ASSIGNMENT A3

Shantnu Kakkar

CS 6320, Spring 2016

February 10, 2016

Section 1: Intro:

This assignment is based on the Camera Calibration technique learnt in the lecture. Basically, camera calibration is an optimization process, where the discrepancy between the observed image feature and their theoretical positions is minimized with respect to the camera's intrinsic and extrinsic parameters. In this assignment, I address the problem of estimating the intrinsic and extrinsic parameters of a camera from image coordinates of a scene (6 points) whose positions are known in world frame. The following figure shows the setup:

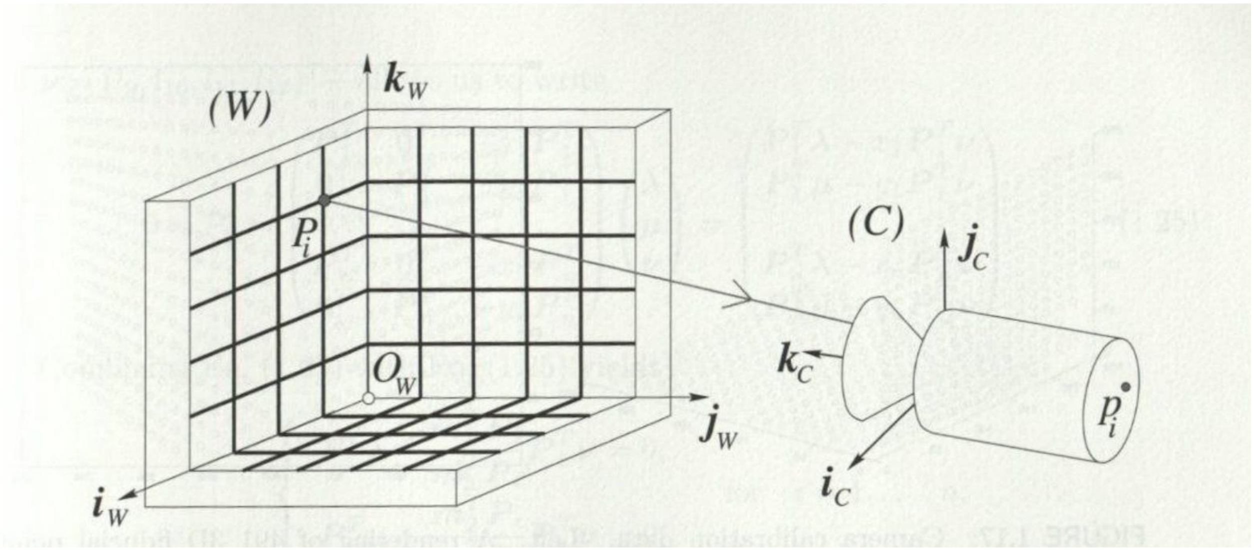


Figure 1 Camera Calibration setup. In this example, the calibration rig is formed by three grids drawn orthogonal planes

A linear approach is applied to estimate the intrinsic and extrinsic parameters of the camera. The approach is divided into two steps:

- Computation of perspective projection matrix M associated with the camera
- Using M to find intrinsic and extrinsic parameters

It is assumed that the Camera has a non-zero skew. According to Theorem 1 in text, matrix M is not singular, but otherwise arbitrary. We get the following equtions to solve for ith point in the scene.

$$(m_1 - x_i m_3).P_i = P_i^T m_1 + 0^T m_2 - x_i P_i^T m_3 = 0$$

$$(m_2 - y_i m_3).P_i = P_i^T m_2 + 0^T m_1 - y_i P_i^T m_3 = 0$$

Where,

- m_1, m_2, m_3 are the three rows of the M matrix
- P_i is the corrdinates of the ith point in world frame
- X_i and y_i are the coordinates of image of P_i

If we make a single matrix out of this we get:

$$Pm = 0$$

Where,

$$\mathcal{P} \overset{\text{def}}{=} \begin{pmatrix} \boldsymbol{P}_1^T & \boldsymbol{0}^T & -x_1\boldsymbol{P}_1^T \\ \boldsymbol{0}^T & \boldsymbol{P}_1^T & -y_1\boldsymbol{P}_1^T \\ \dots & \dots & \dots \\ \boldsymbol{P}_n^T & \boldsymbol{0}^T & -x_n\boldsymbol{P}_n^T \\ \boldsymbol{0}^T & \boldsymbol{P}_n^T & -y_n\boldsymbol{P}_n^T \end{pmatrix} \text{ and } \boldsymbol{m} \overset{\text{def}}{=} \begin{pmatrix} \boldsymbol{m}_1 \\ \boldsymbol{m}_2 \\ \boldsymbol{m}_3 \end{pmatrix} = 0.$$

When n>=6; the homogeneous linear least square can be used to compute the value of the unit vector m (and hence the matrix M) that minimizes $||Pm||^2$ as the eigenvector of the 12X12 matrix P^TP associated with its smallest eigen value.

<u>A note about degenerate cases:</u> The fiducial points P_i should not lie in the same plane

The following figures shows various formulas used in calculating intrinsic and extrinsic parameters:

$$\begin{aligned} \rho &= \varepsilon/||\boldsymbol{a}_3||, \\ \boldsymbol{r}_3 &= \rho\boldsymbol{a}_3, \\ x_0 &= \rho^2(\boldsymbol{a}_1 \cdot \boldsymbol{a}_3) \\ y_0 &= \rho^2(\boldsymbol{a}_2 \cdot \boldsymbol{a}_3), \end{aligned}$$

$$\begin{aligned} \cos\theta &= -\frac{(\boldsymbol{a}_1 \times \boldsymbol{a}_3) \cdot (\boldsymbol{a}_2 \times \boldsymbol{a}_3)}{||\boldsymbol{a}_1 \times \boldsymbol{a}_3||\,||\boldsymbol{a}_2 \times \boldsymbol{a}_3||} \\ \alpha &= \rho^2||\boldsymbol{a}_1 \times \boldsymbol{a}_3||\sin\theta, \\ \beta &= \rho^2||\boldsymbol{a}_2 \times \boldsymbol{a}_3||\sin\theta, \end{aligned}$$

$$\begin{aligned} \boldsymbol{r}_1 &= \frac{\rho^2\sin\theta}{\beta}(\boldsymbol{a}_2 \times \boldsymbol{a}_3) = \frac{1}{||\boldsymbol{a}_2 \times \boldsymbol{a}_3||}(\boldsymbol{a}_2 \times \boldsymbol{a}_3), \\ \boldsymbol{r}_2 &= \boldsymbol{r}_3 \times \boldsymbol{r}_1. \end{aligned}$$

$$\boldsymbol{t} = \rho\mathcal{K}^{-1}\boldsymbol{b}.$$

The following question is going to be answered in this report:

- How sensitive is the camera calibration to the noise in image locations?
- Develop and describe a method to extract the points from the given image in A3

Section 2: Method:

Matlab is used to carry out the experiments.

I do analysis by observing the following steps:

- Plotting variance in image coordinates v/s Mean error in various estimated parameters (rho, N f)
- Computing Mean, Variances and confidence interval for the above plot

Following functions are implemented

- CS5320_calibrate: It determines camera parameters. Its inputs are image points (homogeneous coords) and world coordinates (homogeneous coords), and outputs are alpha, beta, theta, x0,y0 , R and t
- CS5320_errors: It finds statistics on intrinsic and extrinsic parameters. Inputs are image points (homogeneous coords), world coordinates (homogeneous cords and number of trials to run. Output is result variable which is as defined as follows:
  results (5x4 array): means, variances, and confidence intervals
  - row 1: alpha
  - row 2: beta
  - row 3: theta
  - row 4: x0
  - row 5: y0
  - col 1: mean
  - col 2: variance
  - col 3: lower value of confidence interval
  - col 4: upper value of confidence interval
- CS5320_part2: It Returns the image points by clicking them in the image, and corresponding world points. Just click six times in the image in the sequence given in A3 and you would get back the coordinates. Outputs are are image points (homogeneous coords) and world coordinates (homogeneous coords)

The following algorithms are used for calibration:

1. Make P matrix from given world and image coordinates.
2. Find eigen vectors of P'P using [V,D] = eigs(PMatrix'*PMatrix , 12);
3. Sort the eigen values using [vOld,indexes] = sort(diag(D));
4. Fnd the smallest eigen vector using index = indexes(1);
5. Contrsuct your M using M=[V(1:4,index)';V(5:8,index)';V(9:12,index)'];
6. Normalize M using M_hat=M/norm(M(3,1:3));
7. Check if any of z is positive. If any z is positive then M_hat is inverted
   ```
   imp = M_hat*P;
   if max(imp(3,:)) > 0
      M_hat = -M_hat;
      imp = M_hat*P;
   end
   ```

```
8.  Form  A and b which will be used
    A = M_hat(:,1:3);
    b = M_hat(:,4);
9.  Extracting information from A and b
    a1Transpose = A(1,1:3);
    a1 = a1Transpose';
    a2Transpose = A(2,1:3);
    a2 = a2Transpose';
    a3Transpose = A(3,1:3);
    a3 = a3Transpose';
10. Substituting various formulas as follows
    rho = 1/norm(a3);
    r3 = rho*a3;
    r1 = cross(a2,a3)/norm(cross(a2,a3));
    r2 = cross(r3,r1);
    x0 = rho*rho*dot(a1,a3);
    y0 = rho*rho*dot(a2,a3);
    theta = acos((-dot(cross(a1,a3),cross(a2,a3)))...
       /(norm(cross(a1,a3))*norm(cross(a2,a3))));
    alpha = rho*rho*norm(cross(a1,a3))*sin(theta);
    beta =   rho*rho*norm(cross(a2,a3))*sin(theta);
    K = [alpha,-alpha*cot(theta),x0; 0,beta/sin(theta),y0; 0,0,1];
    t = rho*inv(K)*b;
    R = [r1'; r2'; r3'];
11. Finding Tranformation using calculated R and t, invert that transformation, and finally extract R and t from
    the inverse transformation
    TCameratoWorld = [R(1,:)   ,   t(1) ;...
                      R(2,:)   ,   t(2) ;...
                      R(3,:)   ,   t(3) ;...
                      0,0,0    ,     1 ];

    Ti = inv(TCameratoWorld);
    R = Ti(1:3,1:3);
    t = Ti(1:3,4);
```

