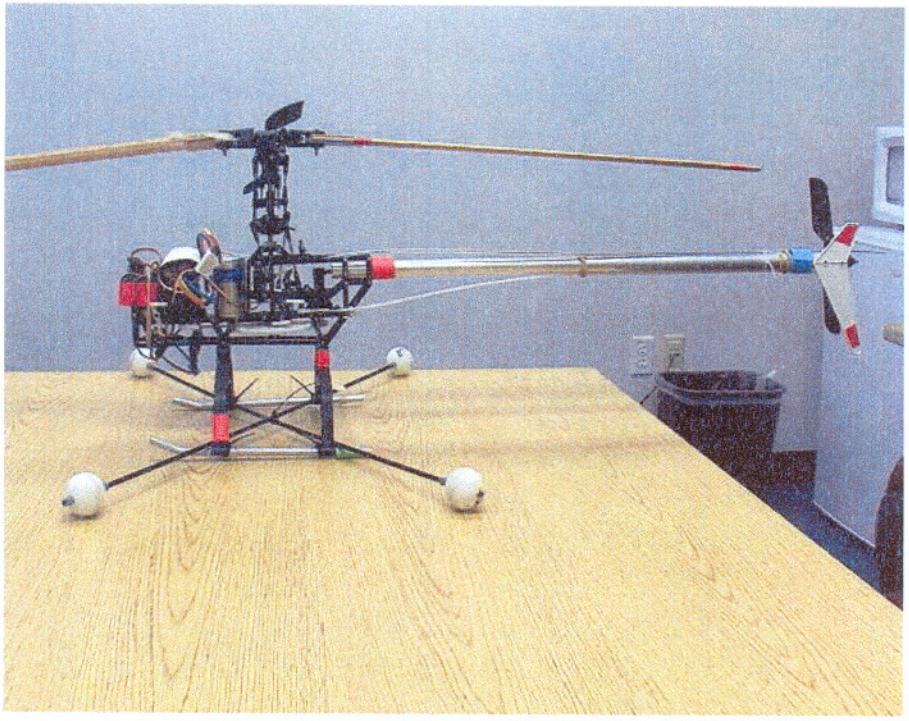

Figure 1.3.1: Helicopter prepared with the control points seen in red.

A second set of markers which does not use the color information was prepared and positioned on the helicopter as shown in the figure.

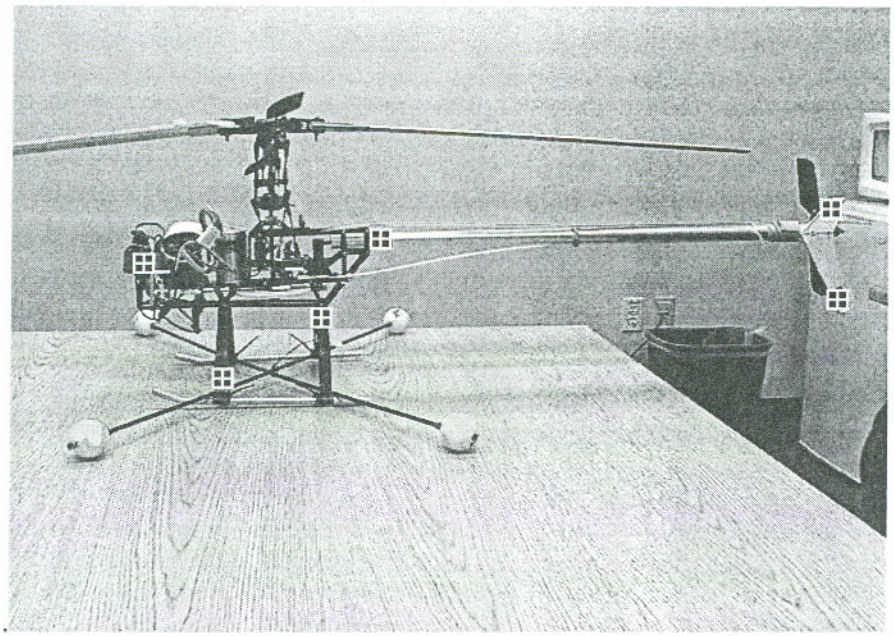

Figure 1.3.2: Model with a different set of markers for Corner Detection

Also shown next is the scale drawing which was made of the helicopter to enable the correct measurement of the coordinates of the markers on the helicopter.

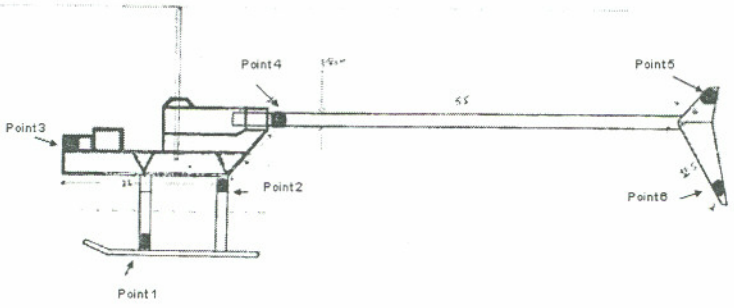

Figure1.3.3: Scale drawing of the Helicopter Skeleton (side view)

This scale drawing is used to obtain the exact physical coordinates of any point on the model. Now we obtain the XYZ coordinates of each of the control points and note them for the purpose of calibration.

The origin is chosen to be the point of the bend in the landing strut of the helicopter. The six points chosen on the body are shown in the figure above. The XYZ coordinates of all points are tabulated below.

	X cms	Y cms	Z cms
Origin	0	0	0
Point 1	5	2.5	0.5
Point 2	16	10	6.5
Point 3	-4	16	8
Point 4	23	18.5	8.5
Point 5	81	23	7.5
Point 6	83	10	7.5

Table 1.3.1: XYZ coordinates of the six Control Points on the model

Chapter 2
Camera Calibration

Camera Calibration [R. Y. Tsai (1987), T.A. Clarke and J.G. Fryer (1998), Z. Zhang (2000), Z. Zhang (1999)] in a broad sense is the procedure carried out to give us the relationships between the real world coordinate system and the camera coordinate system. In other words, camera calibration enables us to obtain the focal length, image aspect ratios and their scaling ratios, i.e., how many pixels is a definite distance measure in the object space converted to in the image plane, and also it enables us to understand how the camera is aligned, in terms of tilt and distance, with respect to the object world. The Direct Linear Transform discussed below is a method which carries out the camera calibration. The DLT works by defining a world coordinate reference frame and a camera coordinate reference frame and distills the implicit relationships between various intrinsic and extrinsic properties in the framework into eleven DLT parameters. As explained later, six known points are chosen on the model. These six points' XYZ coordinates are measured and fixed. Then the DLT is applied using these six known points to get the eleven DLT parameters. These eleven DLT parameters when solved, give the intrinsic camera parameters like the focal length, image scaling factors and location of the principal point. The eleven DLT parameters are also solved to obtain extrinsic camera parameters like the transformation matrices.

2.1 Direct Linear Transform

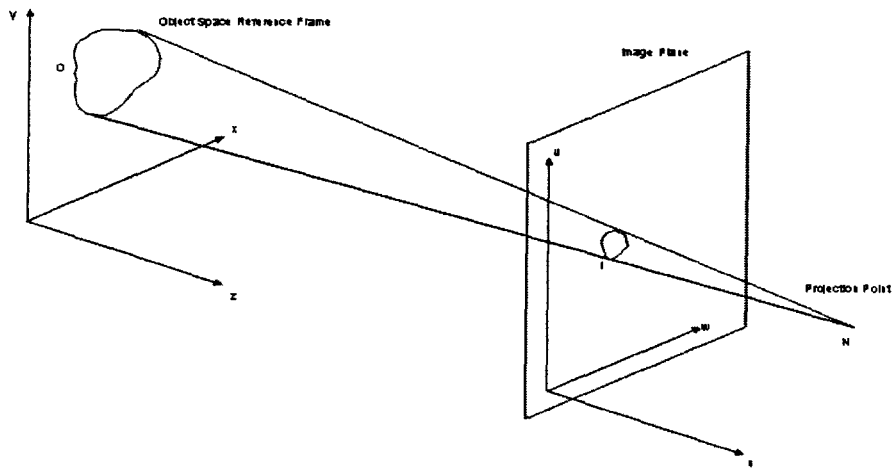

Figure 2.1.1: Line drawing showing the DLT reference frames

Two reference frames are defined. Object Space Reference Frame (XYZ-system) and the Image Plane Reference Frame (UV-system). The camera maps a point O in the object space to image I in the image plane. [x,y,z] is the object space coordinates of point O. [u,v] is the image plane coordinates of the image I. Assuming another axis W in the image plane perpendicular to the plane, [u,v,w] is the image space reference frame. N is the projection center. In the object space reference frame, the coordinates of N will be assumed to be $[x_0, y_0, z_0]$.

A vector 'A' drawn from N to O will be $[x - x_0, y - y_0, z - z_0]$. The 3-d position of point I in the image space reference frame will be $[u,v,0]$. P is the principal point. The line drawn from the projection center N to the image plane parallel to the w-axis and perpendicular to the image plane is called the principal axis. The principal point is the intersection of the principal axis with the image plane. The principal distance d is the distance between points P&N.

If the image plane coordinates of the principal point are $[u_0, v_0, 0]$, the position of point N in the image space reference frame becomes $[u_0, v_0, d]$. A vector 'B' drawn from point N to I is $[u-u_0, v-v_0, -d]$. Since points O, I & N are collinear, vectors A & B form a single straight line which is equivalent to the vector expression,

$$B = cA \qquad \text{- (1)}$$

where c is a scaling factor.

A & B are originally described in the object space reference frame and Image space reference frame, respectively. In order to directly relate the coordinates, it is necessary to describe them in a common reference frame which is chosen to be the image space reference frame.

