**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:

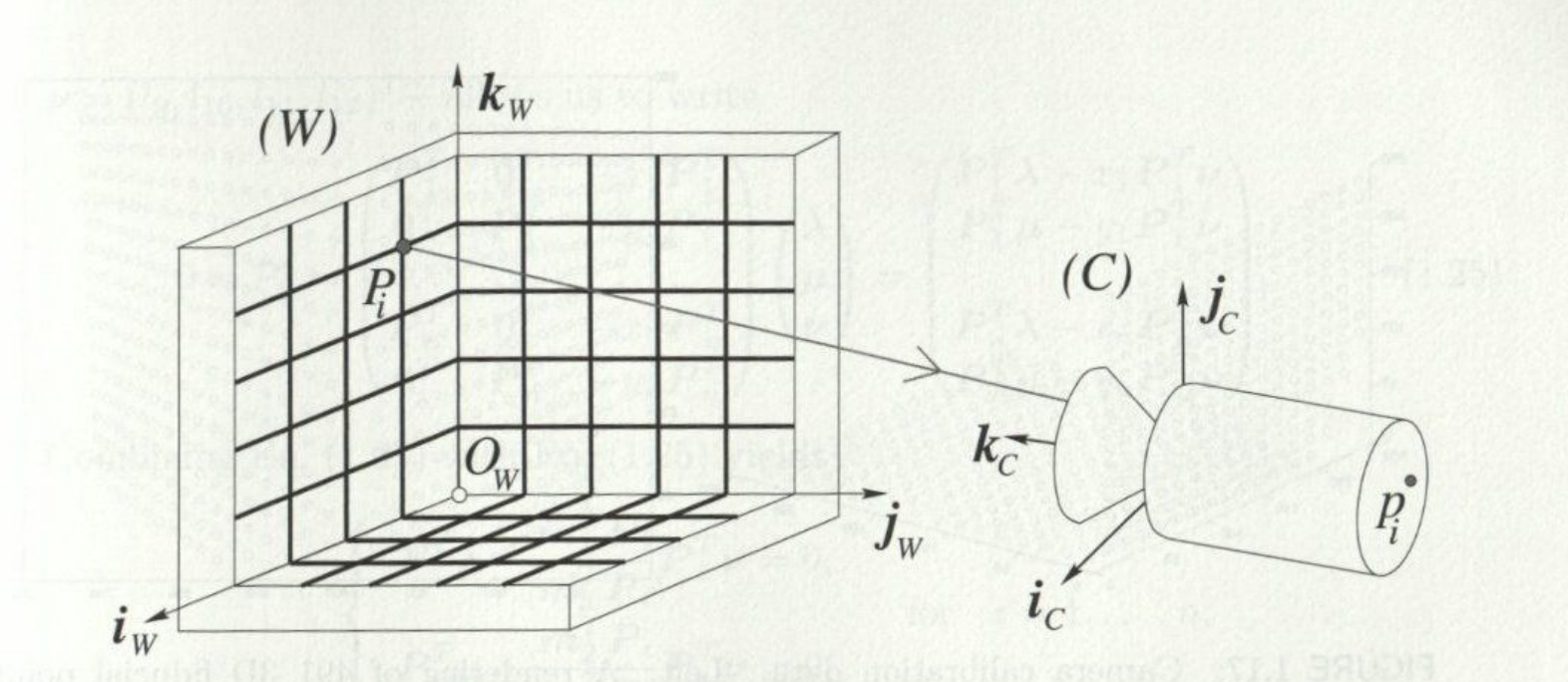


Figure 1 Camera Calibration setup. In this example, the calibration rig is formed by three grids drawn orthogonal planes

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

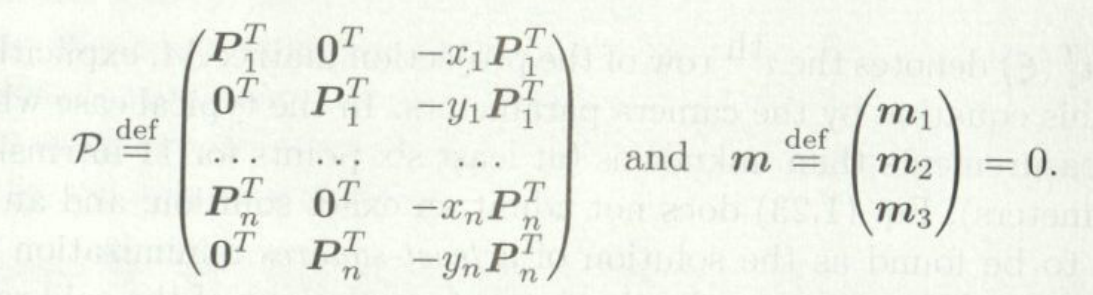
It is assumed that the Camera has a non-zero skew. According to Theorem 1 in text, matrix M is not singular, but otherwise arbitrary. We get the following equtions to solve for ith point in the scene.

Where,

* m1, m2, m3 are the three rows of the M matrix
* Pi is the corrdinates of the ith point in world frame
* Xi and yi are the coordinates of image of Pi

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

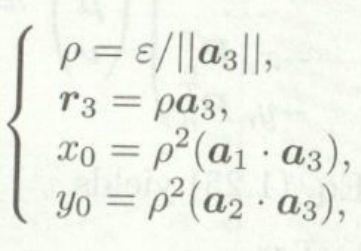
Where,

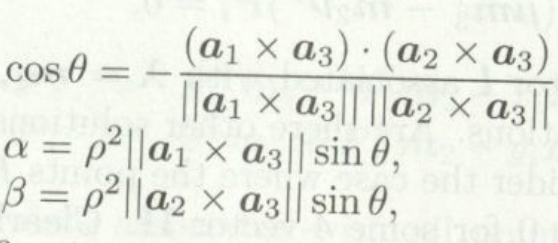


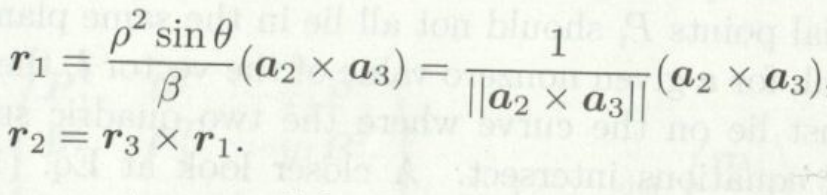
When n>=6; the homogeneous linear least square can be used to compute the value of the unit vector m (and hence the matrix M) that minimizes ||Pm||2 as the eigenvector of the 12X12 matrix PTP associated with its smallest eigen value.