The following algorithms are used for study of sensitivity in calibration:

```
1.  Calibrate camera withut noise in image coordinates.
    [alpha,beta,theta,x0,y0,R,t] = CS5320_calibrate(pts_im , pts_world);
2.  Set vs = [0.1:0.1:1];
3.  Set num_vs = length(vs) ;
4.  Loop v_index = 1:num_vs  % set variance
    4.1.    Set v = vs(v_index);
    4.2.    Set Error_alpha = [];Error_beta = [];Error_theta = []; Error_x0 = []; Error_y0 =[];  Error_R =
            [];Error_t = [];
    4.3.    Loop for i=1:NumOfTrials
    4.4.     pts_im_Noisy(1,:) = pts_im(1,:) + sqrt(v) * randn(1 , size(pts_im, 2));
    4.5.     pts_im_Noisy(2,:) = pts_im(2,:) + sqrt(v) * randn(1 , size(pts_im, 2));
    4.6.     pts_im_Noisy(3,:) = pts_im(3,:);
    4.7.     [alphaN,betaN,thetaN,x0N,y0N,RN,tN]...
           = CS5320_calibrate(pts_im_Noisy , pts_world);
    4.8.     Error_alpha = [Error_alpha ; alpha-alphaN];
```

```
4.9.    Error_beta = [Error_beta ; beta-betaN];
        Error_theta = [Error_theta ; theta-thetaN];
        Error_x0 = [Error_x0 ; x0-x0N];
        Error_y0 = [Error_y0 ; y0-y0N];
        %Error_R(v_index,i)  = mean(mean(abs(R-RN)));
        %Error_t(v_index,i)  = norm(t-tN);
        End the inner loop
5.    MeanError(v_index,1) = mean(Error_alpha);
      MeanError(v_index,2) = mean(Error_beta);
      MeanError(v_index,3) = mean(Error_theta);
      MeanError(v_index,4) = mean(Error_x0);
      MeanError(v_index,5) = mean(Error_y0);
      MeanError_R = mean(Error_R(v_index,:));
      MMeanErorr_t = mean(Error_t(v_index,:));
6.
      VarError_alpha(v_index) = var(Error_alpha);
      VarError_beta(v_index) = var(Error_beta);
      VarError_theta(v_index) = var(Error_theta);
      VarError_x0(v_index) = var(Error_x0);
      VarError_y0(v_index) = var(Error_y0);
7.  End the outer loop

8.  Finding mean variance, CI low and CI high of above mean vector
    for r = 1:5
       results(r,1) = mean(MeanError(:,r));
       results(r,2) = var(MeanError(:,r));
       results(r,3) = results(r,1) - 1.66*sqrt(results(r,2)/NumOfTrials);
       results(r,4) = results(r,1) + 1.66*sqrt(results(r,2)/NumOfTrials);
9.  end
```

The following method is used for finding points in the image:

1. pts_im_part2 = [];
2. Read image image = imread('cal_im.jpg');
3. Change to gray scale newimage = rgb2gray(image);
4. Show with certain intensities imshow(newimage<40)
5. Take input [imX,imY]=ginput(6);
6. Set x values pts_im_part2(1,1:6) = imX;
7. Set y values pts_im_part2(2,1:6) = imY;
8. Make homogeneous pts_im_part2(3,1:6) = ones(1,6);

Metric for R:

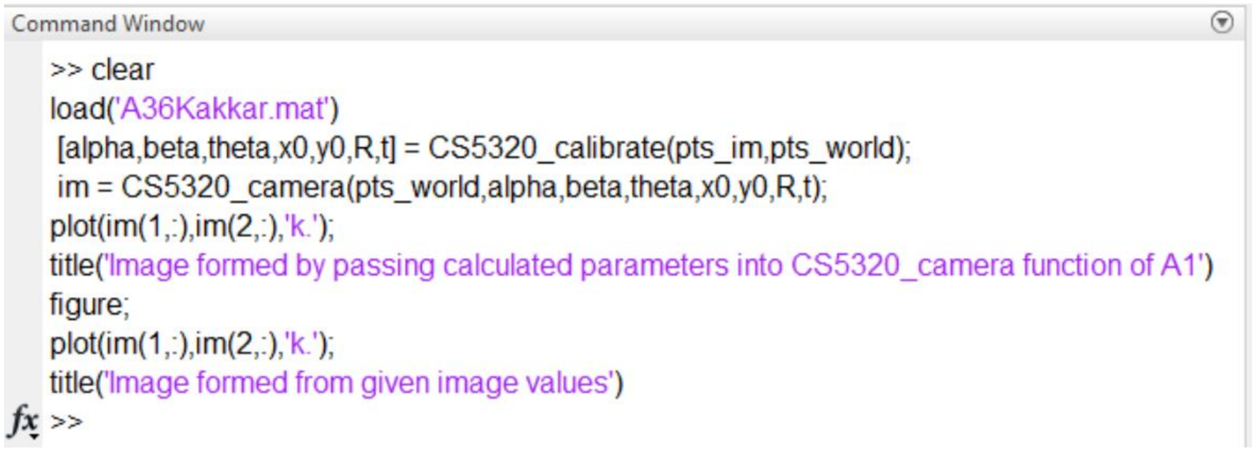
Using Berkeley website, I got the following:

- Assume that we are trying to estimate a matrix A, and came up with an estimate $\hat{A}$. How can we measure the quality of our estimate? One way is to evaluate by how much they differ when they act on the standard basis. This leads to the Frobenius norm.

Section 3: Verification:

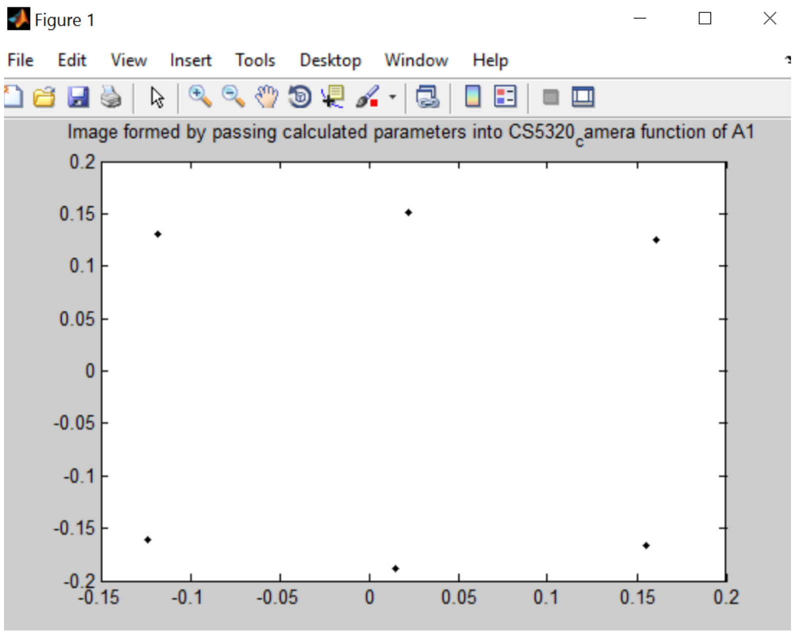
Testing CS5320_calibrate

- In the following screen shot, I used the calculated parameters to find the image using the CS5320_camera function from A1. I am getting the exact image.



```
>> clear
load('A36Kakkar.mat')
 [alpha,beta,theta,x0,y0,R,t] = CS5320_calibrate(pts_im,pts_world);
 im = CS5320_camera(pts_world,alpha,beta,theta,x0,y0,R,t);
plot(im(1,:),im(2,:),'k.');
title('Image formed by passing calculated parameters into CS5320_camera function of A1')
figure;
plot(im(1,:),im(2,:),'k.');
title('Image formed from given image values')
>>
```

I am getting my image coordinates back as seen from following two images.



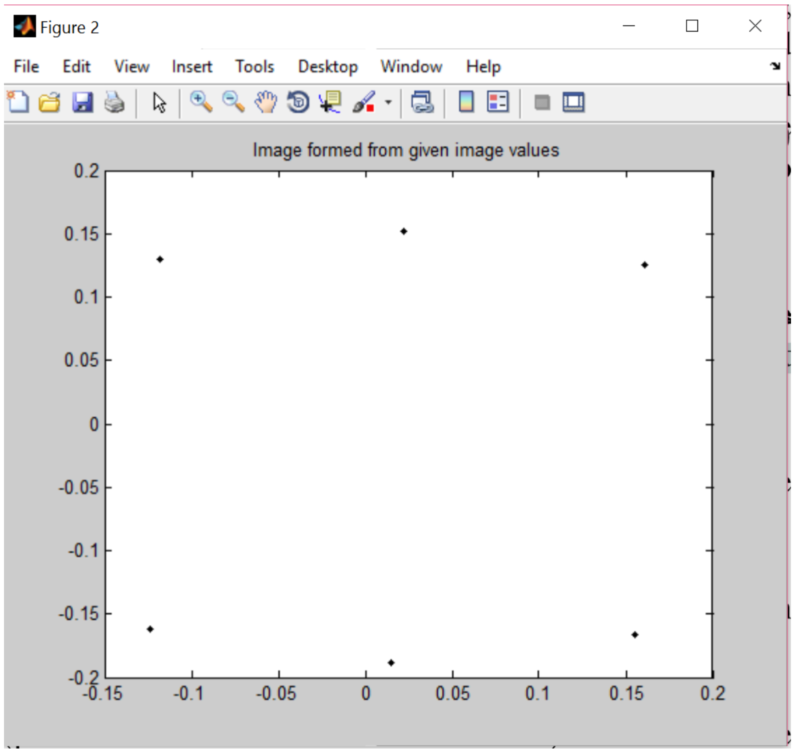
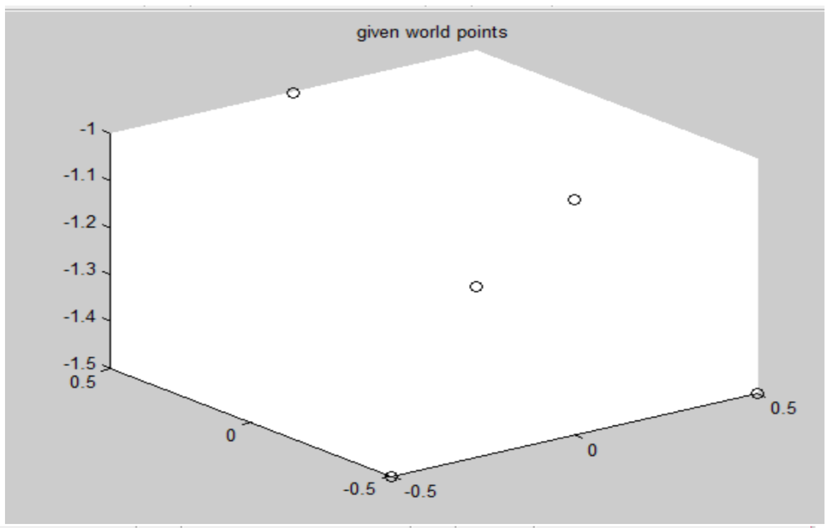
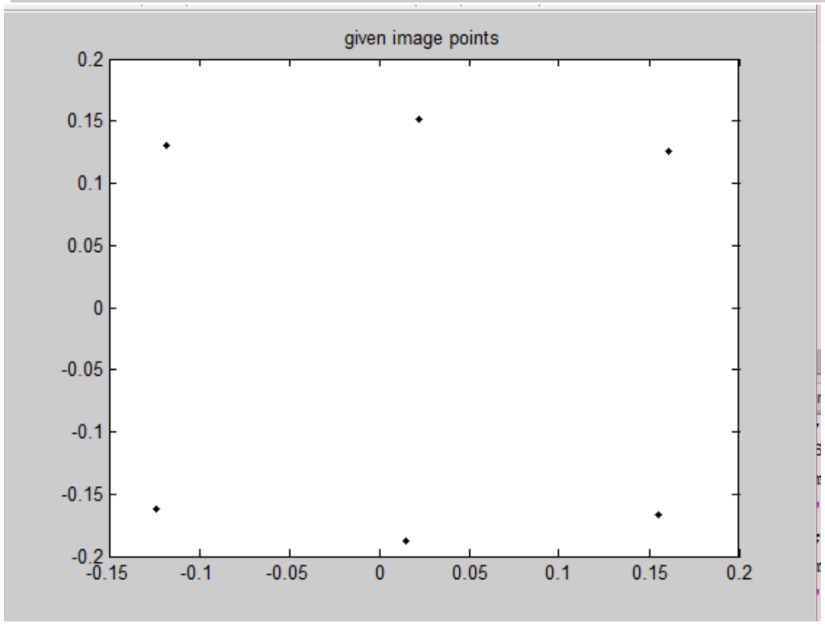