To do this, we assume a transformation matrix consisting of the euclidean angles which is

$$T = T_x(\phi) \cdot T_y(\psi) \cdot T_z(\theta)$$

where -

$$T_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

$$T_y(\psi) = \begin{bmatrix} \cos\psi & 0 & \sin\psi \\ 0 & 1 & 0 \\ -\sin\psi & 0 & \cos\psi \end{bmatrix}$$

$$T_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} \cos\psi\cos\theta & -\cos\phi\sin\theta + \sin\phi\sin\psi\cos\theta & \sin\phi\sin\theta + \cos\phi\sin\psi\cos\theta \\ \cos\psi\sin\theta & \cos\phi\cos\theta + \sin\phi\sin\psi\sin\theta & -\sin\phi\cos\theta + \cos\phi\sin\psi\sin\theta \\ -\sin\psi & \sin\phi\cos\psi & \cos\phi\cos\psi \end{bmatrix} \quad \text{- (2)}$$

the above matrix T can be represented as

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Applying this transformation to the A matrix in (1),

$$A_i = T \bullet A$$

$$B = cA_i$$

$$\begin{bmatrix} u - u_0 \\ v - v_0 \\ -d \end{bmatrix} = c \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \bullet \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix} \quad \text{- (3)}$$

$$\begin{aligned} u - u_0 &= c\left[r_{11}(x - x_0) + r_{12}(y - y_0) + r_{13}(z - z_0)\right] \\ v - v_0 &= c\left[r_{21}(x - x_0) + r_{22}(y - y_0) + r_{23}(z - z_0)\right] \\ -d &= c\left[r_{31}(x - x_0) + r_{32}(y - y_0) + r_{33}(z - z_0)\right] \end{aligned} \quad \text{- (4)}$$

from (4), we obtain

$$c = \frac{-d}{r_{31}(x - x_0) + r_{32}(y - y_0) + r_{33}(z - z_0)} \qquad \text{- (5)}$$

By substituting (5) for c in (4), we get

$$\begin{aligned} u - u_0 &= -d \frac{r_{11}(x - x_0) + r_{12}(y - y_0) + r_{13}(z - z_0)}{r_{31}(x - x_0) + r_{32}(y - y_0) + r_{33}(z - z_0)} \\ v - v_0 &= -d \frac{r_{21}(x - x_0) + r_{22}(y - y_0) + r_{23}(z - z_0)}{r_{31}(x - x_0) + r_{32}(y - y_0) + r_{33}(z - z_0)} \end{aligned} \qquad \text{- (6)}$$

u, v, u_0 & v_0 are the image plane coordinates in actual units. The camera digitization system uses different length units. So to accommodate this,

$$\begin{aligned} u - u_0 &\Rightarrow \lambda_u (u - u_0) \\ v - v_0 &\Rightarrow \lambda_v (v - v_0) \end{aligned}$$

where $[\lambda_u, \lambda_v]$ are the unit conversion factors for the u and v axes.

$$\begin{aligned} u - u_0 &= -\frac{d}{\lambda_u} \frac{r_{11}(x - x_0) + r_{12}(y - y_0) + r_{13}(z - z_0)}{r_{31}(x - x_0) + r_{32}(y - y_0) + r_{33}(z - z_0)} \\ v - v_0 &= -\frac{d}{\lambda_v} \frac{r_{21}(x - x_0) + r_{22}(y - y_0) + r_{23}(z - z_0)}{r_{31}(x - x_0) + r_{32}(y - y_0) + r_{33}(z - z_0)} \end{aligned}$$

rearranging for u, v & x, y, z we get

$$u = \frac{L_1 x + L_2 y + L_3 z + L_4}{L_9 x + L_{10} y + L_{11} z + 1}$$
$$v = \frac{L_5 x + L_6 y + L_7 z + L_8}{L_9 x + L_{10} y + L_{11} z + 1} \qquad \text{- (8)}$$

if $d_v = \frac{d}{\lambda_v}$, $d_u = \frac{d}{\lambda_u}$ and $D = -\left(x_0 r_{31} + y_0 r_{32} + z_0 r_{33}\right)$, then the coefficients $L_1 - L_{11}$ are given as –

$$L_1 = \frac{u_0 r_{31} - d_u r_{11}}{D}$$

$$L_2 = \frac{u_0 r_{32} - d_u r_{12}}{D}$$

$$L_3 = \frac{u_0 r_{33} - d_u r_{13}}{D}$$

$$L_4 = \frac{(d_u r_{11} - u_0 r_{31}) x_0 + (d_u r_{12} - u_0 r_{32}) y_0 + (d_u r_{13} - u_0 r_{33}) z_0}{D}$$

$$L_5 = \frac{v_0 r_{31} - d_v r_{21}}{D}$$

$$L_6 = \frac{v_0 r_{32} - d_v r_{22}}{D}$$

$$L_7 = \frac{v_0 r_{33} - d_v r_{23}}{D}$$

$$L_8 = \frac{(d_v r_{21} - v_0 r_{31})x_0 + (d_v r_{22} - v_0 r_{32})y_0 + (d_v r_{23} - v_0 r_{33})z_0}{D}$$

$$L_9 = \frac{r_{31}}{D}$$

$$L_{10} = \frac{r_{32}}{D}$$

$$L_{11} = \frac{r_{33}}{D}$$

The above coefficients are called the DLT coefficients. DLT stands for Direct Linear Transform.

Rearranging (8), we obtain –

$$\begin{aligned} \frac{1}{R}u &= \frac{1}{R}(L_1 x + L_2 y + L_3 z + L_4 - L_9 ux - L_{10} uy - L_{11} uz) \\ \frac{1}{R}v &= \frac{1}{R}(L_5 x + L_6 y + L_7 z + L_8 - L_9 vx - L_{10} vy - L_{11} vz) \end{aligned} \qquad \text{- (9)}$$

where

$$R = L_9 x + L_{10} y + L_{11} z + 1 \qquad \text{- (10)}$$

(9) can be written as –

$$\frac{1}{R}\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{R}\begin{bmatrix} x & y & z & 1 & 0 & 0 & 0 & 0 & -ux & -uy & -uz \\ 0 & 0 & 0 & 0 & x & y & z & 1 & -vx & -vy & -vz \end{bmatrix}\begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \\ L_5 \\ L_6 \\ L_7 \\ L_8 \\ L_9 \\ L_{10} \\ L_{11} \end{bmatrix} \qquad \text{- (11)}$$

This is for one control point on the object. To solve for the coefficients $L_1 - L_{11}$, we have to have an over determined system. So we expand (11) to include more control points.

Expanding (11) for n control points –

$$\begin{bmatrix} \frac{x_1}{R_1} & \frac{y_1}{R_1} & \frac{z_1}{R_1} & \frac{1}{R_1} & 0 & 0 & 0 & 0 & \frac{-u_1x_1}{R_1} & \frac{-u_1y_1}{R_1} & \frac{-u_1z_1}{R_1} \\ 0 & 0 & 0 & 0 & \frac{x_1}{R_1} & \frac{y_1}{R_1} & \frac{z_1}{R_1} & \frac{1}{R_1} & \frac{-v_1x_1}{R_1} & \frac{-v_1y_1}{R_1} & \frac{-v_1z_1}{R_1} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \frac{x_n}{R_n} & \frac{y_n}{R_n} & \frac{z_n}{R_n} & \frac{1}{R_n} & 0 & 0 & 0 & 0 & \frac{-u_nx_n}{R_n} & \frac{-u_ny_n}{R_n} & \frac{-u_nz_n}{R_n} \\ 0 & 0 & 0 & 0 & \frac{x_n}{R_n} & \frac{y_n}{R_n} & \frac{z_n}{R_n} & \frac{1}{R_n} & \frac{-v_nx_n}{R_n} & \frac{-v_ny_n}{R_n} & \frac{-v_nz_n}{R_n} \end{bmatrix}\begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \\ L_5 \\ L_6 \\ L_7 \\ L_8 \\ L_9 \\ L_{10} \\ L_{11} \end{bmatrix} = \begin{bmatrix} \frac{u_1}{R_1} \\ \frac{v_1}{R_1} \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \frac{u_n}{R_n} \\ \frac{v_n}{R_n} \end{bmatrix} \qquad \text{- (12)}$$

(12) is basically in the form of –

$$X \bullet L = Y \qquad \text{- (13)}$$

$$L = X^{-1}Y \qquad \text{- (14)}$$

This equation can be solved in many different ways to obtain the DLT coefficients $L_1 - L_{11}$.