A note about degenerate cases: The fiducial points Pi should not lie in the same plane

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









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

1. Form A and b which will be used

A = M\_hat(:,1:3);

b = M\_hat(:,4);

1. 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';

1. 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'];

1. 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);

1. Set vs = [0.1:0.1:1];
2. Set num\_vs = length(vs) ;
3. Loop v\_index = 1:num\_vs % set variance
   1. Set v = vs(v\_index);
   2. Set Error\_alpha = [];Error\_beta = [];Error\_theta = []; Error\_x0 = []; Error\_y0 =[]; Error\_R = [];Error\_t = [];
   3. Loop for i=1:NumOfTrials
   4. pts\_im\_Noisy(1,:) = pts\_im(1,:) + sqrt(v) \* randn(1 , size(pts\_im, 2));
   5. pts\_im\_Noisy(2,:) = pts\_im(2,:) + sqrt(v) \* randn(1 , size(pts\_im, 2));
   6. pts\_im\_Noisy(3,:) = pts\_im(3,:);
   7. [alphaN,betaN,thetaN,x0N,y0N,RN,tN]...

= CS5320\_calibrate(pts\_im\_Noisy , pts\_world);

* 1. Error\_alpha = [Error\_alpha ; alpha-alphaN];
  2. 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

1. 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,:));



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);

1. End the outer loop

1. 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);

1. 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:**

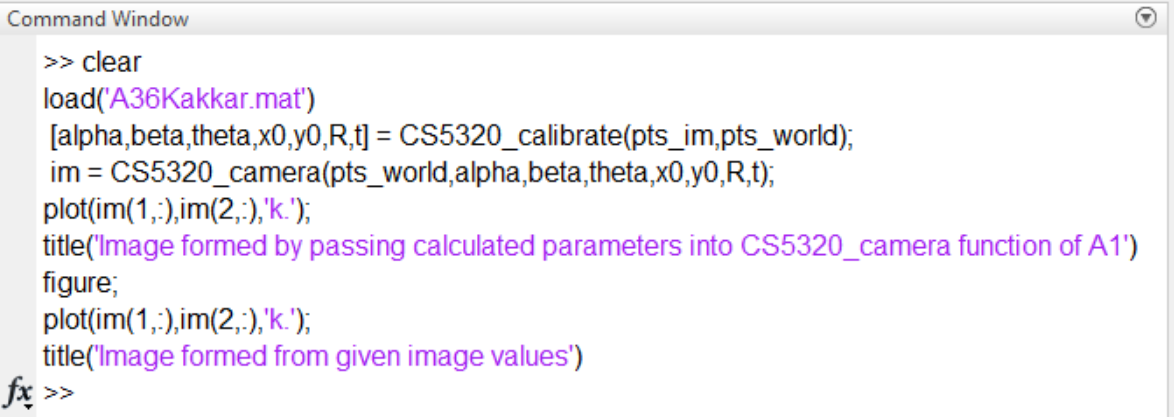
Using Berkeley website, I got the following:

* 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.