Figure 2
File
Edit
View
Insert
Tools
Desktop
Window
Help
Image formed from given image values
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.15
-0.1
-0.05
0
0.05
0.1
0.15
0.2

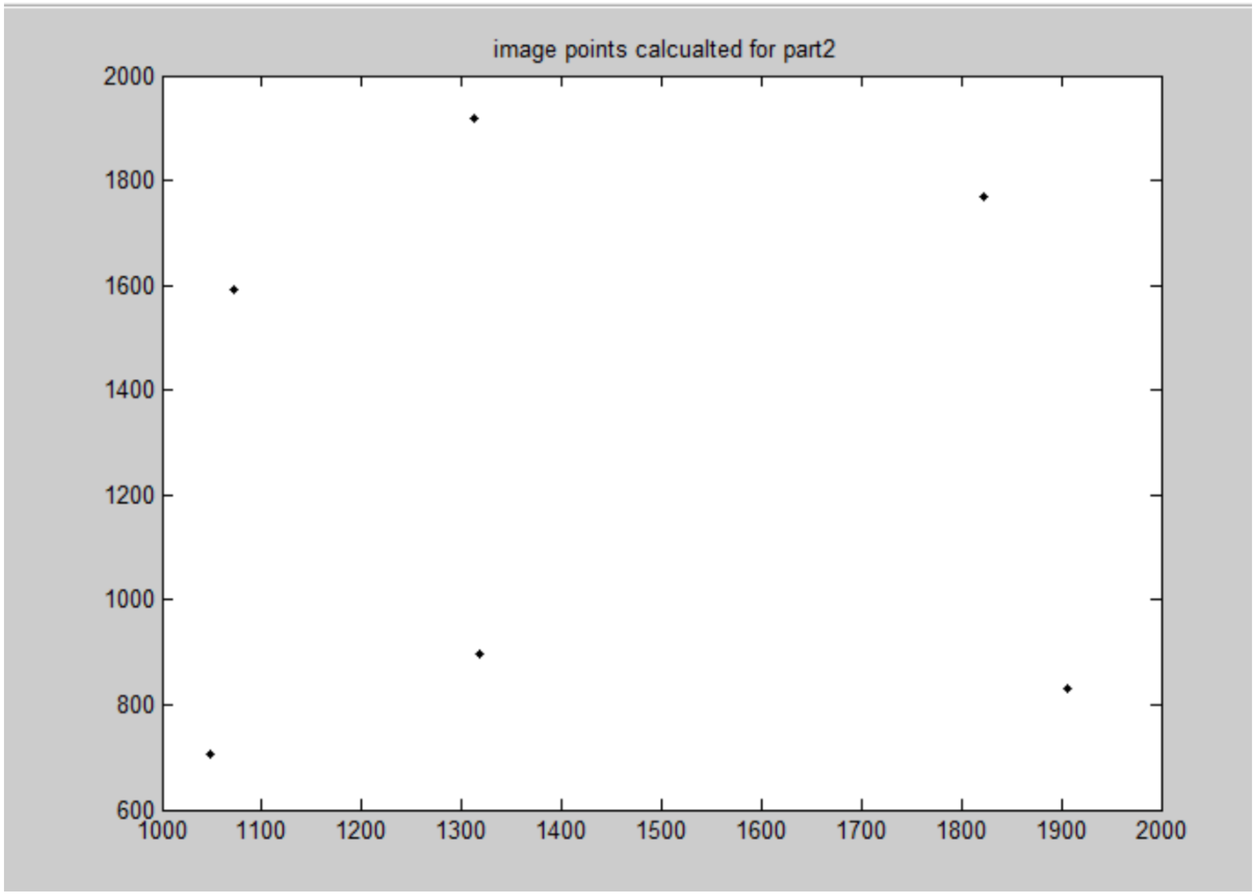
Section 4: Data:

Following figure show the image and world coordinate given in mat file, that go into the CS5320_calibrate function