2.2 Transformation Matrix & Camera Parameters

From the 11 DLT coefficients,

$$\begin{aligned} L_1x_0 + L_2y_0 + L_3z_0 &= -L_4 \\ L_5x_0 + L_6y_0 + L_7z_0 &= -L_8 \\ L_9x_0 + L_{10}y_0 + L_{11}z_0 &= -1 \end{aligned} \qquad \text{- (20)}$$

$$\begin{bmatrix} L_1 & L_2 & L_3 \\ L_5 & L_6 & L_7 \\ L_9 & L_{10} & L_{11} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = \begin{bmatrix} -L_4 \\ -L_8 \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = \begin{bmatrix} L_1 & L_2 & L_3 \\ L_5 & L_6 & L_7 \\ L_9 & L_{10} & L_{11} \end{bmatrix}^{-1} \cdot \begin{bmatrix} -L_4 \\ -L_8 \\ -1 \end{bmatrix} \qquad \text{- (21)}$$

Similarly, again from the 11 DLT coefficients,

$$L_9^{\,2} + L_{10}^{\,2} + L_{11}^{\,2} = \frac{1}{D^2}[r_{31}^{\,2} + r_{32}^{\,2} + r_{33}^{\,2}] = \frac{1}{D^2}$$

$$D^2 = \frac{1}{L_9^{\,2} + L_{10}^{\,2} + L_{11}^{\,2}} \qquad \text{- (22)}$$

$$\begin{aligned} u_0 &= D^2(L_1L_9 + L_2L_{10} + L_3L_{11}) = \frac{L_1L_9 + L_2L_{10} + L_3L_{11}}{L_9^{\,2} + L_{10}^{\,2} + L_{11}^{\,2}} \\ v_0 &= D^2(L_5L_9 + L_6L_{10} + L_7L_{11}) = \frac{L_5L_9 + L_6L_{10} + L_7L_{11}}{L_9^{\,2} + L_{10}^{\,2} + L_{11}^{\,2}} \end{aligned} \qquad \text{- (23)}$$

$$d_u^{\,2} = \frac{(u_0 L_9 - L_1)^2 + (u_0 L_{10} - L_2)^2 + (u_0 L_{11} - L_3)^2}{L_9^{\,2} + L_{10}^{\,2} + L_{11}^{\,2}}$$
$$d_v^{\,2} = \frac{(v_0 L_9 - L_5)^2 + (v_0 L_{10} - L_6)^2 + (v_0 L_{11} - L_7)^2}{L_9^{\,2} + L_{10}^{\,2} + L_{11}^{\,2}} \qquad \text{- (24)}$$

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = D \cdot \begin{bmatrix} \frac{u_0 L_9 - L_1}{d_u} & \frac{u_0 L_{10} - L_2}{d_u} & \frac{u_0 L_{11} - L_3}{d_u} \\ \frac{v_0 L_9 - L_5}{d_v} & \frac{v_0 L_9 - L_6}{d_v} & \frac{v_0 L_9 - L_7}{d_v} \\ L_9 & L_{10} & L_{11} \end{bmatrix} \qquad \text{- (25)}$$

Where,

u_0 & v_0 are the positions of the Principal Point.

d_u & d_v are the Focal lengths.

T is the Transformation Matrix.

2.3 Reconstruction

As a means of verification of the DLT method, it is possible to reconstruct the locations of the control points in space. i.e., their XYZ coordinates. The following procedure does that. The attached Matlab code in the appendix, when executed, shows that this method works to a fairly good accuracy.

Rearranging (8) for x, y, z, we obtain,

$$\begin{aligned}\frac{1}{R}(uL_9 - L_1)x + \frac{1}{R}(uL_{10} - L_2)y + \frac{1}{R}(uL_{11} - L_3)z &= \frac{1}{R}(L_4 - u)\\ \frac{1}{R}(vL_9 - L_5)x + \frac{1}{R}(vL_{10} - L_6)y + \frac{1}{R}(vL_{11} - L_7)z &= \frac{1}{R}(L_8 - v)\end{aligned} \quad \text{- (15)}$$

(15) is equivalent to the matrix expression

$$\frac{1}{R}\begin{bmatrix} uL_9 - L_1 & uL_{10} - L_2 & uL_{11} - L_3 \\ vL_9 - L_5 & vL_{10} - L_6 & vL_{11} - L_7 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{1}{R}\begin{bmatrix} L_4 - u \\ L_8 - v \end{bmatrix} \quad \text{- (16)}$$

Expanding (16) for m camera positions:

$$\begin{bmatrix} \frac{u^{(1)}L_9^{(1)} - L_1^{(1)}}{R^{(1)}} & \frac{u^{(1)}L_{10}^{(1)} - L_2^{(1)}}{R^{(1)}} & \frac{u^{(1)}L_{11}^{(1)} - L_3^{(1)}}{R^{(1)}} \\ \frac{v^{(1)}L_9^{(1)} - L_5^{(1)}}{R^{(1)}} & \frac{v^{(1)}L_{10}^{(1)} - L_6^{(1)}}{R^{(1)}} & \frac{v^{(1)}L_{11}^{(1)} - L_7^{(1)}}{R^{(1)}} \\ \cdots & \cdots & \cdots \\ \frac{u^{(m)}L_9^{(m)} - L_1^{(m)}}{R^{(m)}} & \frac{u^{(m)}L_{10}^{(m)} - L_2^{(m)}}{R^{(m)}} & \frac{u^{(m)}L_{11}^{(m)} - L_3^{(m)}}{R^{(m)}} \\ \frac{v^{(m)}L_9^{(m)} - L_5^{(m)}}{R^{(m)}} & \frac{v^{(m)}L_{10}^{(m)} - L_6^{(m)}}{R^{(m)}} & \frac{v^{(m)}L_{11}^{(m)} - L_7^{(m)}}{R^{(m)}} \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{L_4^{(1)} - u^{(1)}}{R^{(1)}} \\ \frac{L_8^{(1)} - v^{(1)}}{R^{(1)}} \\ \cdots \\ \frac{L_4^{(m)} - u^{(m)}}{R^{(m)}} \\ \frac{L_8^{(m)} - v^{(m)}}{R^{(m)}} \end{bmatrix} \quad \text{- (17)}$$

where $R^{(i)} = L_9{}^{(i)}x + L^{(i)}{}_{10}y + L^{(i)}{}_{11}z + 1$

(17) is basically in the form of –

$$A \bullet x = B \qquad \text{- (18)}$$

$$x = A^{-1}B \qquad \text{- (19)}$$

This equation can be solved in many different ways to obtain the (x, y, z) coordinates.

Chapter 3

Image Processing

The following image processing techniques are investigated here, Corner Detection [H. Asada and M. Brady. (1986)], Template matching by cross correlation and Color Thresholding. The goal of these methods is to facilitate the detection of point features in a given image of the helicopter. By detecting these point features, it is possible for us to obtain the Actual Observation which is required in the Bayesian estimator discussed earlier.

3.1 Harris Corner Detector

The Harris Corner Detector [J. Harris and M. Stephens. (1988)] works as follows. Consider the following matrix

$$M = \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) \\ \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) & \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} \qquad \text{- (1)}$$

where $I(x, y)$ is the grey level intensity of the image. If at a certain point the two eigen values of the matrix M are large, then a small motion in any direction will cause an important change of gray level. This indicates that the point is a corner. The corner response function is given by:

$$C = \det(M) - k(traceM)^2 \qquad \text{- (2)}$$

where k is a parameter set to 0.04 as suggested by Harris. Corners are defined as the local maxima of the cornerness function. To avoid the corners due to image noise, the images are smoothed with a Gaussian filter. The smoothing is done on the images containing the squared derivatives rather than the input images.

The derivatives of the images are obtained using operators such as the Sobel operator or the Prewitt operator shown below.

$$\text{Sobel Operator - } \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\text{Prewitt Operator - } \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

The corner extracted image usually has far more corners then necessary for our use. So it is necessary to suitably threshold the C value to obtain the corners we need. The results are shown in the figures below.

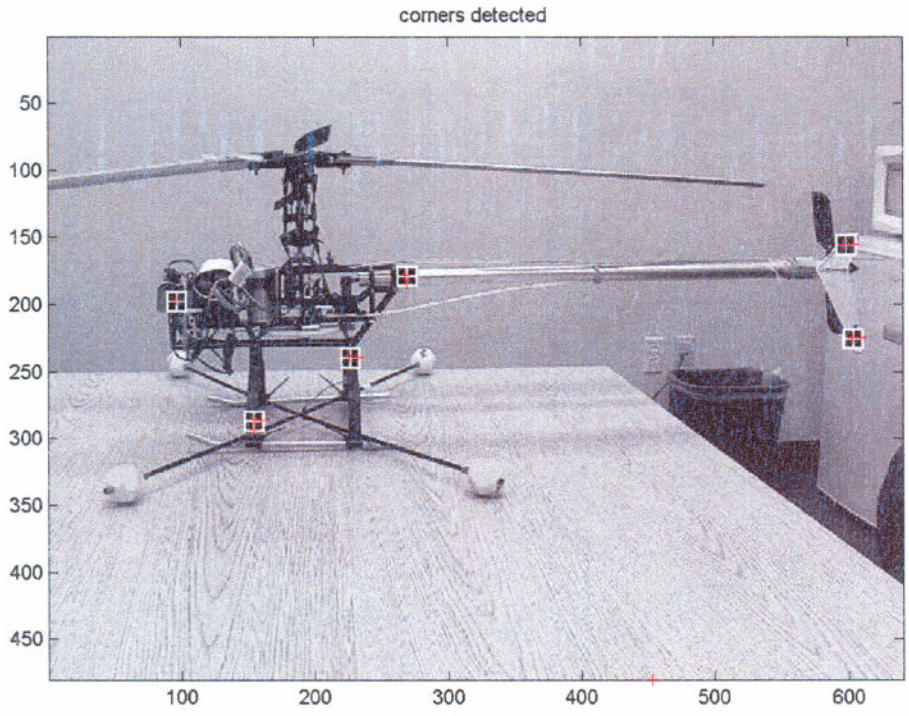

Figure 3.1.1: Harris Corner Detector - Red crosses show the locations of detected corners.

3.1.1 Discussion

This method has the advantages that it is simple to implement and it is robust with respect to the variations in the orientation of the helicopter. It is possible to detect the markers regardless of which direction the model is aligned. The disadvantages being that it is not robust with respect to noise. False corners can easily and incorrectly be detected in the presence of noise in the image. And also it is essential that the lighting and brightness of the image be maintained optimally and constantly in all the frames of the video stream. Lighting affects the cornerness strength detected by the Harris detector. The contrast for the markers is necessary to be maintained.

3.2 Template Matching

Correlation can also be used to locate features within an image; in this context correlation is called template matching. A separate image of the pattern of the marker is prepared or cropped from the input image. These two images are then cross-correlated to obtain peaks at the locations where the markers are located. This method is also fairly effective as shown in the images below. When the helicopter changes its orientation, the template has to be appropriately transformed with the affine transformation for images. Thus the affine transformed template can be rotated by the angles in the allowable range and matched.