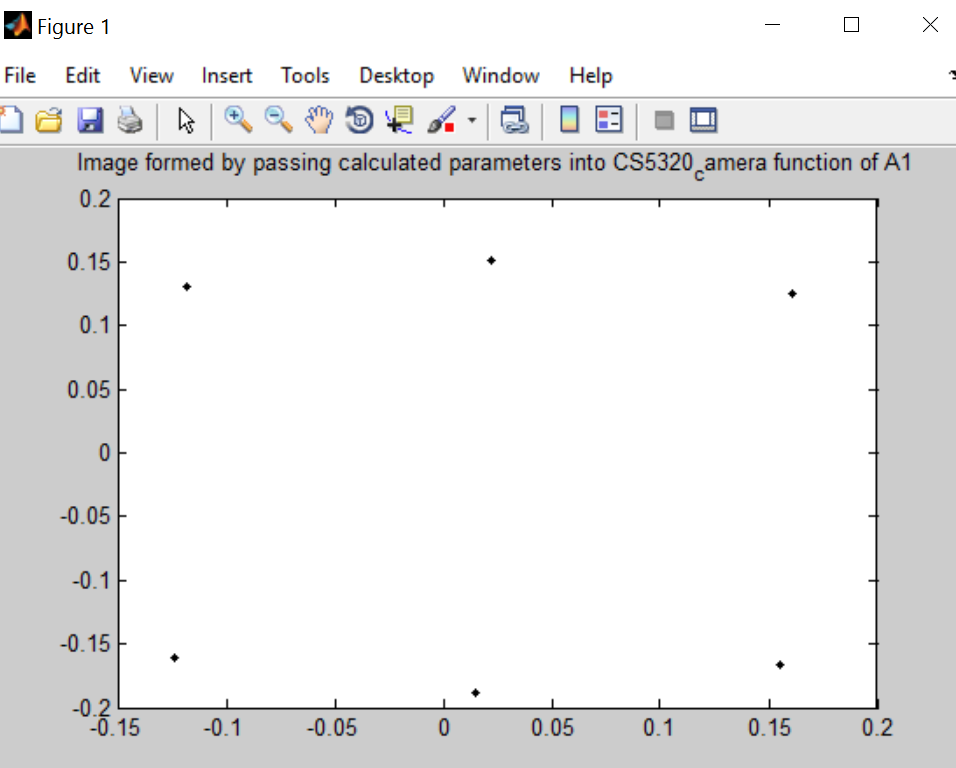
**Section 3: Verification:**

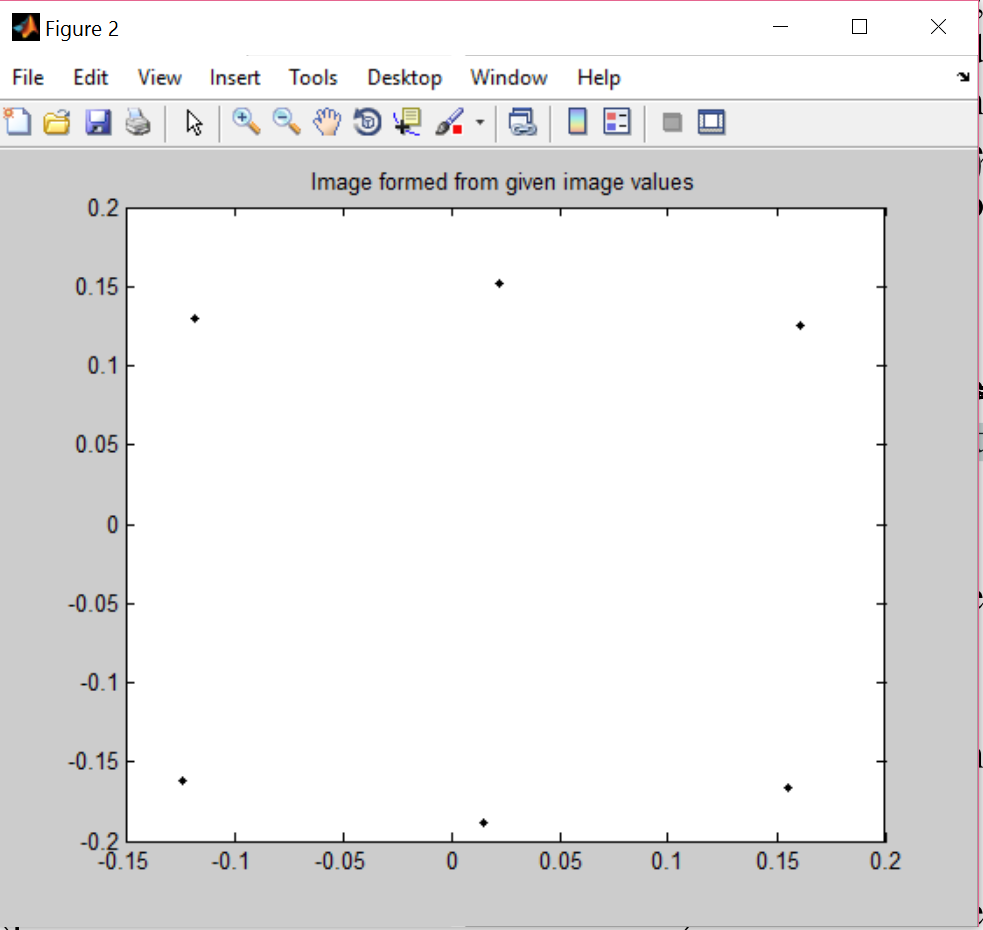
Testing CS5320\_calibrate

* 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.



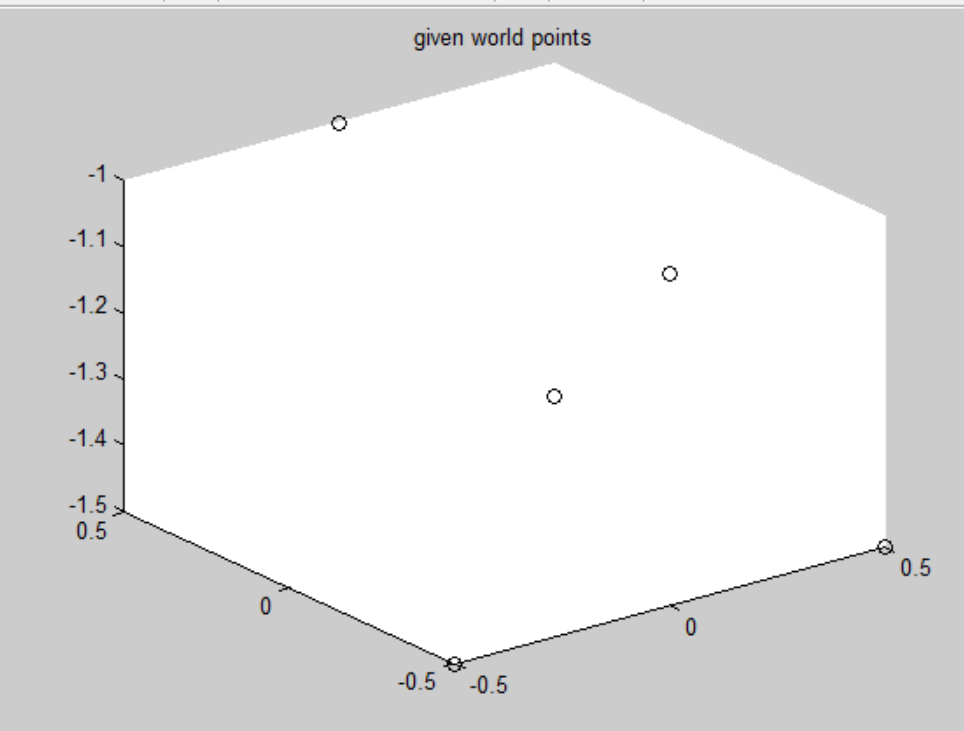
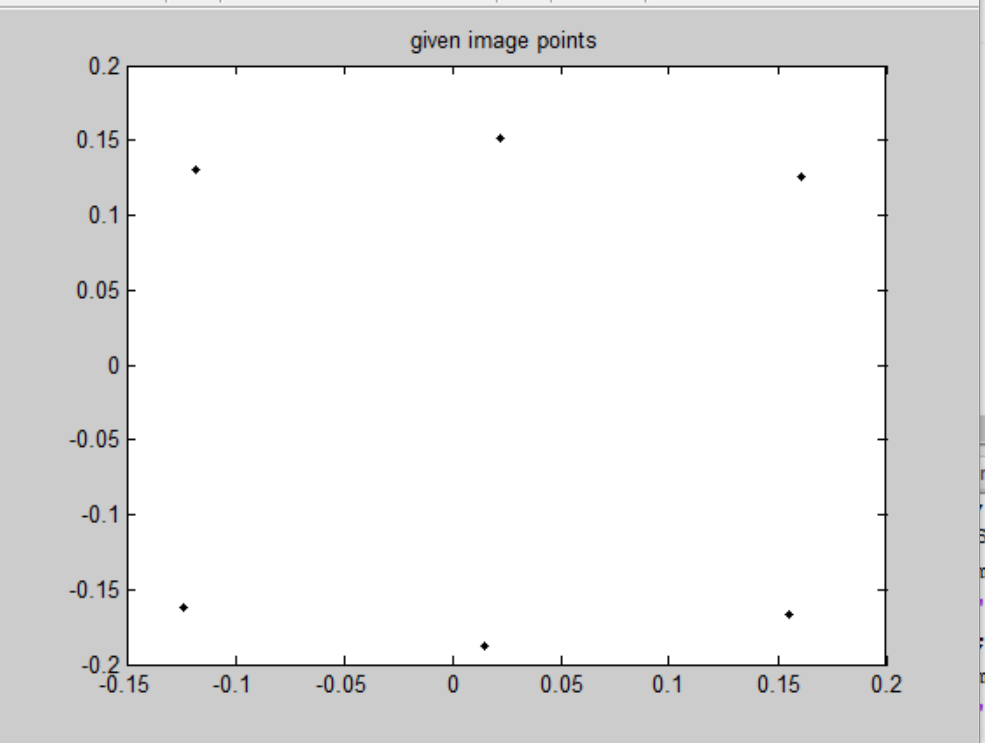
I am getting my image coordinates back as seen from following two images.