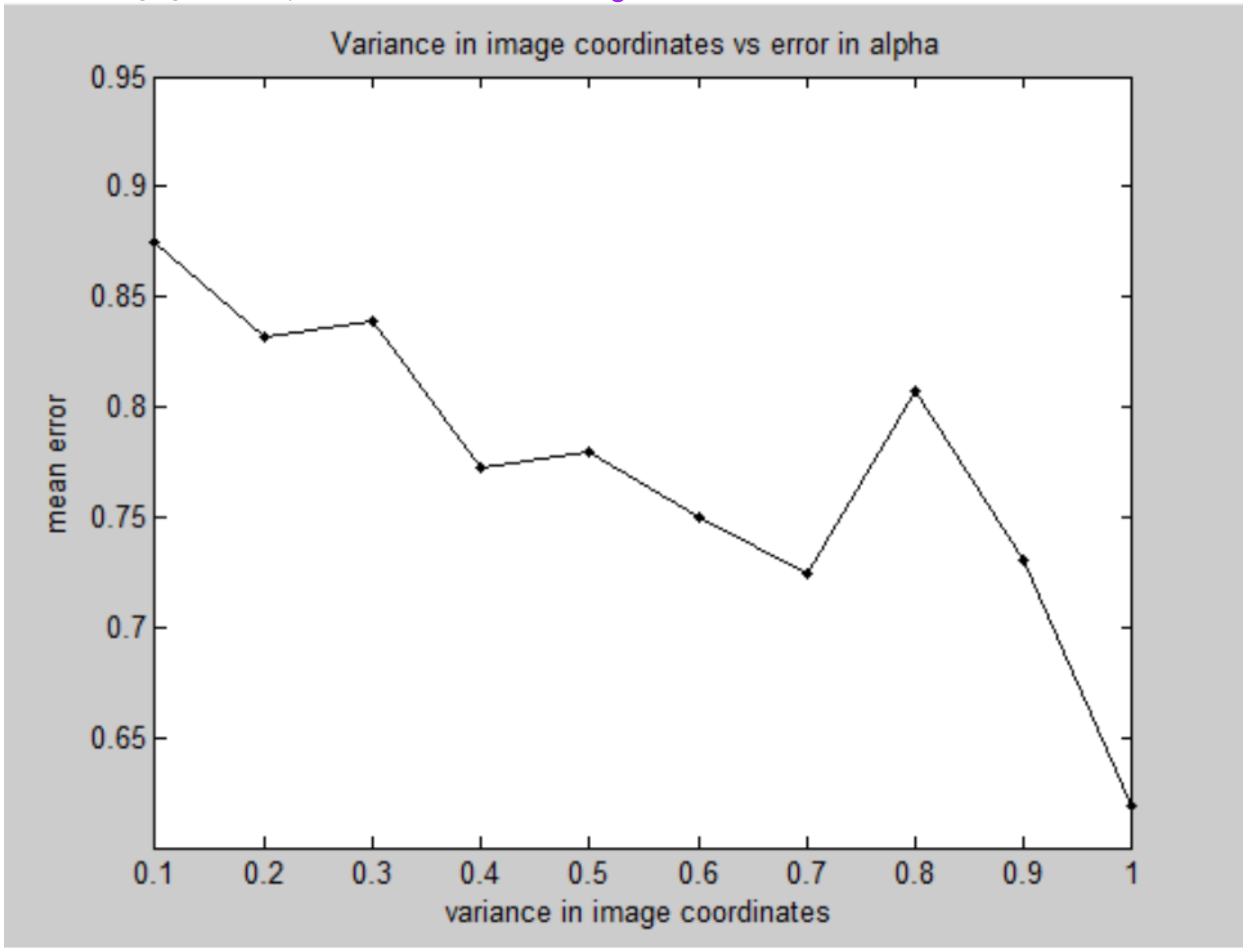


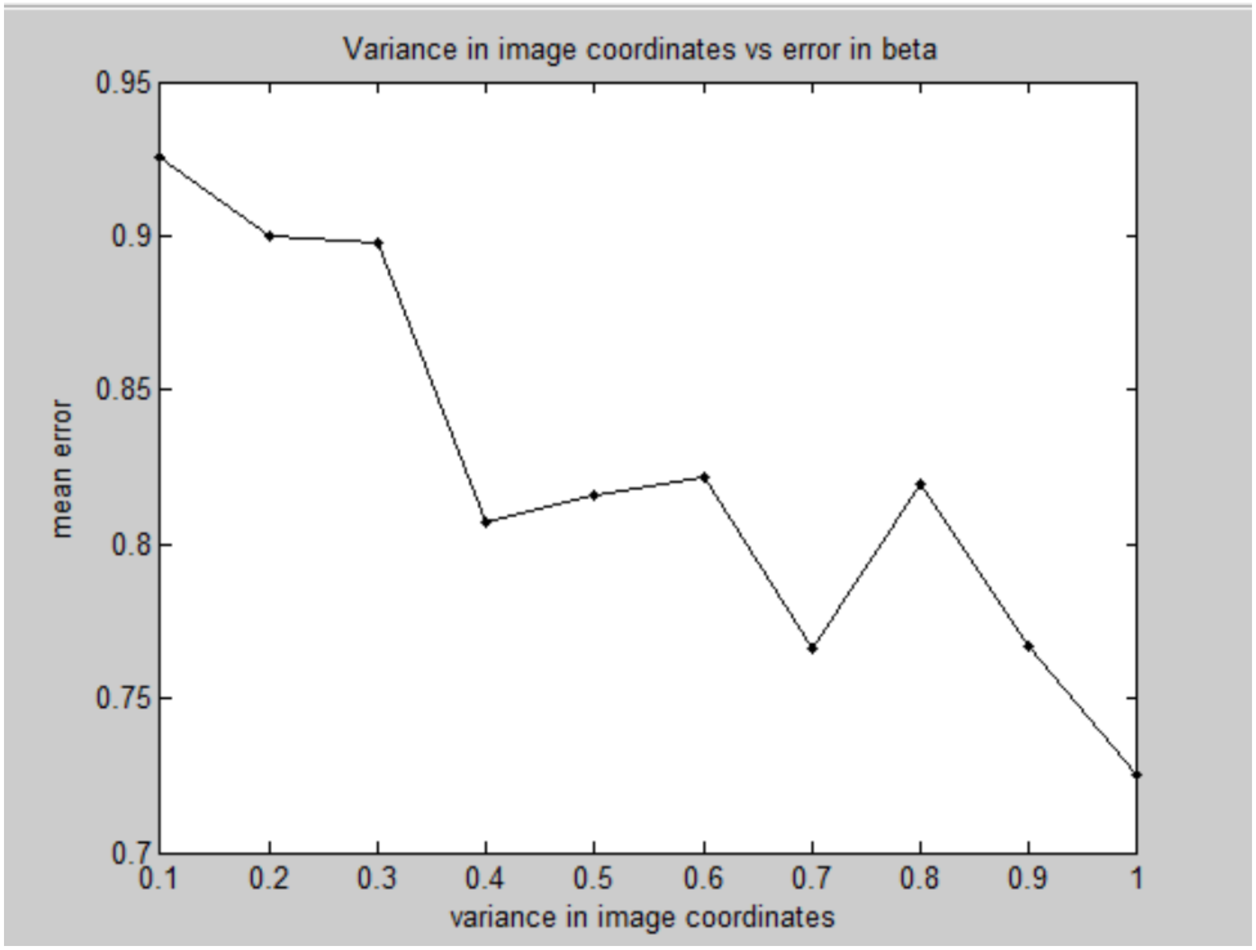


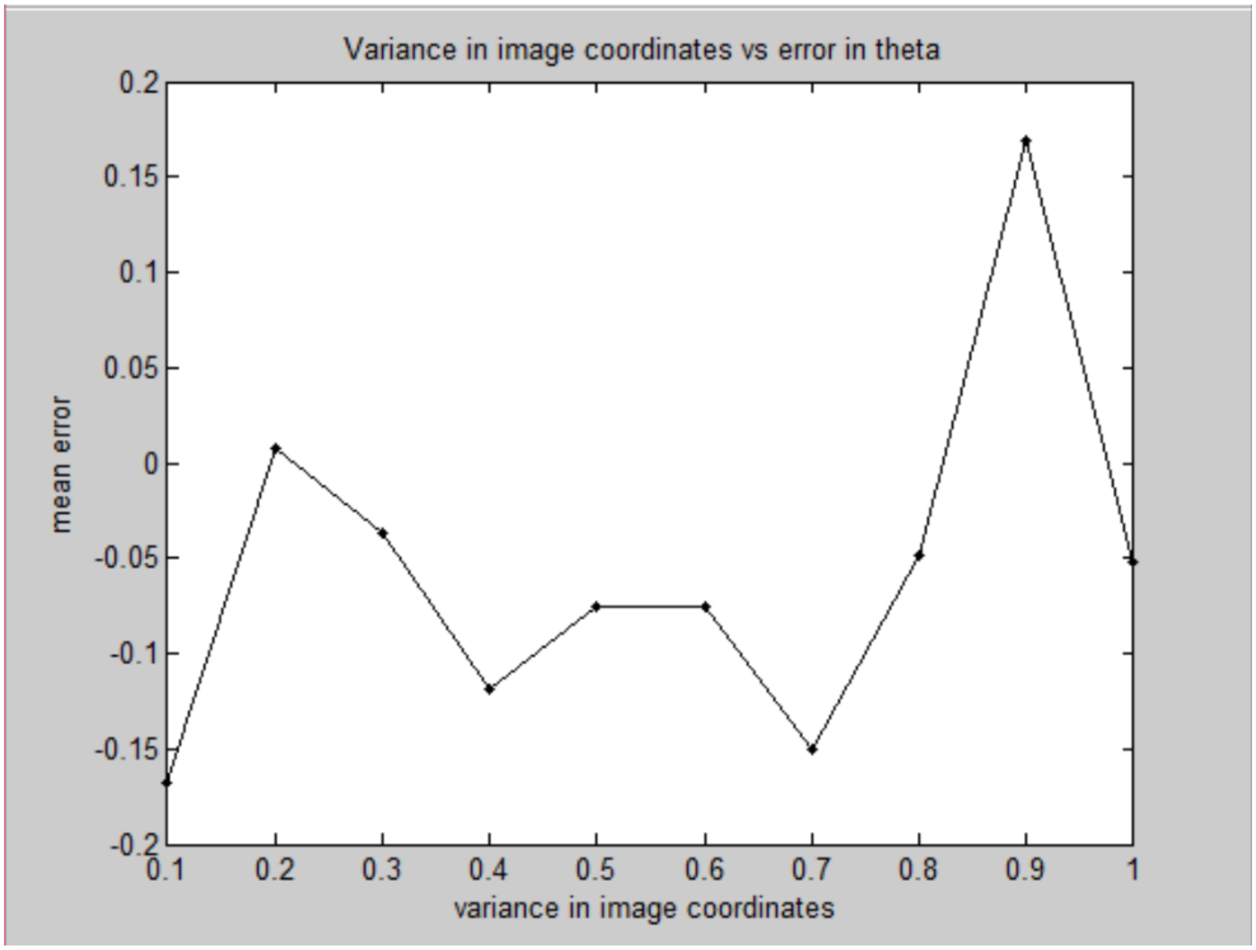
I have got these points for part 2 using my method:

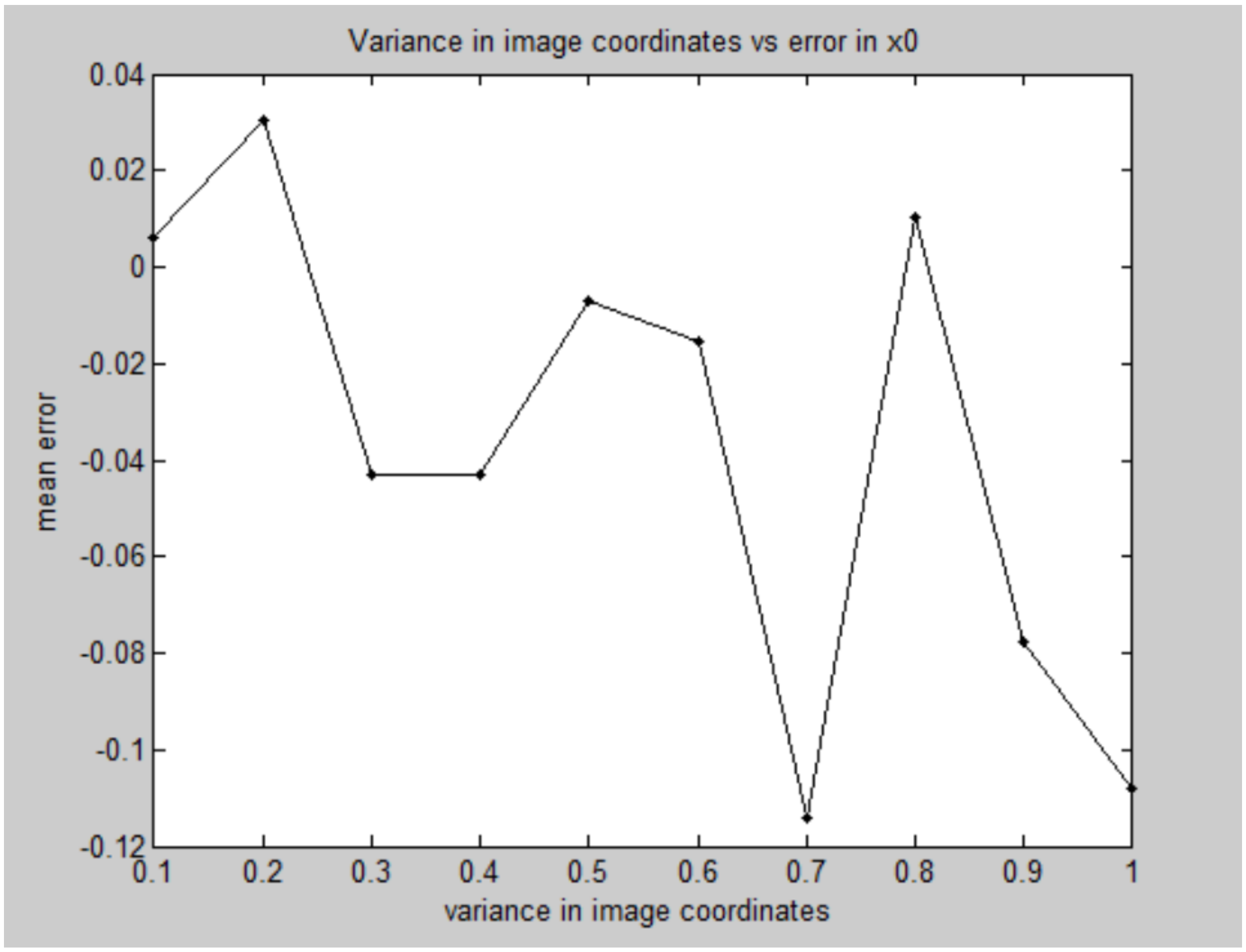


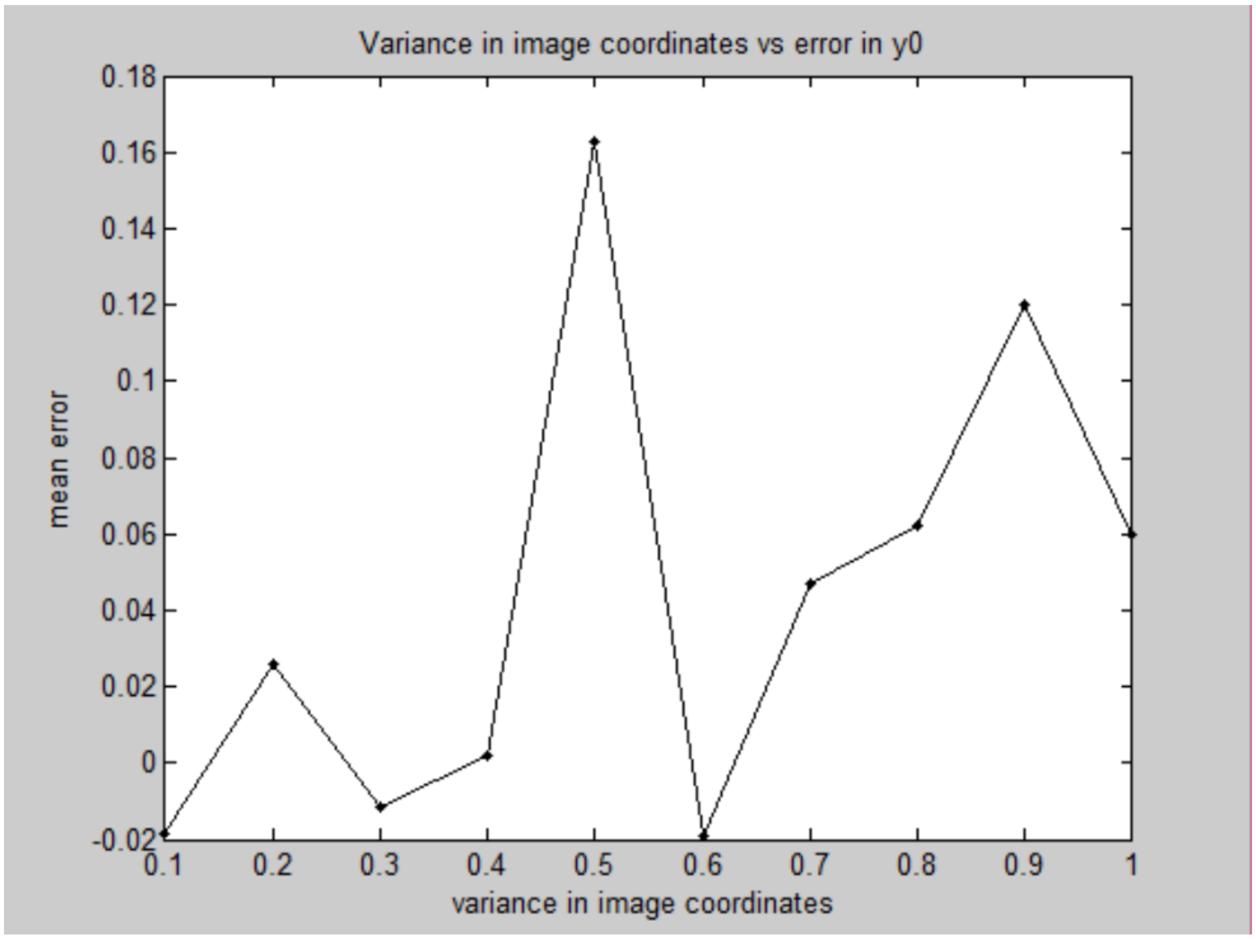
The following figure show plots for 'Variance in image coordinates vs error in PARAMETERS

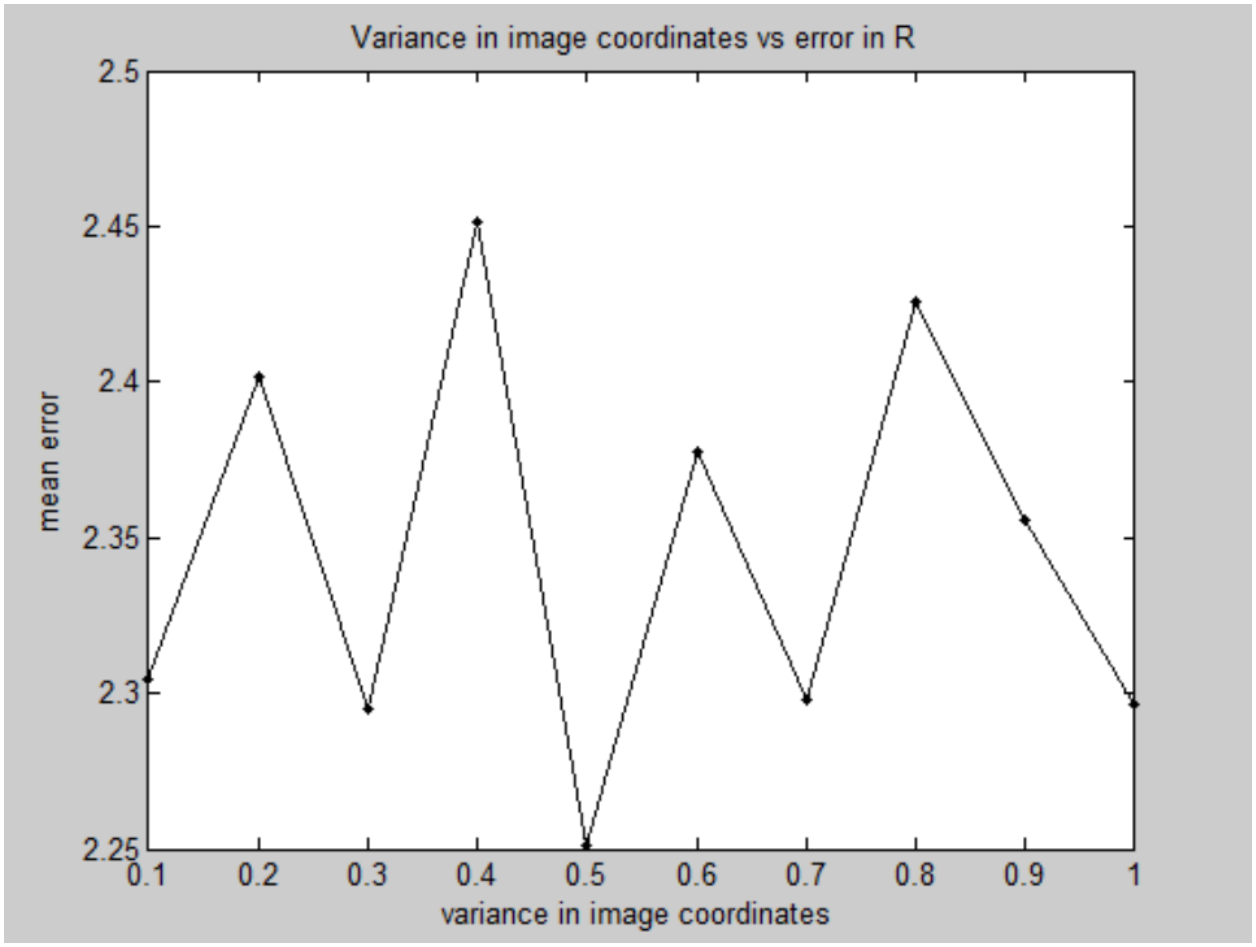



Variance in image coordinates vs error in beta
mean error
variance in image coordinates
0.95
0.9
0.85
0.8
0.75
0.7
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

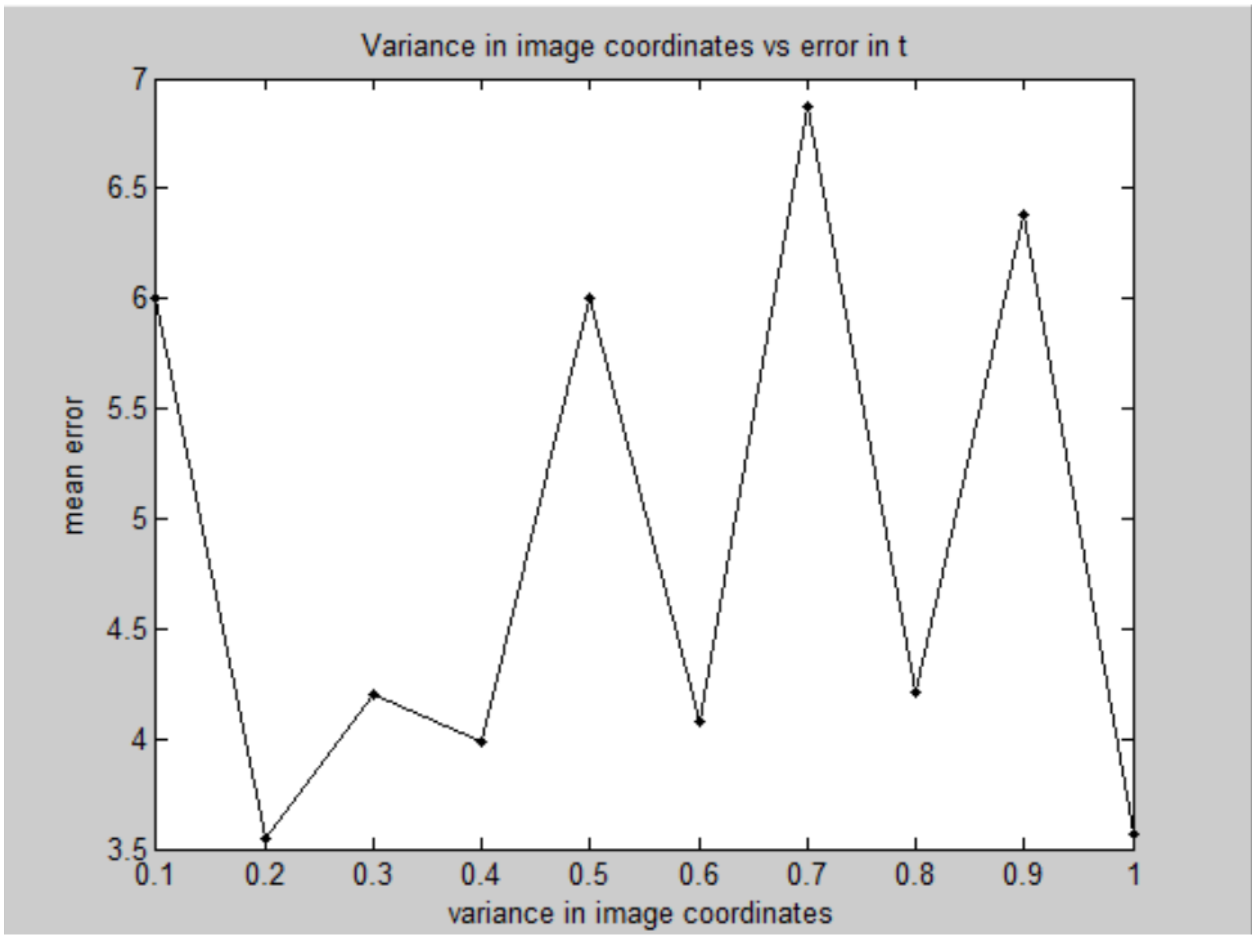

Variance in image coordinates vs error in theta
mean error
variance in image coordinates
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Variance in image coordinates vs error in x0
mean error
variance in image coordinates
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
-0.12
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