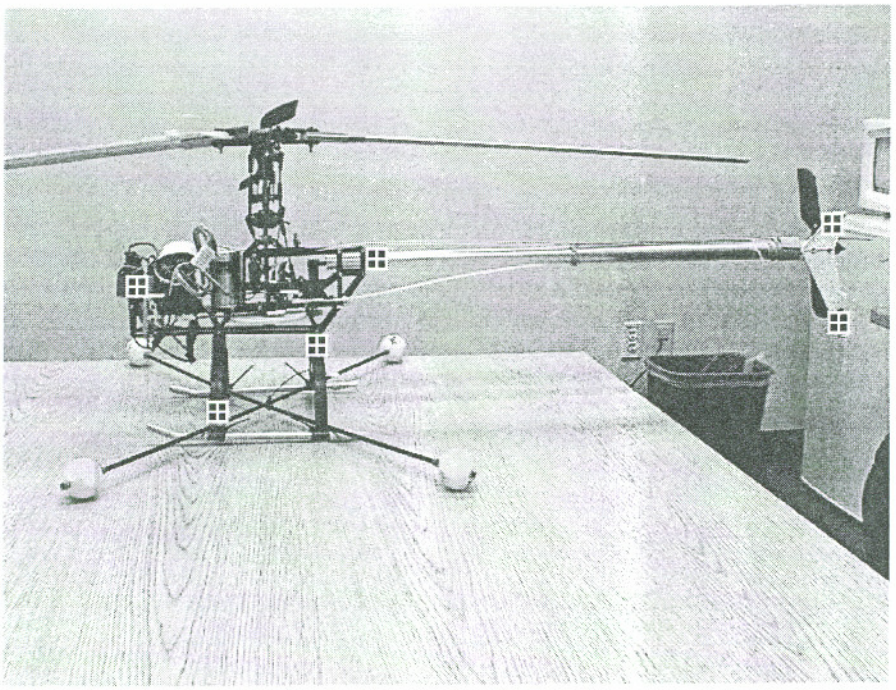

Figure 3.2.1: Original Image for Template Matching

Figure 3.2.2: Enlarged image of the marker used for cross-correlation.

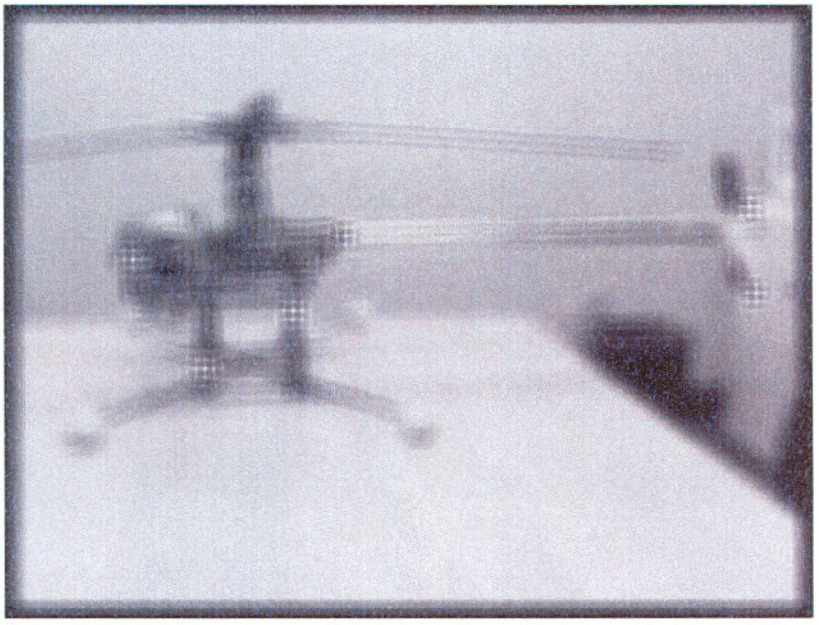

Figure 3.2.3: Cross-correlated image

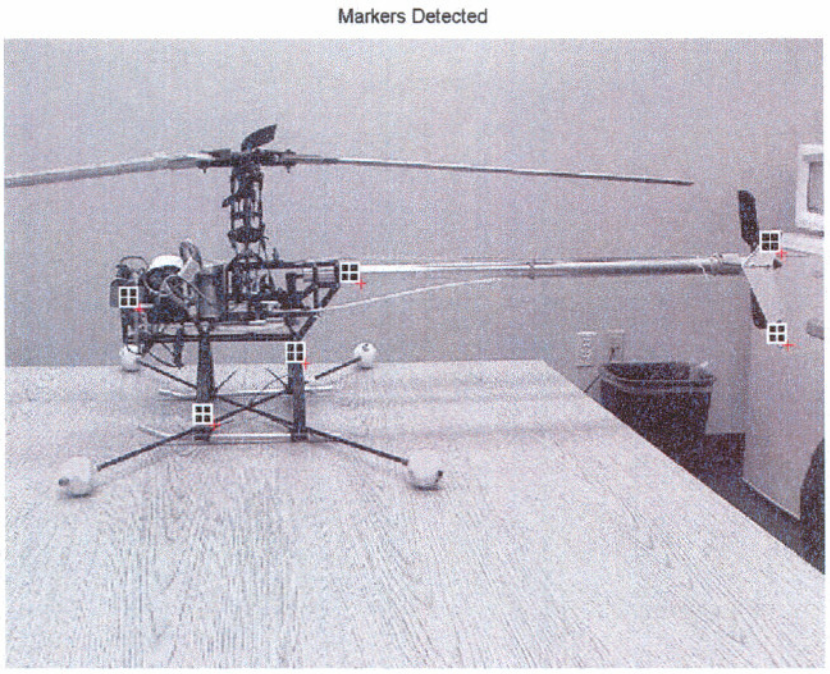

Figure 3.2.4: Template Matching-Red crosses show the locations of the markers detected.

3.2.1 Discussion

Cross-correlation is one of the most intuitive approaches one can think of. The advantage of this method is the simplicity of implementation. The drawback with this scheme is that it can become computationally intensive when we have to match all the spatially transformed images of the marker with the given image. All reasonable possibilities have to be tried before arriving at a suitable match. This process can be made faster by estimating the next state of the model and constraining the possibilities accordingly. The following spatial transforms can be applied to the marker image to make it possible for cross correlation with the helicopter image when it is oriented in different directions.

1. Image rotation.
2. Affine transform for images.

Image rotation by the desired angle is achieved by applying the rotation transform to each of the pixels in the image. The image rotation transform in 2-D is

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where θ is the angle of rotation desired.

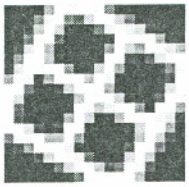

Figure 3.2.1.1: Rotated image of the marker.

Affine transformation is achieved by sequentially applying rotation, translation, and shear on images to obtain the objective.

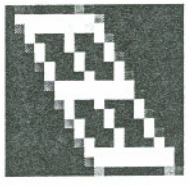

Figure 3.2.1.2: Affine transformed image of the marker.

3.3 Color Thresholding

The idea here is to use the color coding and representation information in the image to achieve our end of locating the markers in the image. Each pixel in a color image is associated with an ordered triplet consisting of the intensities of each of the primary constituent colors, red, green and blue (RGB information). It is possible to access these individual color intensity values from the image pixel. Intuitively, we know that red, green and blue colored markers will have the highest value of R, G & B values respectively. For example a pixel representing a red marker will have high value of R and low values for G and B. So appropriate thresholds are applied for each of the R, G, B values for each pixel to obtain the set of pixels representing the red marker. In a similar manner, thresholding is applied to the image to obtain the blue and green markers too.

The following pictures show the results of the above scheme.

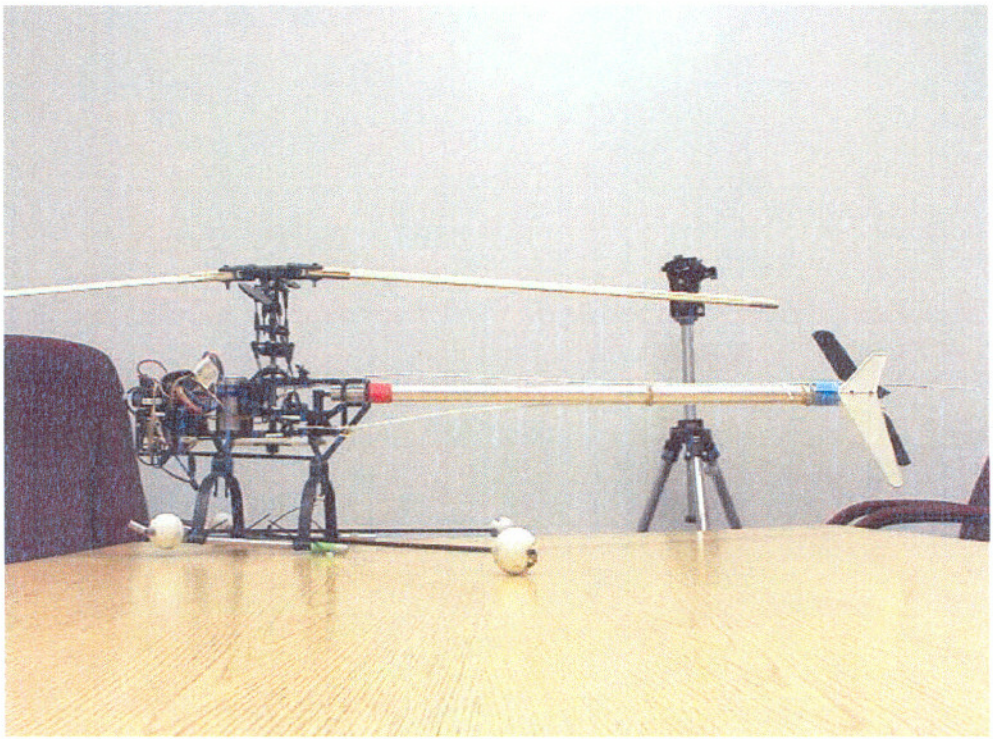

Figure 3.3.1: Picture showing the Red, Green and Blue markers.