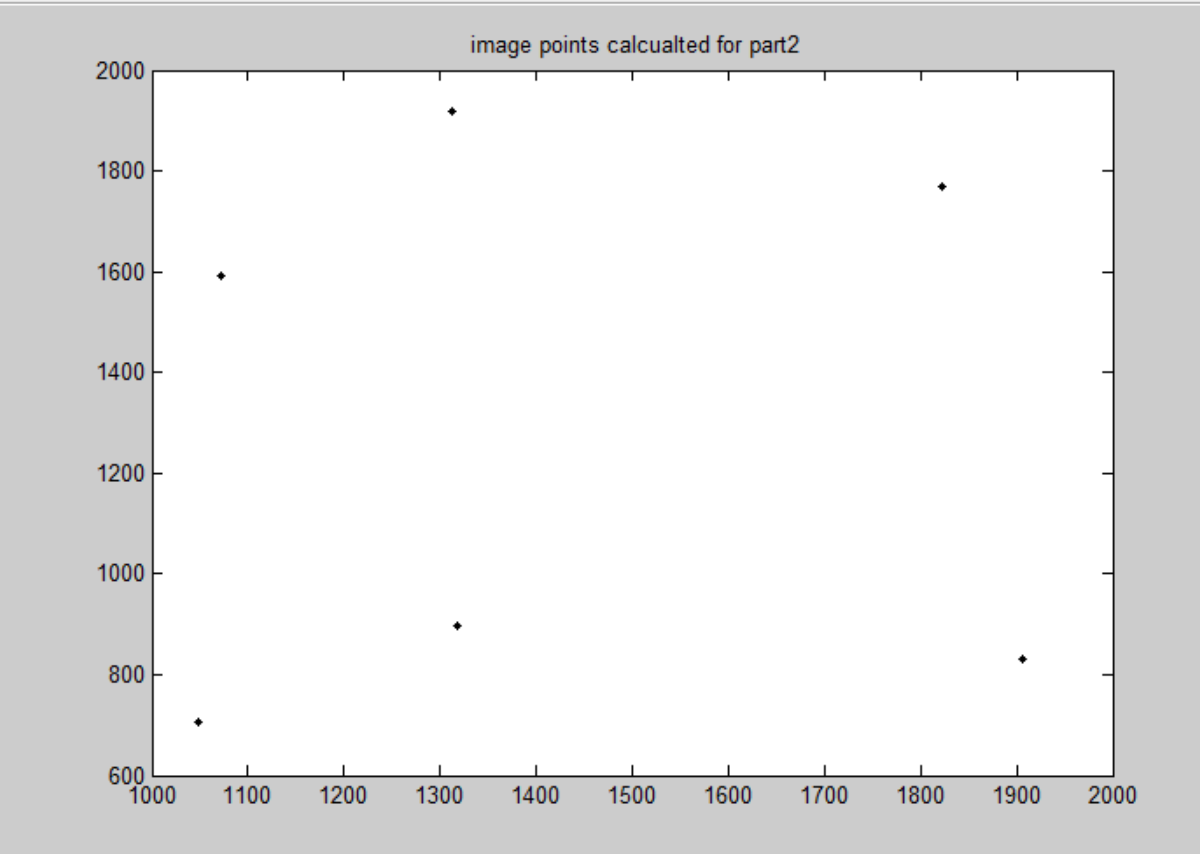


**Section 4: Data:**

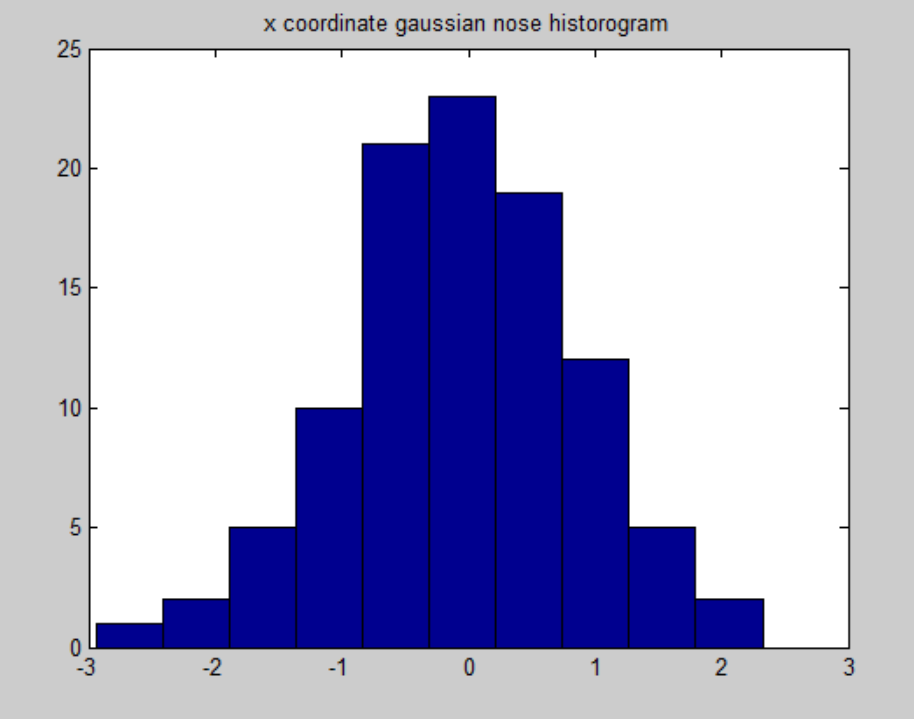
Following figure show the image and world coordinate given in mat file, that go into the CS5320\_calibrate function