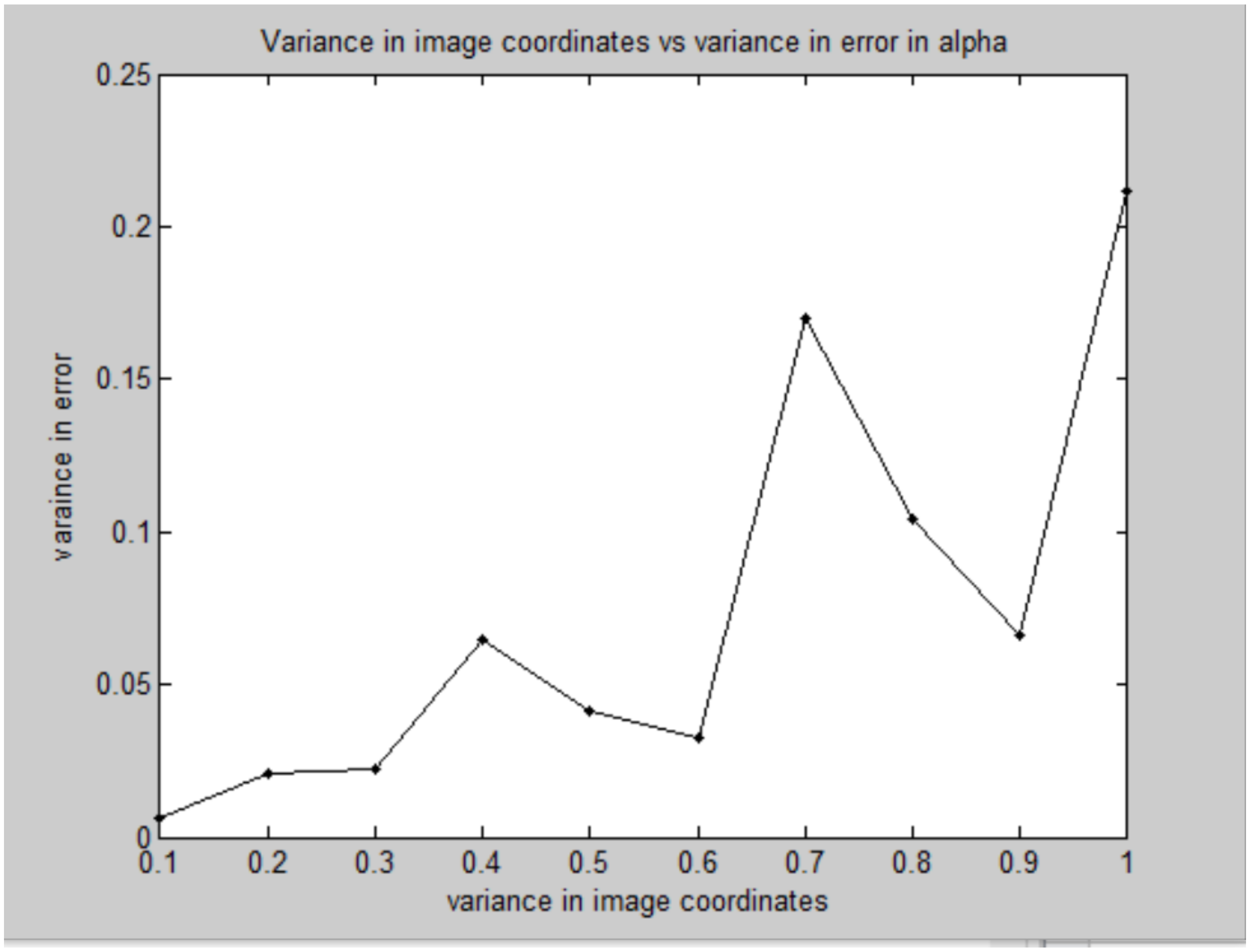

Variance in image coordinates vs error in y0
mean error
variance in image coordinates
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
-0.02
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

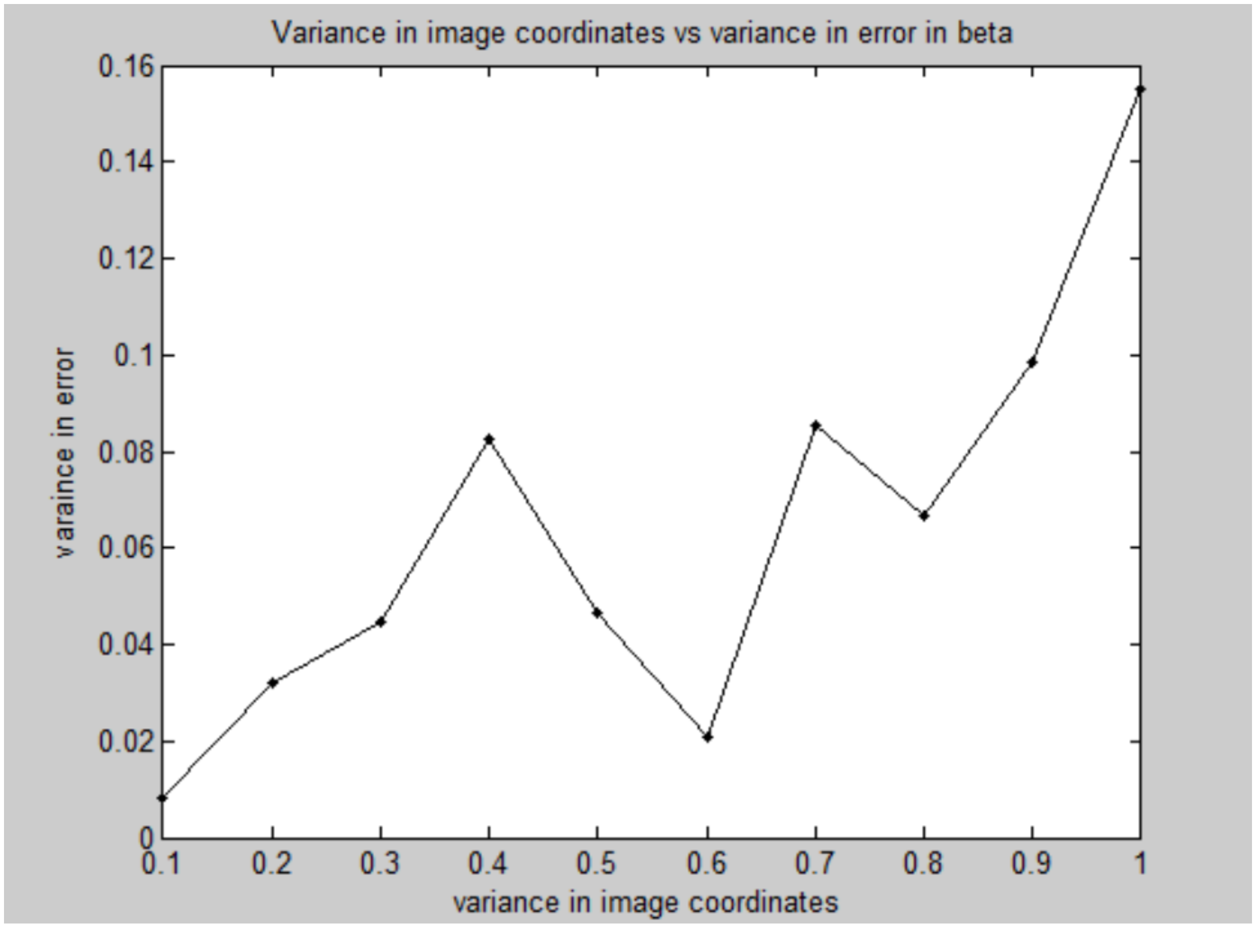


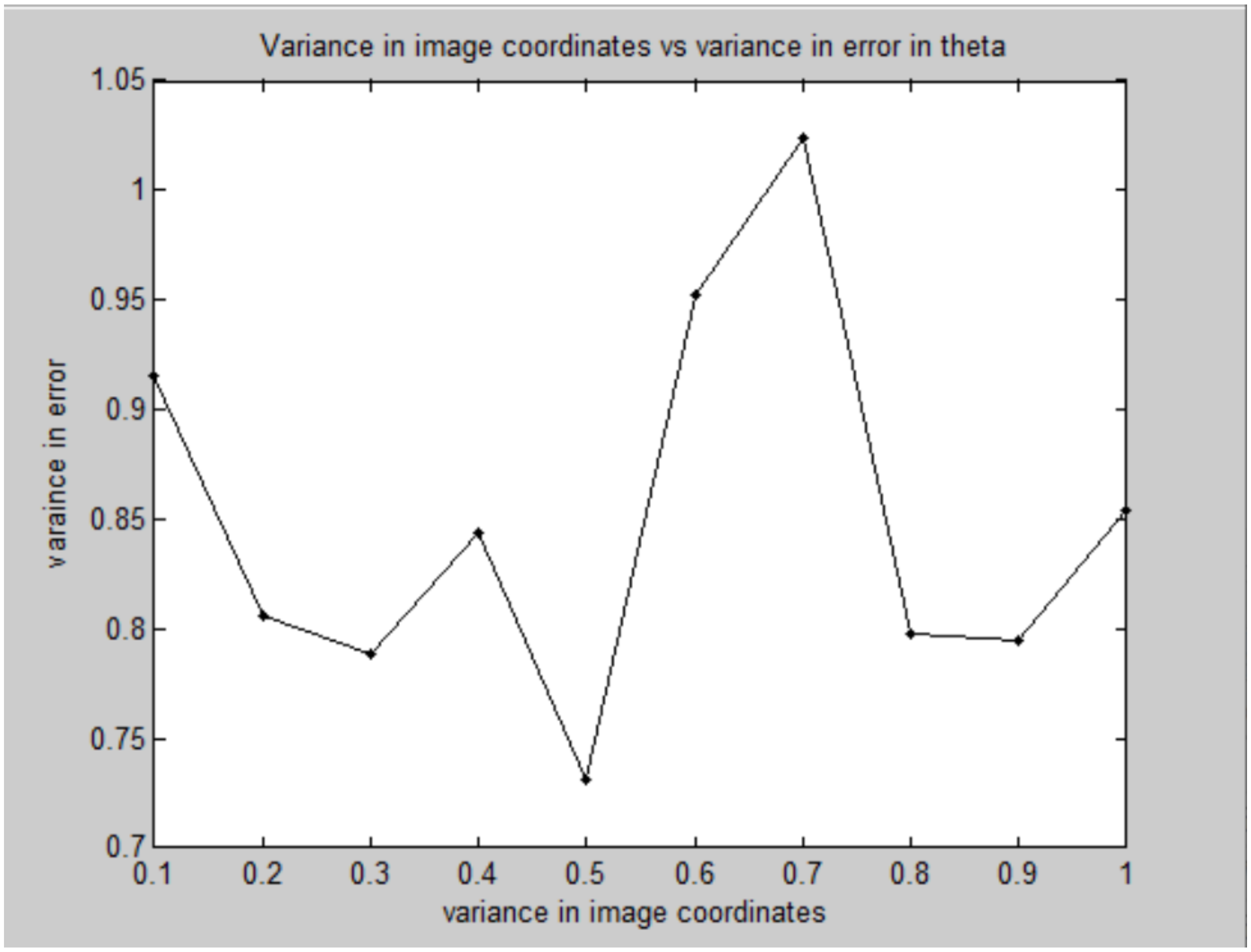
Variance in image coordinates vs error in R
mean error
variance in image coordinates
2.25
2.3
2.35
2.4
2.45
2.5
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