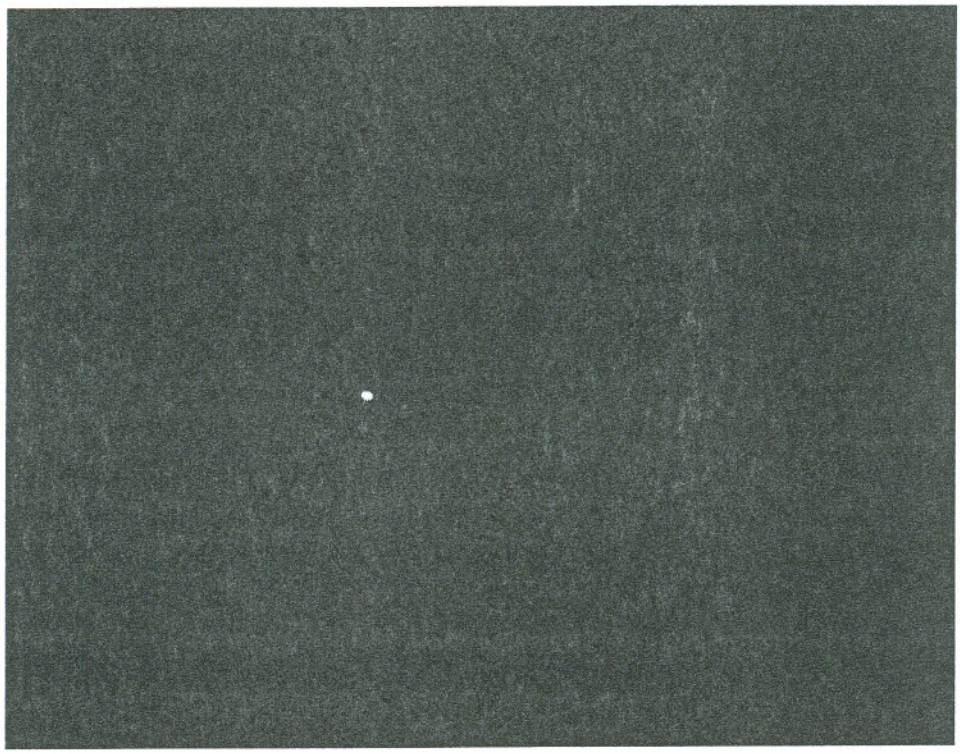

Figure 3.3.2: Image showing the location of the Red marker.

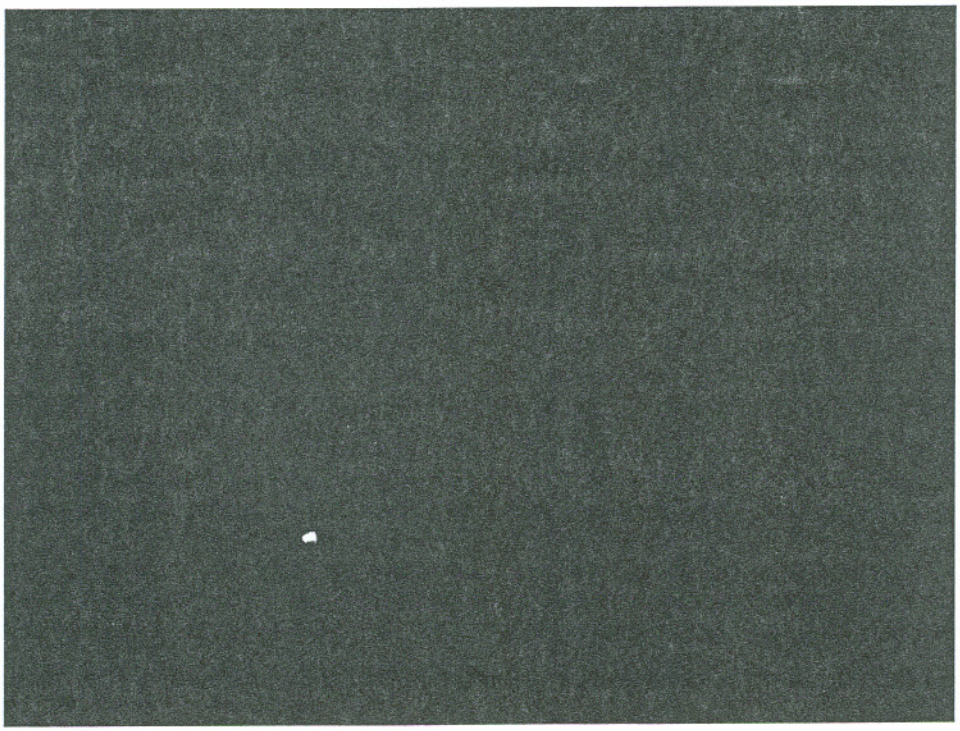

Figure 3.3.3: Image showing the location of the Green marker.

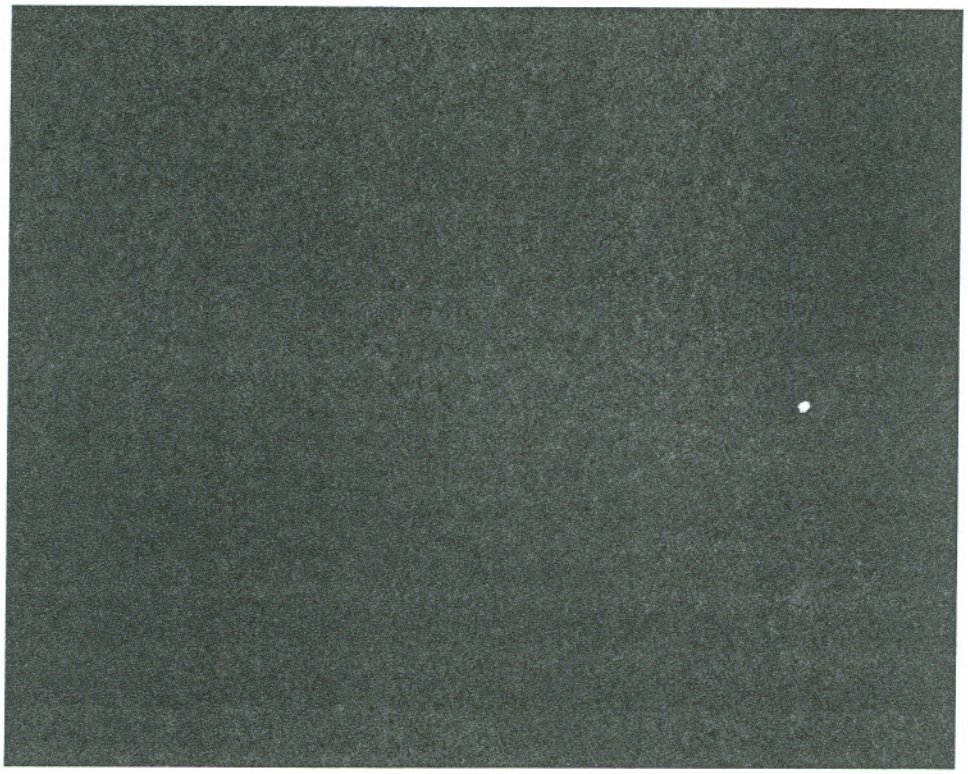

Figure 3.3.4: Image showing the location of the Blue marker.

Chapter 4
Conclusion

The above methods so far discussed, enable us to obtain the observation density value needed to be plugged into the Bayesian estimator.

$$P(x_k \mid I_0^{\,k}) = \frac{P(x_k \mid I_0^{\,k-1}) \cdot P(I_k \mid x_k)}{P(I_k \mid I_0^{\,k-1})}$$

The following equation gives the observation density modeled as a Gaussian distribution.

$$P(I \mid x) = \exp\left(-\frac{1}{\sigma^2} \sum_{i=0}^{n-1} \left[p(i) - p'(i) \right]^2 \right)$$

Where,

$p'(i)$ is the detected image coordinate.

$p(i)$ is the actual image coordinate.

So $P(I \mid x)$ will be the probabilistic deviation between the following –

1. Ideal Observation given the state.
2. Actual Observation.

References

R. Y. Tsai (1987)
A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses -IEEE J. Robotics Automat, pages 323-344, Vol. RA-3, No. 4 1987.

T.A. Clarke and J.G. Fryer (1998)
The Development of Camera Calibration Methods and Models, Photogrammetric Record, 16(91): 51-66, April 1998.

Abdel-Aziz, Y.I., & Karara, H.M. (1971)
Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. Proceedings of the Symposium on Close-Range Photogrammetry (pp. 1-18). Falls Church, VA: American Society of Photogrammetry.

Shapiro, R. (1978)
Direct linear transformation method for three-dimensional cinematography. Res. Quart. 49, 197-205.

Z. Zhang (2000)
A flexible new technique for camera calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11):1330-1334, 2000.

Z. Zhang (1999)
Flexible Camera Calibration by Viewing a Plane from Unknown Orientations.
International Conference on Computer Vision (ICCV'99), Corfu, Greece, pages 666-673, September 1999.

H. Asada and M. Brady. (1986)
The curvature primal sketch.
IEEE Trans. Pattern Analysis and Machine Intelligence, 8:2-14, 1986.