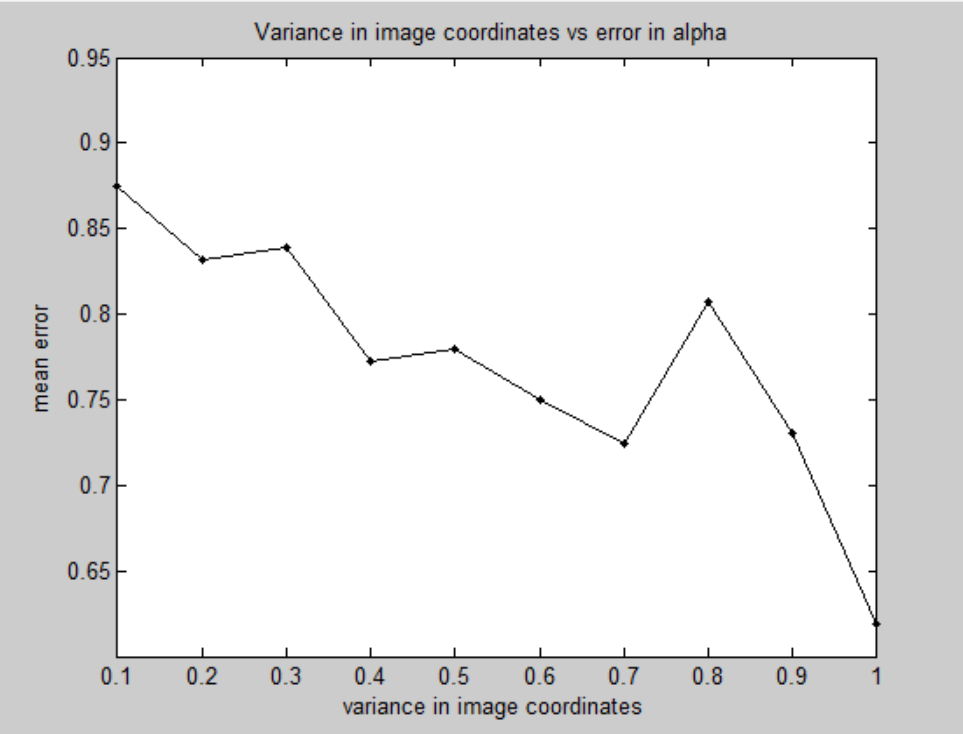
I have got these points for part 2 using my method:

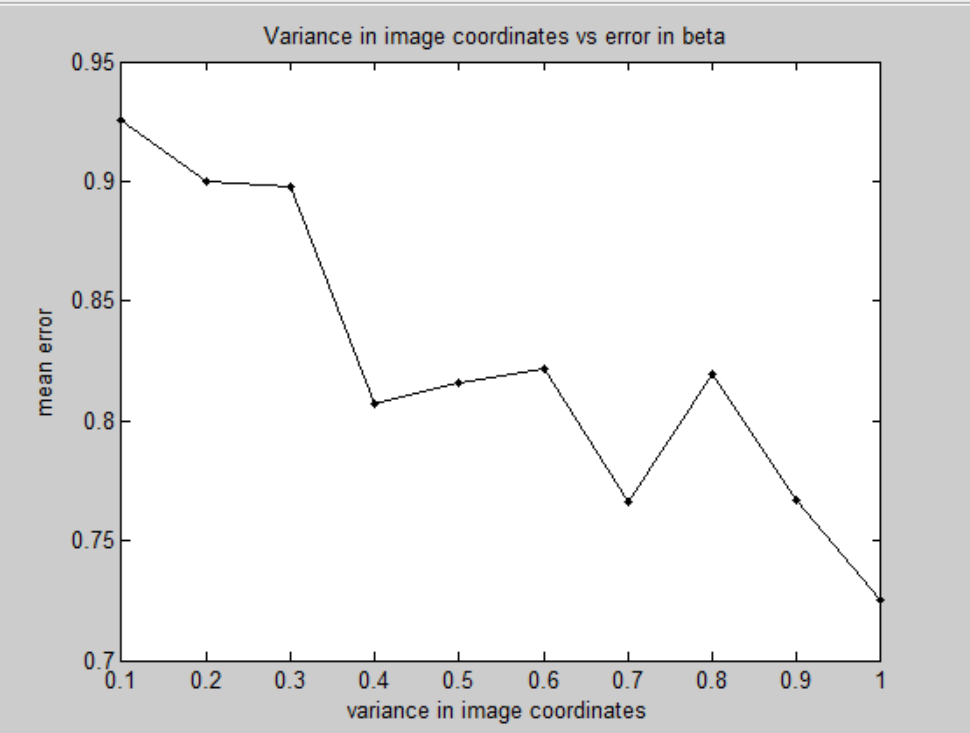


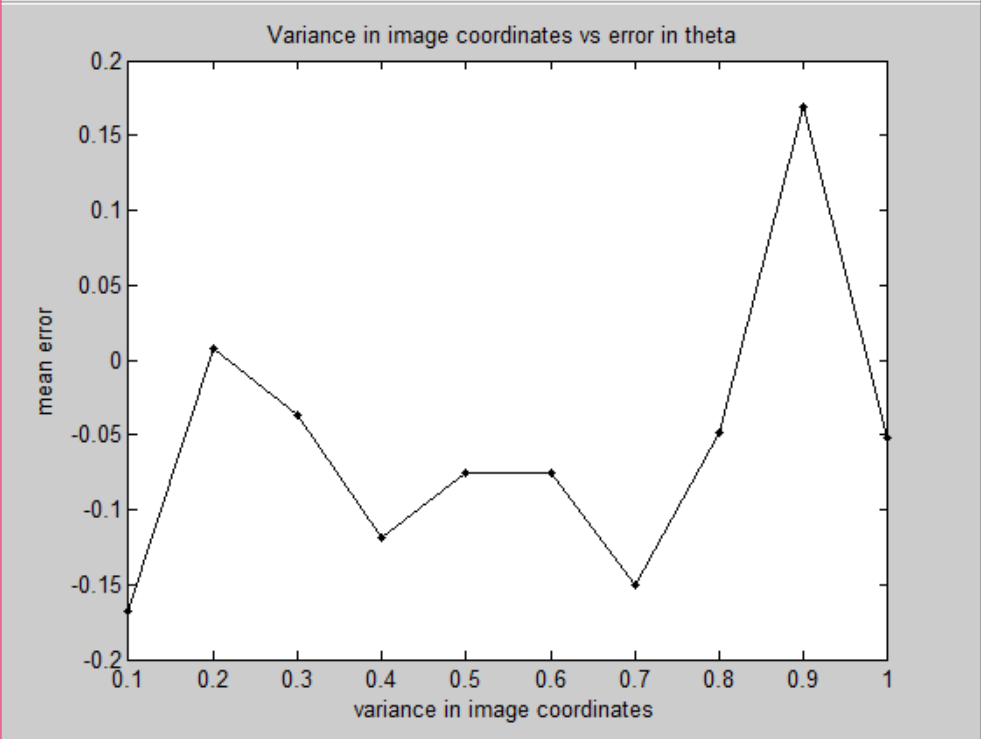
Getting a good historogram after adding noise to image points. The following shows for x corrdinates:

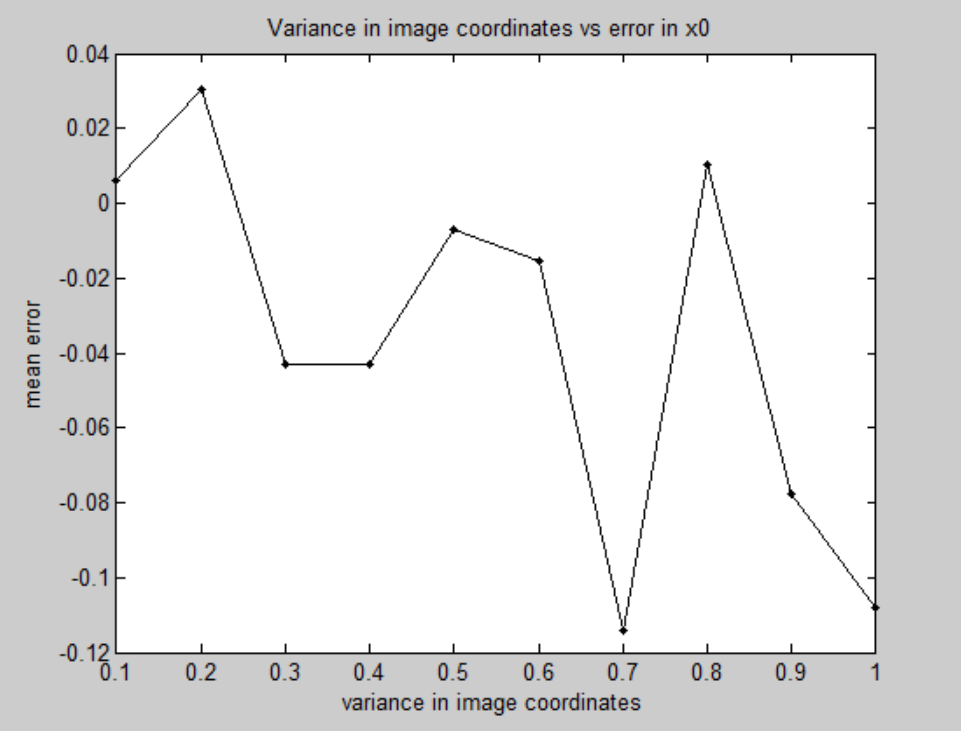


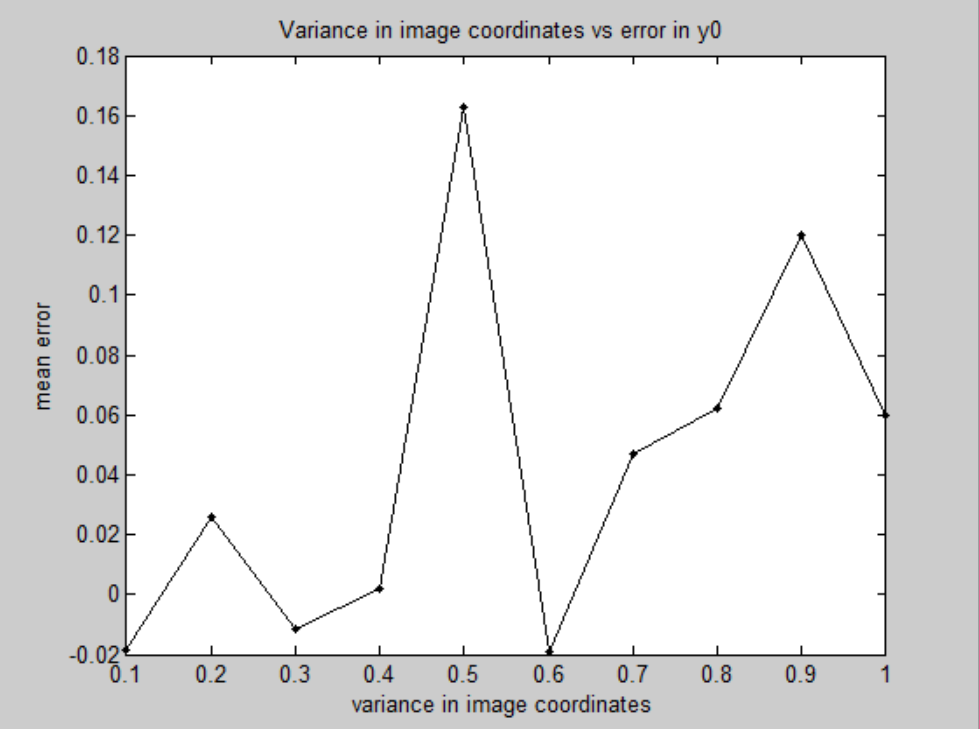
The following figure show plots for 'Variance in image coordinates vs error in PARAMETERS