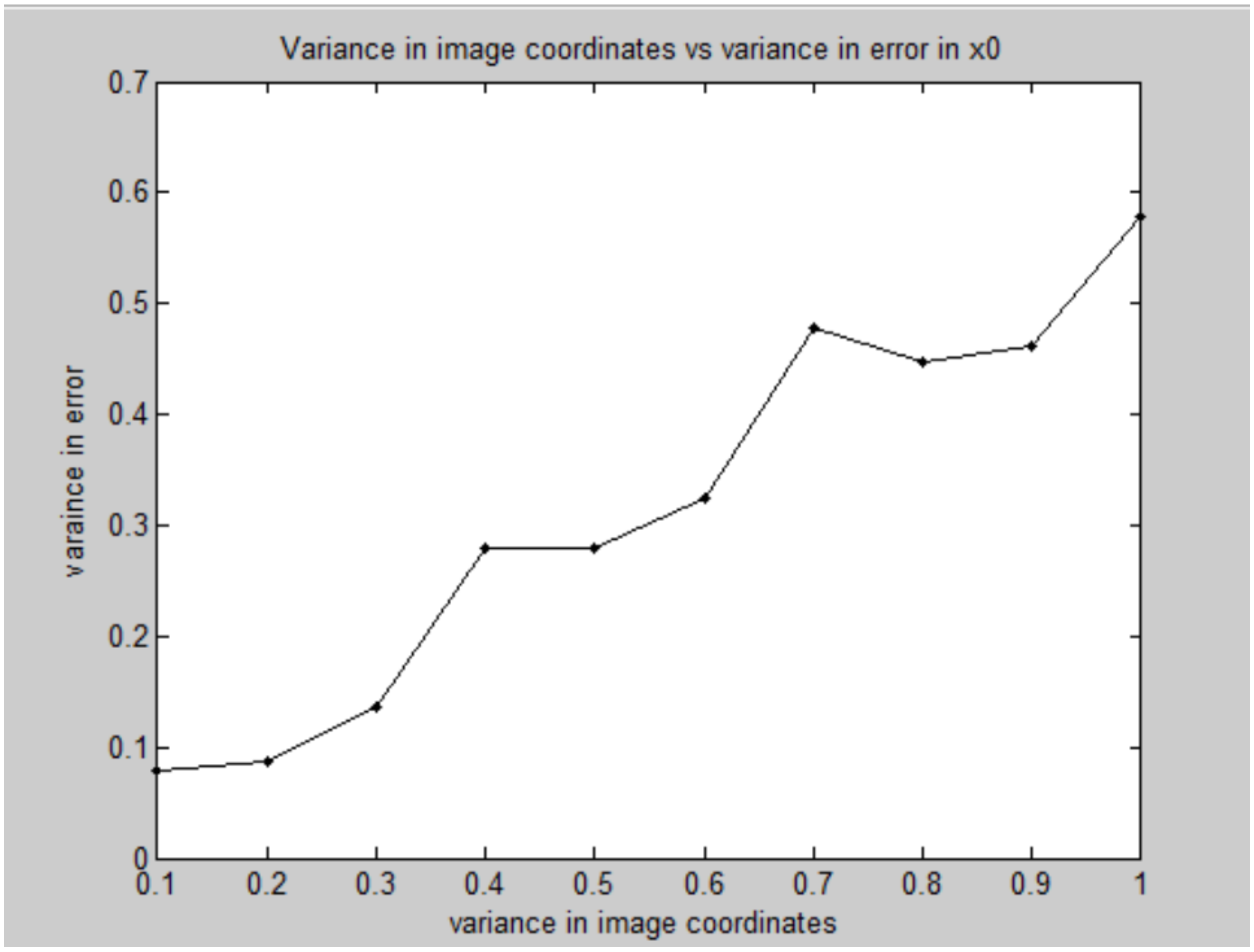

Variance in image coordinates vs error in t
mean error
variance in image coordinates
7
6.5
6
5.5
5
4.5
4
3.5
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

The following figure show plots for 'Variance in image coordinates vs variance in error in PARAMETERS

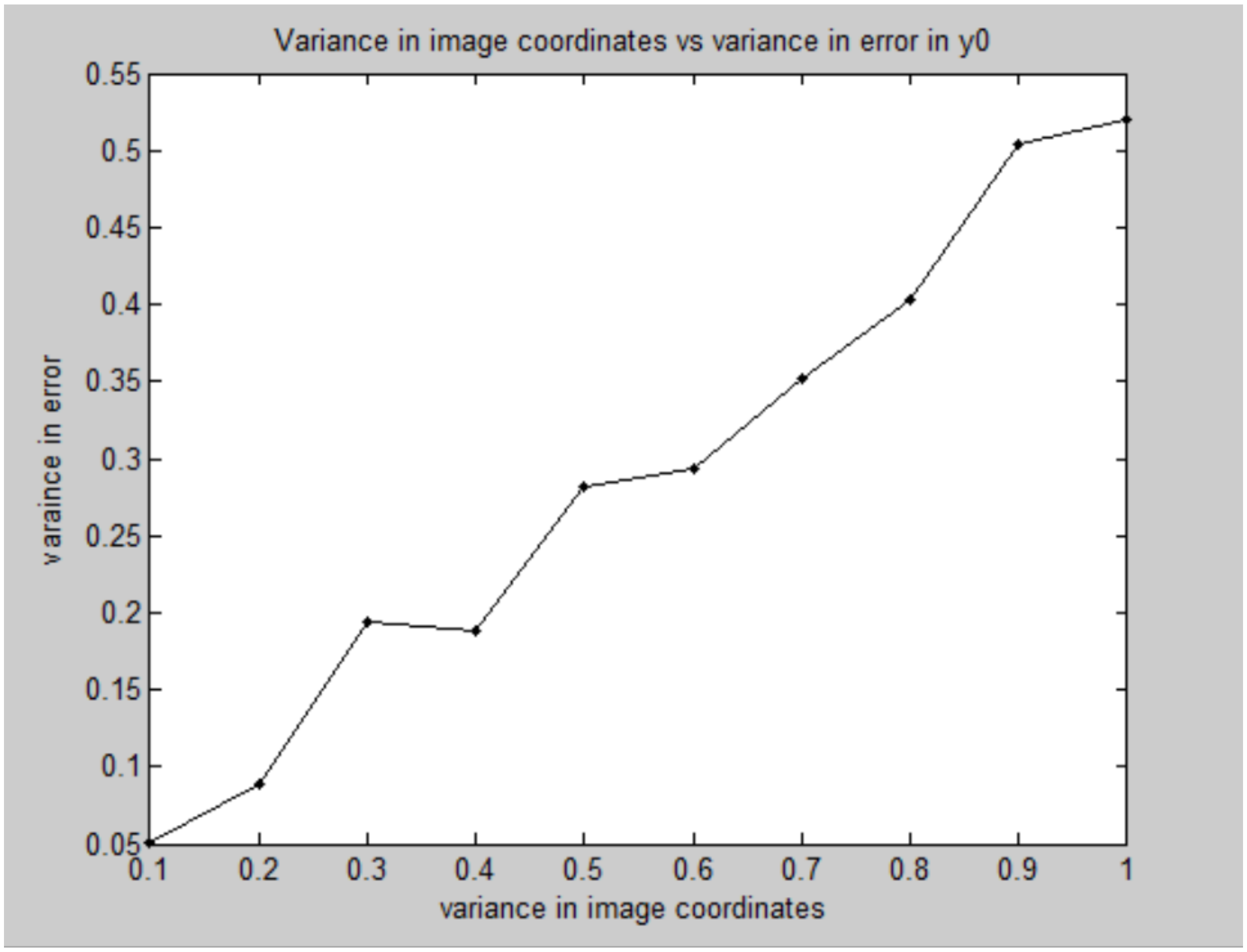



Variance in image coordinates vs variance in error in beta
varaince in error
variance in image coordinates
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

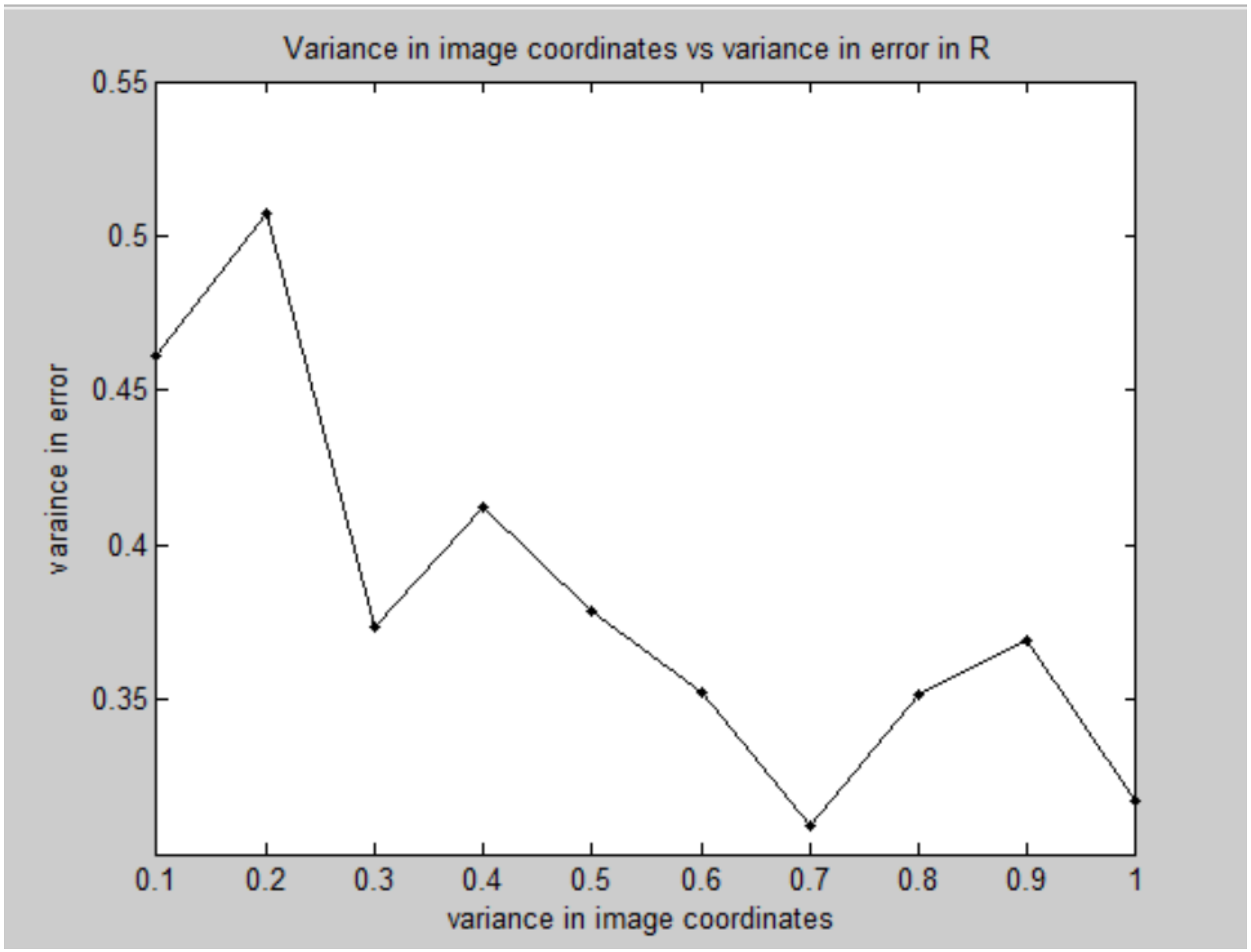

Variance in image coordinates vs variance in error in theta
varaince in error
variance in image coordinates
1.05
1
0.95
0.9
0.85
0.8
0.75
0.7
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