H.L. Beus and S.S.H. Tiu. (1987)
An improved corner detection algorithm based on chain-coded plane curves.
Pattern Recognition, 20:291-296, 1987.

H. Freeman and L.S. Davis.(1977)
A corner finding algorithm for chain-coded curves.
IEEE Trans. Computers, 26:297-303, 1977.

H.C. Liu and M.D. Srinath. (1990)

Corner detection from chain-code.
Pattern Recognition, 23:51-68, 1990.

F. Mokhtarian and A.K. Mackworth. (1992)
A theory of multiscale, curvature-based shape representation for planar curves.
IEEE Trans. Pattern Analysis and Machine Intelligence, 14:789-805, 1992.

Rosenfeld and E. Johnston. (1973)
Angle detection on digital curves.
IEEE Trans. Computers, 22:875-878, 1973.

Rosenfeld and J.S. Weszka. (1975)
An improved method of angle detection on digital curves.
IEEE Trans. Computers, 24:940-941, 1975.

J. Harris and M. Stephens. (1988)
A combined corner and edge detector.
In Proc. 4th Alvey Vision Conf., Manchester, pages 147-151, 1988.

J. Canny. (1986)
A computational approach to edge detection.
IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679-698, 1986

Ravi Samtaney. (1999)
A Method to Solve Interior and Exterior Camera Calibration Parameters for Image Resection, NAS-99-03.

Gatrell, L., Hoff, W. and Sklair, C. (1991)
Robust Image Features: Concentric Contrasting Circles and Their Image Extraction.
SPIE Proc. Conf. Intelligent Robotic Systems,
Boston, Vol. 1612.

Cho, Y., Lee, W.J. and Neumann, U. (1998)
A Multi-ring Color Fiducial System and Intensity-Invariant Detection Method for Scalable Fiducial-tracking Augmented Reality. Proceedings of the 1st Intl. Workshop on Augmented Reality (IWAR 98), San Francisco

Sklair, C., Hoff, W. and Gatrell, L. (1991)
Accuracy of Locating Circular Features Using Machine Vision
SPIE Proc. Conf. Intelligent Robotic Systems, Boston, Vol. 1612.

Leonid Naimark & Eric Foxlin (2002)
Circular Data Matrix Fiducial System and Robust Image Processing for a Wearable Vision-Inertial Self-Tracker. IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2002), Darmstadt, Germany

Rafael C.Gonzalez and Richard E.Woods (1992)
Digital Image Processing. Pearson Education. ISBN: 8178080877. 1992.

E.R.Davies (1996)
Machine Vision: Theory, Algorithms, Practicalities. Academic Press, 2nd Edition. ISBN: 012206092X. November 1996.

Milan Sonka, Vaclav Hlavac, Roger Boyle (1998)
Image Processing: Analysis and Machine Vision, Brooks/Cole Publishing Company 2nd Edition. ISBN: 053495393X. 1998.

Appendix

Matlab Codes for Specific Routines

The following is the Matlab code for the DLT method.

File **DLTvalues.m**

```
% Ground Truth
x1 = 5;      y1 = 2.5;     z1 = 0.5;
x2 = 16;     y2 = 10;      z2 = 6.5;
x3 = -4;     y3 = 16;      z3 = 8;
x4 = 23;     y4 = 18.5;    z4 = 8.5;
x5 = 81;     y5 = 23;      z5 = 7.5;
x6 = 83;     y6 = 10;      z6 = 7.5;

% Picture 1
p1u1 = 26;     p1v1 = 17;
p1u2 = 98;     p1v2 = 64;
p1u3 = -35;    p1v3 = 107;
p1u4 = 140;    p1v4 = 125;
p1u5 = 472;    p1v5 = 150;
p1u6 = 475;    p1v6 = 82;

uvp1 = [26 17;
        98 64;
        -35 107;
        140 125;
        472 150;
        475 82];

% Picture 2
p2u1 = 28;     p2v1 = 18;
p2u2 = 93;     p2v2 = 67;
p2u3 = -42;    p2v3 = 110;
p2u4 = 135;    p2v4 = 128;
p2u5 = 438;    p2v5 = 152;
p2u6 = 441;    p2v6 = 87;

uvp2 = [28 18;
        93 67;
        -42 110;
```

```
        135 128;
        438 152;
        441 87];

% Picture 3
p3u1 = 27;      p3v1 = 12;
p3u2 = 101;     p3v2 = 48;
p3u3 = -4;      p3v3 = 89;
p3u4 = 139;     p3v4 = 101;
p3u5 = 468;     p3v5 = 122;
p3u6 = 477;     p3v6 = 50;

uvp3 = [27 12;
        101 48;
        -4 89;
        139 101;
        468 122;
        477 50];

% Picture 4
p4u1 = 24;      p4v1 = 10;
p4u2 = 93;      p4v2 = 49;
p4u3 = -12;     p4v3 = 94;
p4u4 = 129;     p4v4 = 100;
p4u5 = 455;     p4v5 = 98;
p4u6 = 458;     p4v6 = 32;

uvp4 = [24 10;
        93 49;
        -12 94;
        129 100;
        455 98;
        458 32];

% Picture 5
p5u1 = 25;      p5v1 = 11;
p5u2 = 89;      p5v2 = 55;
p5u3 = -25;     p5v3 = 94;
p5u4 = 126;     p5v4 = 108;
p5u5 = 426;     p5v5 = 124;
p5u6 = 426;     p5v6 = 67;

uvp5 = [25 11;
```

```
        89 55;
        -25 94;
        126 108;
        426 124;
        426 67];

% Picture 6
p6u1 = 19;    p6v1 = 13;
p6u2 = 61;    p6v2 = 60;
p6u3 = -40;   p6v3 = 86;
p6u4 = 91;    p6v4 = 112;
p6u5 = 320;   p6v5 = 144;
p6u6 = 318;   p6v6 = 96;

uvp6 = [19 13;
        61 60;
        -40 86;
        91 112;
        320 144;
        318 96];

% Picture 7
p7u1 = 23;    p7v1 = 21;
p7u2 = 68;    p7v2 = 89;
p7u3 = -55;   p7v3 = 103;
p7u4 = 100;   p7v4 = 152;
p7u5 = 356;   p7v5 = 223;
p7u6 = 354;   p7v6 = 171;

uvp7 = [23 21;
        68 89;
        -55 103;
        100 152;
        356 223;
        354 171];

% Picture 8
p8u1 = 27;    p8v1 = 9;
p8u2 = 100;   p8v2 = 53;
p8u3 = -15;   p8v3 = 101;
p8u4 = 140;   p8v4 = 104;
p8u5 = 476;   p8v5 = 101;
p8u6 = 474;   p8v6 = 38;
```

```
uvp8 = [27 9;
        100 53;
        -15 101;
        140 104;
        476 101;
        474 38];

% Picture 9
p9u1 = 23;    p9v1 = 6;
p9u2 = 96;    p9v2 = 41;
p9u3 = -1;    p9v3 = 94;
p9u4 = 132;   p9v4 = 90;
p9u5 = 466;   p9v5 = 63;
p9u6 = 466;   p9v6 = -5;

uvp9 = [23 6;
        96 41;
        -1 94;
        132 90;
        466 63;
        466 -5];
```