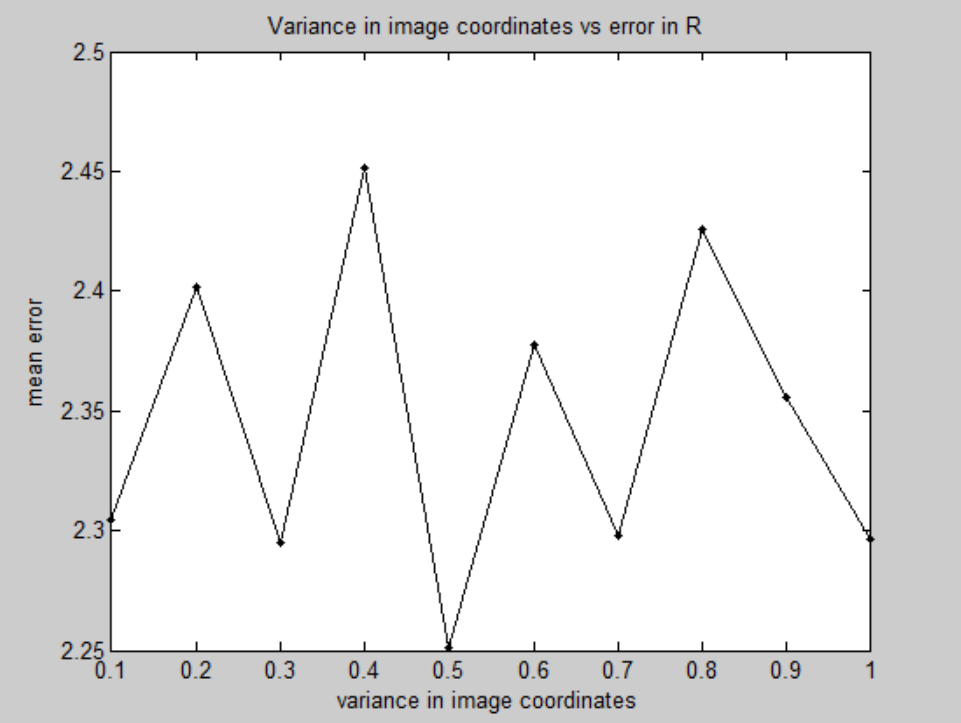


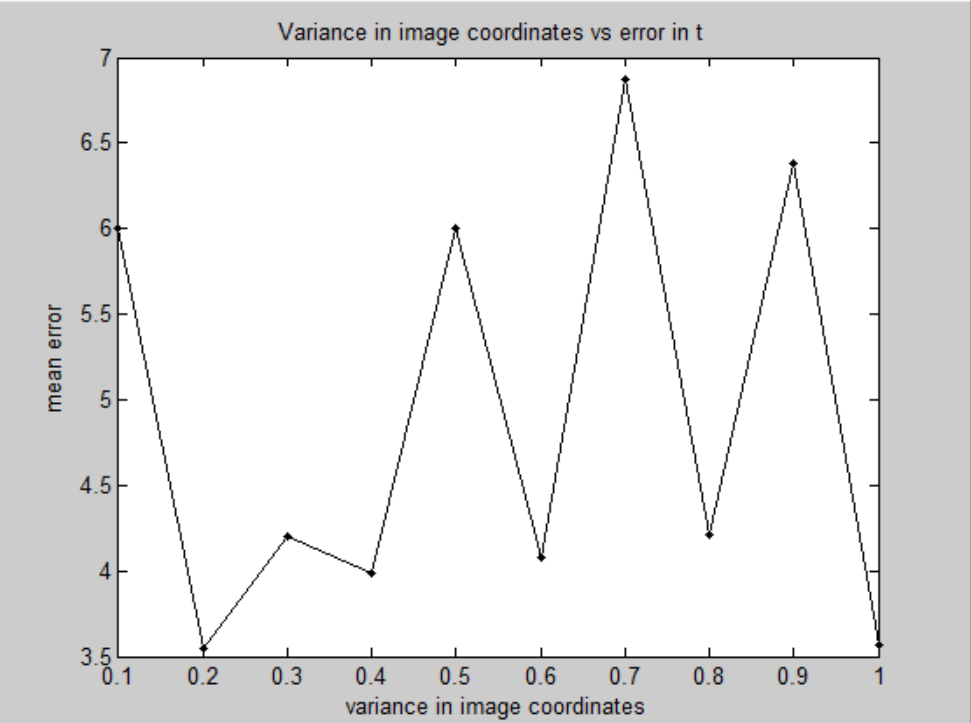




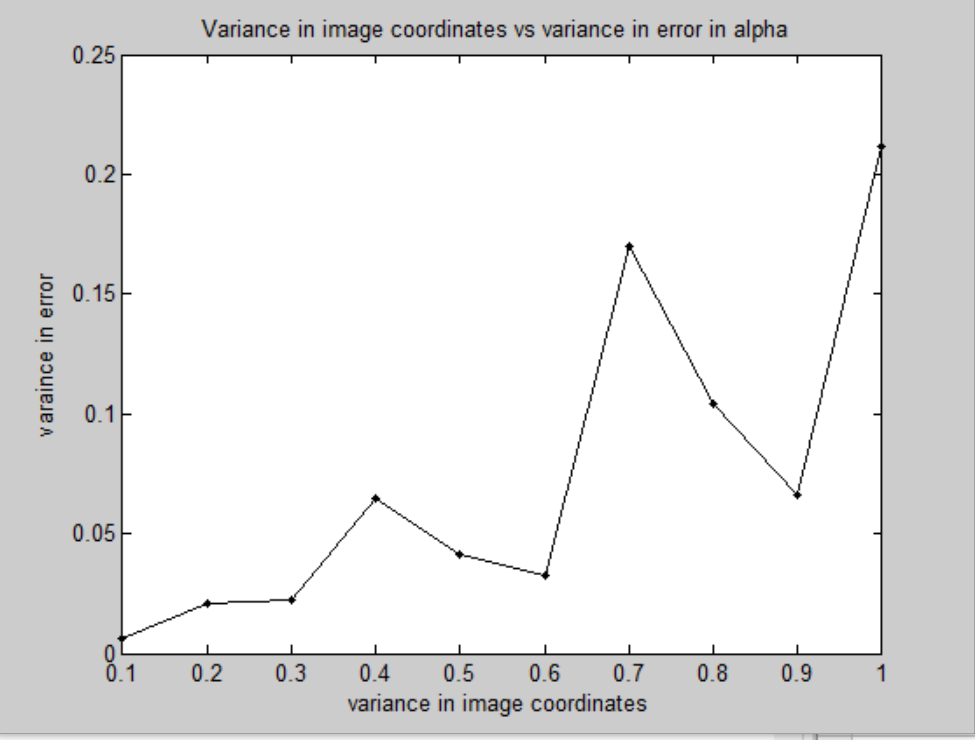