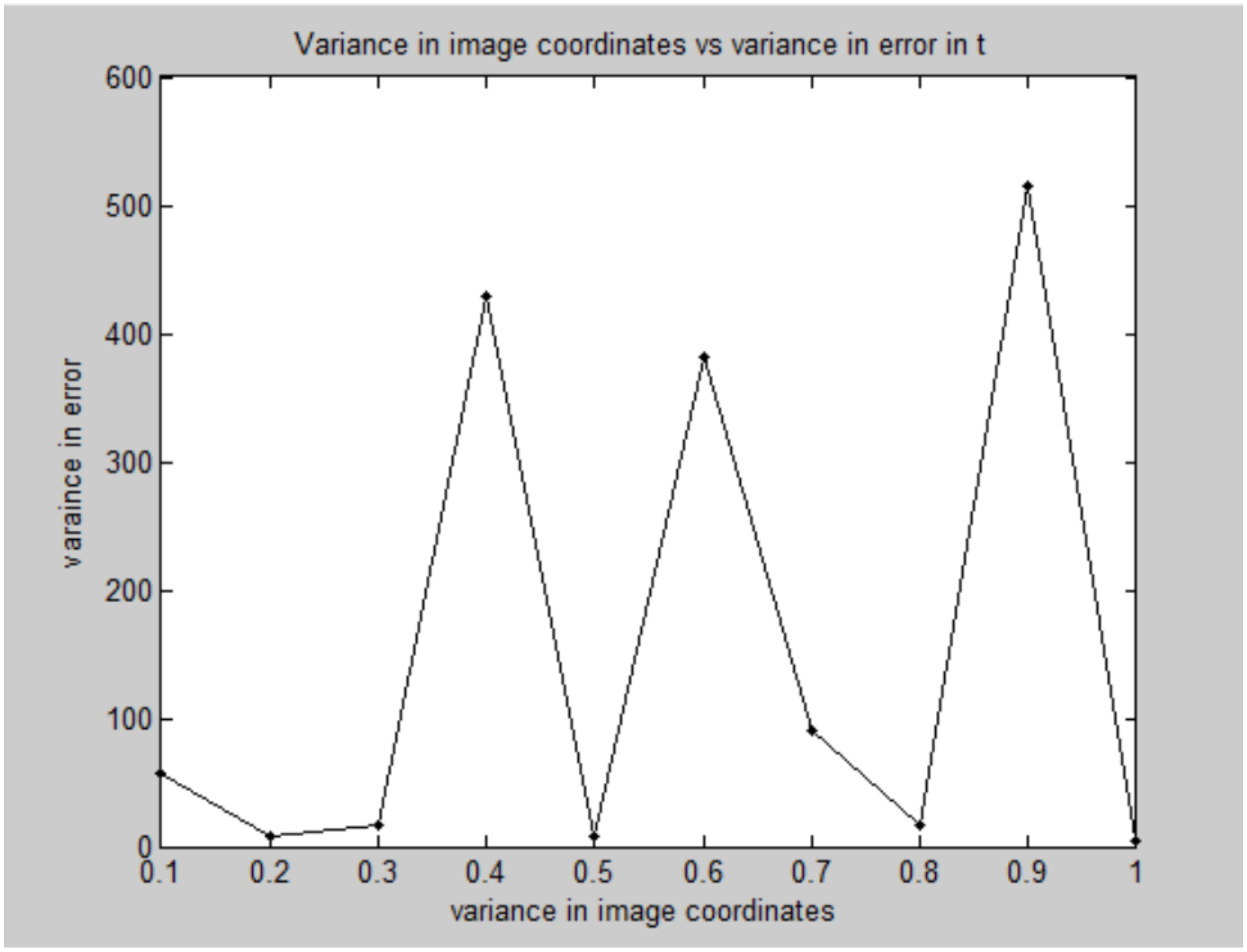


Variance in image coordinates vs variance in error in x0
varaince in error
variance in image coordinates
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



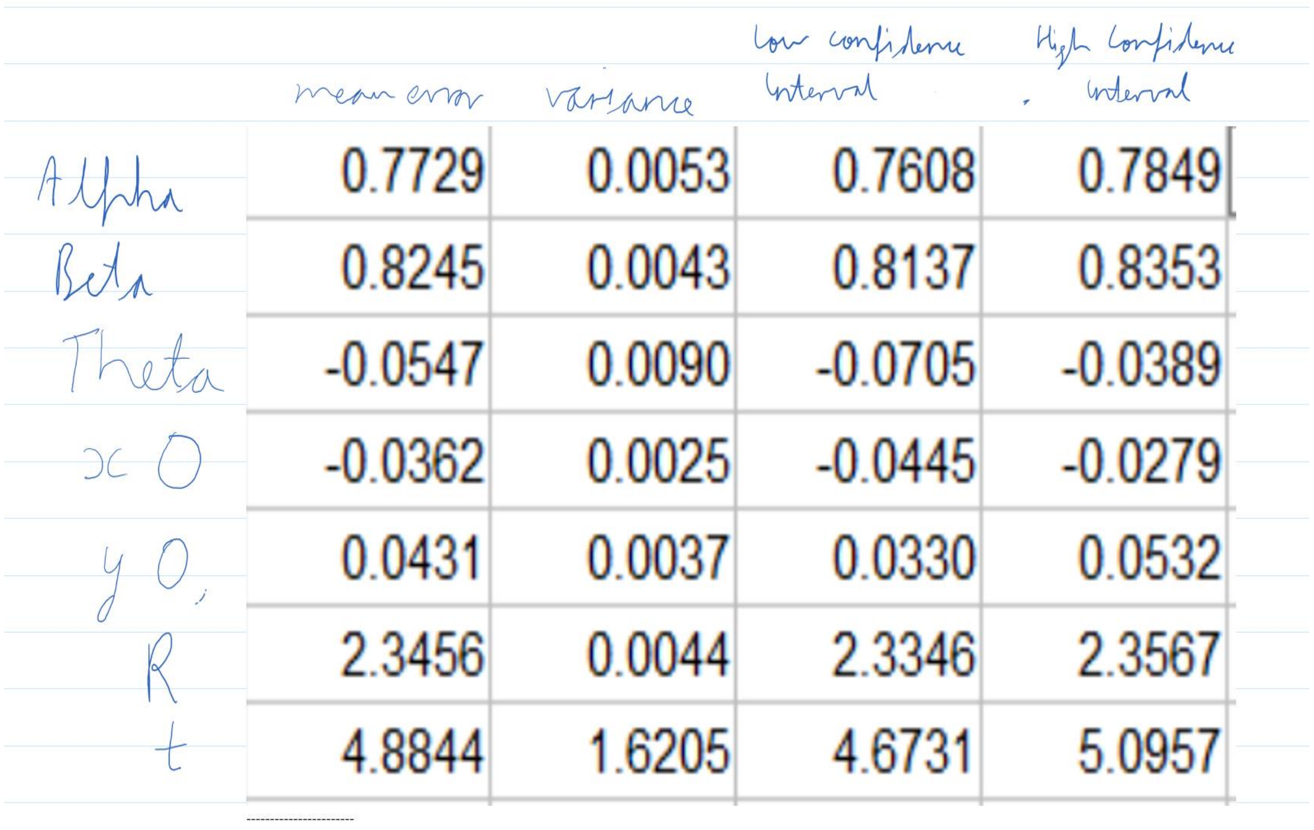
Variance in image coordinates vs variance in error in y0
varaince in error
variance in image coordinates
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5
0.55
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Variance in image coordinates vs variance in error in R
varaince in error
variance in image coordinates
0.55
0.5
0.45
0.4
0.35
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Variance in image coordinates vs variance in error in t
varaince in error
variance in image coordinates
600
500
400
300
200
100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Section 5: Analysis:

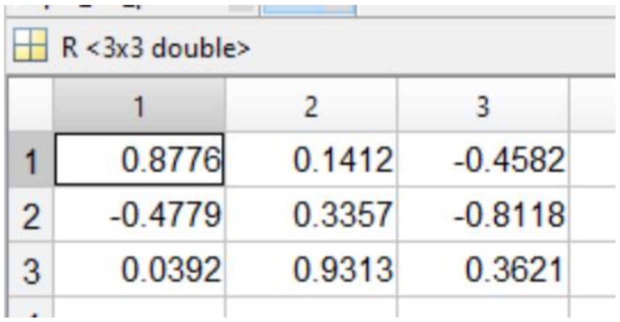
The following is the summazry of error in calibration:



	mean error	variance	Low confidence interval	High confidence interval
Alpha	0.7729	0.0053	0.7608	0.7849
Beta	0.8245	0.0043	0.8137	0.8353
Theta	-0.0547	0.0090	-0.0705	-0.0389
x 0	-0.0362	0.0025	-0.0445	-0.0279
y 0;	0.0431	0.0037	0.0330	0.0532
R	2.3456	0.0044	2.3346	2.3567
t	4.8844	1.6205	4.6731	5.0957

On applying my image corrdinates for part 2, my calibration function gives following reults:

R =



R <3x3 double>

	1	2	3
1	0.8776	0.1412	-0.4582
2	-0.4779	0.3357	-0.8118
3	0.0392	0.9313	0.3621

Alpha = 2922.48064002112

Beta = 2905.69830286640

Theta =- 1.58314410833373

X0 = 1.58314410833373

Y0 = 1065.54613781411

t= [-16.9476013900886; -22.5055940454092; -22.5055940454092]

Section 6: Interpretation:

The following are my observations:

- Mean error in alpha and beta decreases with variance in image coordinates
- Variance in error in alpha and beta increases with variance in image coordinates
- For theta, both the Curves: mean error and variance in error with variance in image coordinates are **erratic**. On different runs, an erratic behaviour is observed
- X0 and y0 follow a good increasing curve for Variance in error with variance in image coordinates
- R has a nice decreasing curve for Variance in error with variance in image coordinates.
- For t, just like theta, both the Curves: mean error and variance in error with variance in image coordinates are **erratic**. On different runs, an erratic behaviour is observed
-

Section 7: Critique:

The experiment could be improved by following ways:

- Considering a better distance matrix for R
- Applying non-linear approach to camera calibration
- Could have played around more with confidence intervals

Section 8: log

10 hours total