File **DLTparams.m**

```
xyz = [5 2.5 0.5;
       16 10 6.5;
       -4 16 8;
       23 18.5 8.5;
       81 23 7.5;
       83 10 7.5];

uv = [23 6;
       96 41;
       -1 94;
       132 90;
       466 63;
       466 -5];

X = [xyz(1,1) xyz(1,2) xyz(1,3) 1 0 0 0 0 -uv(1,1)*xyz(1,1)
uv(1,1)*xyz(1,2) -uv(1,1)*xyz(1,3);
     0  0  0  0  xyz(1,1)  xyz(1,2)  xyz(1,3)  1  -uv(1,2)*xyz(1,1)  -
uv(1,2)*xyz(1,2) -uv(1,2)*xyz(1,3);
     xyz(2,1)  xyz(2,2)  xyz(2,3)  1  0  0  0  0  -uv(2,1)*xyz(2,1)  -
uv(2,1)*xyz(2,2) -uv(2,1)*xyz(2,3);
     0  0  0  0  xyz(2,1)  xyz(2,2)  xyz(2,3)  1  -uv(2,2)*xyz(2,1)  -
uv(2,2)*xyz(2,2) -uv(2,2)*xyz(2,3);
     xyz(3,1)  xyz(3,2)  xyz(3,3)  1  0  0  0  0  -uv(3,1)*xyz(3,1)  -
uv(3,1)*xyz(3,2) -uv(3,1)*xyz(3,3);
     0  0  0  0  xyz(3,1)  xyz(3,2)  xyz(3,3)  1  -uv(3,2)*xyz(3,1)  -
uv(3,2)*xyz(3,2) -uv(3,2)*xyz(3,3);
     xyz(4,1)  xyz(4,2)  xyz(4,3)  1  0  0  0  0  -uv(4,1)*xyz(4,1)  -
uv(4,1)*xyz(4,2) -uv(4,1)*xyz(4,3);
     0  0  0  0  xyz(4,1)  xyz(4,2)  xyz(4,3)  1  -uv(4,2)*xyz(4,1)  -
uv(4,2)*xyz(4,2) -uv(4,2)*xyz(4,3);
     xyz(5,1)  xyz(5,2)  xyz(5,3)  1  0  0  0  0  -uv(5,1)*xyz(5,1)  -
uv(5,1)*xyz(5,2) -uv(5,1)*xyz(5,3);
     0  0  0  0  xyz(5,1)  xyz(5,2)  xyz(5,3)  1  -uv(5,2)*xyz(5,1)  -
uv(5,2)*xyz(5,2) -uv(5,2)*xyz(5,3);
     xyz(6,1)  xyz(6,2)  xyz(6,3)  1  0  0  0  0  -uv(6,1)*xyz(6,1)  -
uv(6,1)*xyz(6,2) -uv(6,1)*xyz(6,3);
     0  0  0  0  xyz(6,1)  xyz(6,2)  xyz(6,3)  1  -uv(6,2)*xyz(6,1)  -
uv(6,2)*xyz(6,2) -uv(6,2)*xyz(6,3)];
```

```
Y = [uv(1,1); uv(1,2); uv(2,1); uv(2,2); uv(3,1); uv(3,2); uv(4,1);
uv(4,2); uv(5,1); uv(5,2); uv(6,1); uv(6,2)];

L = X\Y;

R1 = L(9)*xyz(1,1) + L(10)*xyz(1,2) + L(11)*xyz(1,3) + 1;
R2 = L(9)*xyz(2,1) + L(10)*xyz(2,2) + L(11)*xyz(2,3) + 1;
R3 = L(9)*xyz(3,1) + L(10)*xyz(3,2) + L(11)*xyz(3,3) + 1;
R4 = L(9)*xyz(4,1) + L(10)*xyz(4,2) + L(11)*xyz(4,3) + 1;
R5 = L(9)*xyz(5,1) + L(10)*xyz(5,2) + L(11)*xyz(5,3) + 1;
R6 = L(9)*xyz(6,1) + L(10)*xyz(6,2) + L(11)*xyz(6,3) + 1;

X1 = [xyz(1,1)/R1 xyz(1,2)/R1 xyz(1,3)/R1 1/R1 0 0 0 0 -
uv(1,1)*xyz(1,1)/R1 -uv(1,1)*xyz(1,2)/R1 -uv(1,1)*xyz(1,3)/R1;
      0 0 0 0 xyz(1,1)/R1 xyz(1,2)/R1 xyz(1,3)/R1 1/R1 -
uv(1,2)*xyz(1,1)/R1 -uv(1,2)*xyz(1,2)/R1 -uv(1,2)*xyz(1,3)/R1;
      xyz(2,1)/R2 xyz(2,2)/R2 xyz(2,3)/R2 1/R2 0 0 0 0 -
uv(2,1)*xyz(2,1)/R2 -uv(2,1)*xyz(2,2)/R2 -uv(2,1)*xyz(2,3)/R2;
      0 0 0 0 xyz(2,1)/R2 xyz(2,2)/R2 xyz(2,3)/R2 1/R2 -
uv(2,2)*xyz(2,1)/R2 -uv(2,2)*xyz(2,2)/R2 -uv(2,2)*xyz(2,3)/R2;
      xyz(3,1)/R3 xyz(3,2)/R3 xyz(3,3)/R3 1/R3 0 0 0 0 -
uv(3,1)*xyz(3,1)/R3 -uv(3,1)*xyz(3,2)/R3 -uv(3,1)*xyz(3,3)/R3;
      0 0 0 0 xyz(3,1)/R3 xyz(3,2)/R3 xyz(3,3)/R3 1/R3 -
uv(3,2)*xyz(3,1)/R3 -uv(3,2)*xyz(3,2)/R3 -uv(3,2)*xyz(3,3)/R3;
      xyz(4,1)/R4 xyz(4,2)/R4 xyz(4,3)/R4 1/R4 0 0 0 0 -
uv(4,1)*xyz(4,1)/R4 -uv(4,1)*xyz(4,2)/R4 -uv(4,1)*xyz(4,3)/R4;
      0 0 0 0 xyz(4,1)/R4 xyz(4,2)/R4 xyz(4,3)/R4 1/R4 -
uv(4,2)*xyz(4,1)/R4 -uv(4,2)*xyz(4,2)/R4 -uv(4,2)*xyz(4,3)/R4;
      xyz(5,1)/R5 xyz(5,2)/R5 xyz(5,3)/R5 1/R5 0 0 0 0 -
uv(5,1)*xyz(5,1)/R5 -uv(5,1)*xyz(5,2)/R5 -uv(5,1)*xyz(5,3)/R5;
      0 0 0 0 xyz(5,1)/R5 xyz(5,2)/R5 xyz(5,3)/R5 1/R5 -
uv(5,2)*xyz(5,1)/R5 -uv(5,2)*xyz(5,2)/R5 -uv(5,2)*xyz(5,3)/R5;
      xyz(6,1)/R6 xyz(6,2)/R6 xyz(6,3)/R6 1/R6 0 0 0 0 -
uv(6,1)*xyz(6,1)/R6 -uv(6,1)*xyz(6,2)/R6 -uv(6,1)*xyz(6,3)/R6;
      0 0 0 0 xyz(6,1)/R6 xyz(6,2)/R6 xyz(6,3)/R6 1/R6 -
uv(6,2)*xyz(6,1)/R6 -uv(6,2)*xyz(6,2)/R6 -uv(6,2)*xyz(6,3)/R6];

Y1 = [uv(1,1)/R1; uv(1,2)/R1; uv(2,1)/R2; uv(2,2)/R2; uv(3,1)/R3;
uv(3,2)/R3;
      uv(4,1)/R4; uv(4,2)/R4, uv(5,1)/R5; uv(5,2)/R5; uv(6,1)/R6;
uv(6,2)/R6];

L1 = X1\Y1
```

File **DLTrecon.m**

```
% The DLT coefficients obtained for all pictures

L1 = [3.9126; -2.6272; 3.3129; 10.5323; 0.1463; 2.7558; 1.2216; 8.1244;
0.0006; -0.0064; -0.0382];
L2 = [4.2610; -1.8649; 1.3580; 10.0434; 0.2668; 3.2924; 0.9025; 7.4917;
0.0016; -0.0051; -0.0368];
L3 = [2.9723; -2.9987; 5.1921; 15.9410; -0.0877; 2.3621; 0.8193; 5.5674;
-0.0009; -0.0067; -0.0391];
L4 = [3.1391; -2.7287; 4.5985; 11.9132; -0.2585; 2.4161; 1.6834; 3.9167;
-0.0008; -0.0069; -0.0344];
L5 = [3.3421; -2.3964; 3.1306; 11.8762; 0.0550; 2.1595; 1.9852; 3.8615;
0.0003; -0.0068; -0.0375];
L6 = [3.0609; -1.1705; 0.0908; 6.1299; 0.4329; 2.1909; 1.7181; 4.0775;
0.0015; -0.0053; -0.0382];
L7 = [4.0667; -1.2271; -0.7813; 5.7743; 1.2229; 2.6470; 2.9151; 6.4274;
0.0024; -0.0052; -0.0300];
L8 = [3.7488; -2.5685; 4.3056; 11.6375; -0.2488; 2.6610; 2.3872; 2.0590;
-0.0002; -0.0069; -0.0292];
L9 = [2.8530; -3.0318; 5.8348; 12.4139; -0.5047; 2.4187; 1.7806; 1.2247;
-0.0013; -0.0076; -0.0339];

%[u,v] of the 3rd control point in the different image views.

p1u3 = -35;   p1v3 = 107;
p2u3 = -42;   p2v3 = 110;
p3u3 = -4;    p3v3 = 89;
p4u3 = -12;   p4v3 = 94;
p5u3 = -25;   p5v3 = 94;
p6u3 = -40;   p6v3 = 86;
p7u3 = -55;   p7v3 = 103;
p8u3 = -15;   p8v3 = 101;
p9u3 = -1;    p9v3 = 94;


A1 = [p1u3*L1(9)-L1(1) p1u3*L1(10)-L1(2) p1u3*L1(11)-L1(3);
      p1v3*L1(9)-L1(5) p1v3*L1(10)-L1(6) p1v3*L1(11)-L1(7);
      p2u3*L2(9)-L2(1) p2u3*L2(10)-L2(2) p2u3*L2(11)-L2(3);
      p2v3*L2(9)-L2(5) p2v3*L2(10)-L2(6) p2v3*L2(11)-L2(7);
```

```
       p3u3*L3(9)-L3(1) p3u3*L3(10)-L3(2) p3u3*L3(11)-L3(3);
       p3v3*L3(9)-L3(5) p3v3*L3(10)-L3(6) p3v3*L3(11)-L3(7);
       p4u3*L4(9)-L4(1) p4u3*L4(10)-L4(2) p4u3*L4(11)-L4(3);
       p4v3*L4(9)-L4(5) p4v3*L4(10)-L4(6) p4v3*L4(11)-L4(7);
       p5u3*L5(9)-L5(1) p5u3*L5(10)-L5(2) p5u3*L5(11)-L5(3);
       p5v3*L5(9)-L5(5) p5v3*L5(10)-L5(6) p5v3*L5(11)-L5(7);
       p6u3*L6(9)-L6(1) p6u3*L6(10)-L6(2) p6u3*L6(11)-L6(3);
       p6v3*L6(9)-L6(5) p6v3*L6(10)-L6(6) p6v3*L6(11)-L6(7);
       p7u3*L7(9)-L7(1) p7u3*L7(10)-L7(2) p7u3*L7(11)-L7(3);
       p7v3*L7(9)-L7(5) p7v3*L7(10)-L7(6) p7v3*L7(11)-L7(7);
       p8u3*L8(9)-L8(1) p8u3*L8(10)-L8(2) p8u3*L8(11)-L8(3);
       p8v3*L8(9)-L8(5) p8v3*L8(10)-L8(6) p8v3*L8(11)-L8(7);
       p9u3*L9(9)-L9(1) p9u3*L9(10)-L9(2) p9u3*L9(11)-L9(3);
       p9v3*L9(9)-L9(5) p9v3*L9(10)-L9(6) p9v3*L9(11)-L9(7)];

B1 = [L1(4)-p1u3; L1(8)-p1v3;
      L2(4)-p2u3; L2(8)-p2v3;
      L3(4)-p3u3; L3(8)-p3v3;
      L4(4)-p4u3; L4(8)-p4v3;
      L5(4)-p5u3; L5(8)-p5v3;
      L6(4)-p6u3; L6(8)-p6v3;
      L7(4)-p7u3; L7(8)-p7v3;
      L8(4)-p8u3; L8(8)-p8v3;
      L8(4)-p9u3; L9(8)-p9v3];

reconXYZ1 = A1\B1;

R1 = L1(9)*reconXYZ1(1)+L1(10)*reconXYZ1(2)+L1(11)*reconXYZ1(3)+1;
R2 = L2(9)*reconXYZ1(1)+L2(10)*reconXYZ1(2)+L2(11)*reconXYZ1(3)+1;
R3 = L3(9)*reconXYZ1(1)+L3(10)*reconXYZ1(2)+L3(11)*reconXYZ1(3)+1;
R4 = L4(9)*reconXYZ1(1)+L4(10)*reconXYZ1(2)+L4(11)*reconXYZ1(3)+1;
R5 = L5(9)*reconXYZ1(1)+L5(10)*reconXYZ1(2)+L5(11)*reconXYZ1(3)+1;
R6 = L6(9)*reconXYZ1(1)+L6(10)*reconXYZ1(2)+L6(11)*reconXYZ1(3)+1;
R7 = L7(9)*reconXYZ1(1)+L7(10)*reconXYZ1(2)+L7(11)*reconXYZ1(3)+1;
R8 = L8(9)*reconXYZ1(1)+L8(10)*reconXYZ1(2)+L8(11)*reconXYZ1(3)+1;
R9 = L9(9)*reconXYZ1(1)+L9(10)*reconXYZ1(2)+L9(11)*reconXYZ1(3)+1;

A = [(p1u3*L1(9)-L1(1))/R1 (p1u3*L1(10)-L1(2))/R1 (p1u3*L1(11)-L1(3))/R1;
     (p1v3*L1(9)-L1(5))/R1 (p1v3*L1(10)-L1(6))/R1 (p1v3*L1(11)-L1(7))/R1;
     (p2u3*L2(9)-L2(1))/R2 (p2u3*L2(10)-L2(2))/R2 (p2u3*L2(11)-L2(3))/R2;
     (p2v3*L2(9)-L2(5))/R2 (p2v3*L2(10)-L2(6))/R2 (p2v3*L2(11)-L2(7))/R2;
     (p3u3*L3(9)-L3(1))/R3 (p3u3*L3(10)-L3(2))/R3 (p3u3*L3(11)-L3(3))/R3;
```

```
       (p3v3*L3(9)-L3(5))/R3 (p3v3*L3(10)-L3(6))/R3 (p3v3*L3(11)-L3(7))/R3;
       (p4u3*L4(9)-L4(1))/R4 (p4u3*L4(10)-L4(2))/R4 (p4u3*L4(11)-L4(3))/R4;
       (p4v3*L4(9)-L4(5))/R4 (p4v3*L4(10)-L4(6))/R4 (p4v3*L4(11)-L4(7))/R4;
       (p5u3*L5(9)-L5(1))/R5 (p5u3*L5(10)-L5(2))/R5 (p5u3*L5(11)-L5(3))/R5;
       (p5v3*L5(9)-L5(5))/R5 (p5v3*L5(10)-L5(6))/R5 (p5v3*L5(11)-L5(7))/R5;
       (p6u3*L6(9)-L6(1))/R6 (p6u3*L6(10)-L6(2))/R6 (p6u3*L6(11)-L6(3))/R6;
       (p6v3*L6(9)-L6(5))/R6 (p6v3*L6(10)-L6(6))/R6 (p6v3*L6(11)-L6(7))/R6;
       (p7u3*L7(9)-L7(1))/R7 (p7u3*L7(10)-L7(2))/R7 (p7u3*L7(11)-L7(3))/R7;
       (p7v3*L7(9)-L7(5))/R7 (p7v3*L7(10)-L7(6))/R7 (p7v3*L7(11)-L7(7))/R7;
       (p8u3*L8(9)-L8(1))/R8 (p8u3*L8(10)-L8(2))/R8 (p8u3*L8(11)-L8(3))/R8;
       (p8v3*L8(9)-L8(5))/R8 (p8v3*L8(10)-L8(6))/R8 (p8v3*L8(11)-L8(7))/R8;
       (p9u3*L9(9)-L9(1))/R9 (p9u3*L9(10)-L9(2))/R9 (p9u3*L9(11)-L9(3))/R9;
       (p9v3*L9(9)-L9(5))/R9 (p9v3*L9(10)-L9(6))/R9 (p9v3*L9(11)-L9(7))/R9];


B = [(L1(4)-p1u3)/R1; (L1(8)-p1v3)/R1;
     (L2(4)-p2u3)/R2; (L2(8)-p2v3)/R2;
     (L3(4)-p3u3)/R3; (L3(8)-p3v3)/R3;
     (L4(4)-p4u3)/R4; (L4(8)-p4v3)/R4;
     (L5(4)-p5u3)/R5; (L5(8)-p5v3)/R5;
     (L6(4)-p6u3)/R6; (L6(8)-p6v3)/R6;
     (L7(4)-p7u3)/R7; (L7(8)-p7v3)/R7;
     (L8(4)-p8u3)/R8; (L8(8)-p8v3)/R8;
     (L9(4)-p9u3)/R9, (L9(8)-p9v3)/R9];

reconXYZ = A\B
```

The following is the Matlab code for color marker detection

File **ThreshRed.m**

```
%Routine to identify the Red marker
%--------------------------------
I = imread('cp1.jpg');
RI = I(:,:,1);
figure;
imshow(I);
%figure;
%imshow(RI);
[m,n] = size(RI);
thresRI = zeros(m,n);
k = 0;
x = 0;
y = 0;
for i = 1:m
    for j = 1:n
        if  ((I(i,j,2) <= 150) & (I(i,j,3) <= 150) & (RI(i,j) >= 250))
            thresRI(i,j) = 256;
        else
            thresRI(i,j) = 0;
        end
        if (thresRI(i,j) == 256)
            y = y + i;
            x = x + j;
            k = k + 1;
        end
    end
end
figure;
imshow(thresRI);
redavgx = x / k;
redavgy = y / k;
redavgx
redavgy
```

File **ThreshGreen.m**

```
%Routine to identify the Green marker
%---------------------------------
I = imread('cp1.jpg');
%RI = I(:,:,3);
figure;
imshow(I);
%figure,
%imshow(RI);
[m,n] = size(RI);
thresRI = zeros(m,n);
k = 0;
x = 0;
y = 0;
for i = 1:m
    for j = 1:n
        if ((I(i,j,1) <= 200) & (I(i,j,3) <= 200) & (I(i,j,2) >= 230))
            thresRI(i,j) = 256;
        else
            thresRI(i,j) = 0;
        end
        if (thresRI(i,j) == 256)
            y = y + i;
            x = x + j;
            k = k + 1;
        end
    end
end
figure;
imshow(thresRI);
greenavgx = x / k;
greenavgy = y / k;
greenavgx
greenavgy
```

File **ThreshBlue.m**

```
%Routine to identify the Blue marker
%---------------------------------
I = imread('cp1.jpg');
RI = I(:,:,3);
figure;
imshow(I);
%figure;
%imshow(RI);
[m,n] = size(RI);
thresRI = zeros(m,n);
k = 0;
x = 0;
y = 0;
for i = 1:m
    for j = 1:n
        if  ((I(i,j,2) <= 230) & (I(i,j,1) <= 200) & (I(i,j,3) >= 250))
            thresRI(i,j) = 256;
        else
            thresRI(i,j) = 0;
        end
        if (thresRI(i,j) == 256)
            y = y + i;
            x = x + j;
            k = k + 1;
        end
    end
end
figure;
imshow(thresRI);
blueavgx = x / k;
blueavgy = y / k;
blueavgx
blueavgy
```

The following is the Matlab code for corner detection

File **CornerDetect.m**

```
I = imread('DLTcp1_BW.jpg');
figure;
imshow(I);
im = I;
sigma = 0.355;
thresh = 3000;
radius = 1;
dx = [-1 0 1; -1 0 1; -1 0 1]/3;
dy = dx';

Ix = conv2(im, dx, 'same');
Iy = conv2(im, dy, 'same');

g = fspecial('gaussian',max(1,fix(6*sigma)), sigma);
%g = fspecial('gaussian',2, sigma);

Ix2 = conv2(Ix.^2, g, 'same');
Iy2 = conv2(Iy.^2, g, 'same');
Ixy = conv2(Ix.*Iy, g, 'same');


cim = (Ix .* Iy - Ixy.^2) - 0.04 * ((Ix + Iy).^2);

sze = 2*radius+1;
mx = ordfilt2(cim,sze^2,ones(sze));
cim = (cim==mx)&(cim>thresh);

[r,c] = find(cim);

figure, imagesc(im), axis image, colormap(gray), hold on
plot(c,r,'r+'), title('corners detected');
```

The following is the Matlab code for image correlation

File **DetectbyCorr.m**

```
bw = imread('DLTcp1_BW.jpg');
%figure, imshow(bw);
a = imread('ChecksMark.jpg');
%figure, imshow(a);
C = xcorr2(double(bw),double(a));
%C = real(ifft2(fft2(bw) .* fft2(rot90(a,2),480,640)));
figure, imshow(C,[])                %Display by scaling.
% max(C(:))                         %Find max pixel value in C.
figure, imshow(C > 10000000);       %Display showing pixels over threshold.
Cthresh = (C > 10000000);
[r,c] = find(Cthresh);
figure;
imshow(bw);
colormap(gray);
hold on;
plot(c,r,'r+');
title('Markers Detected');
```