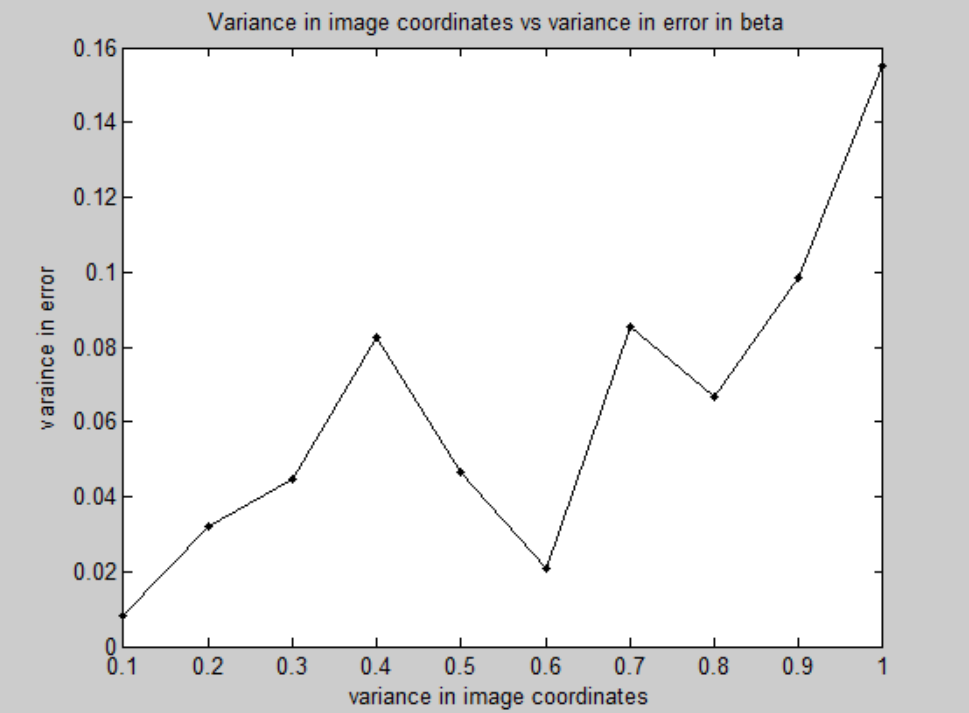


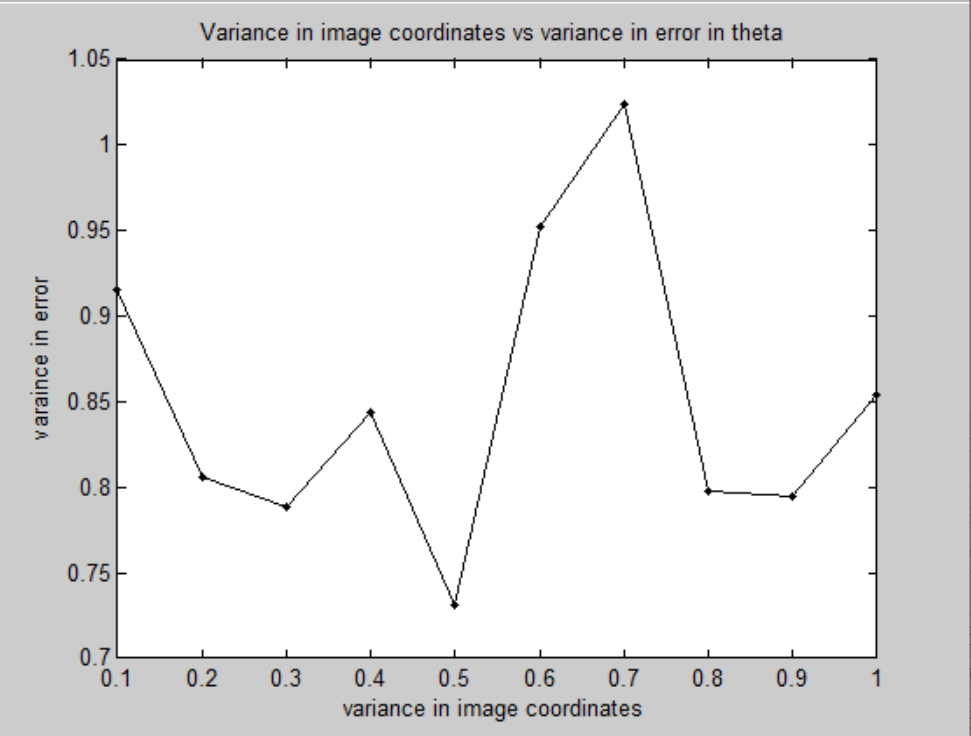


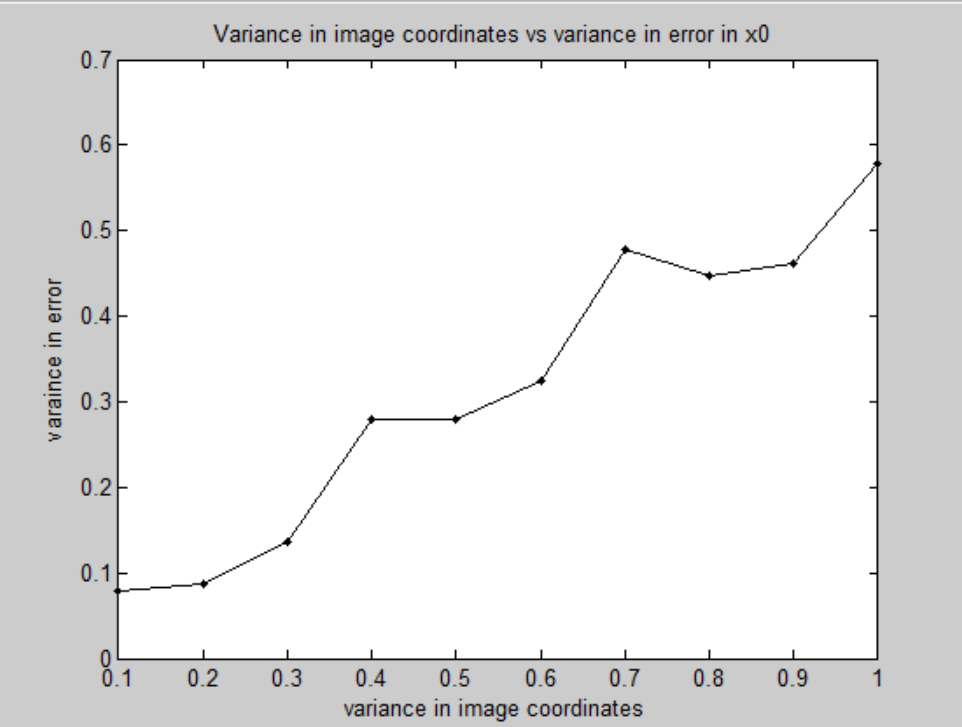


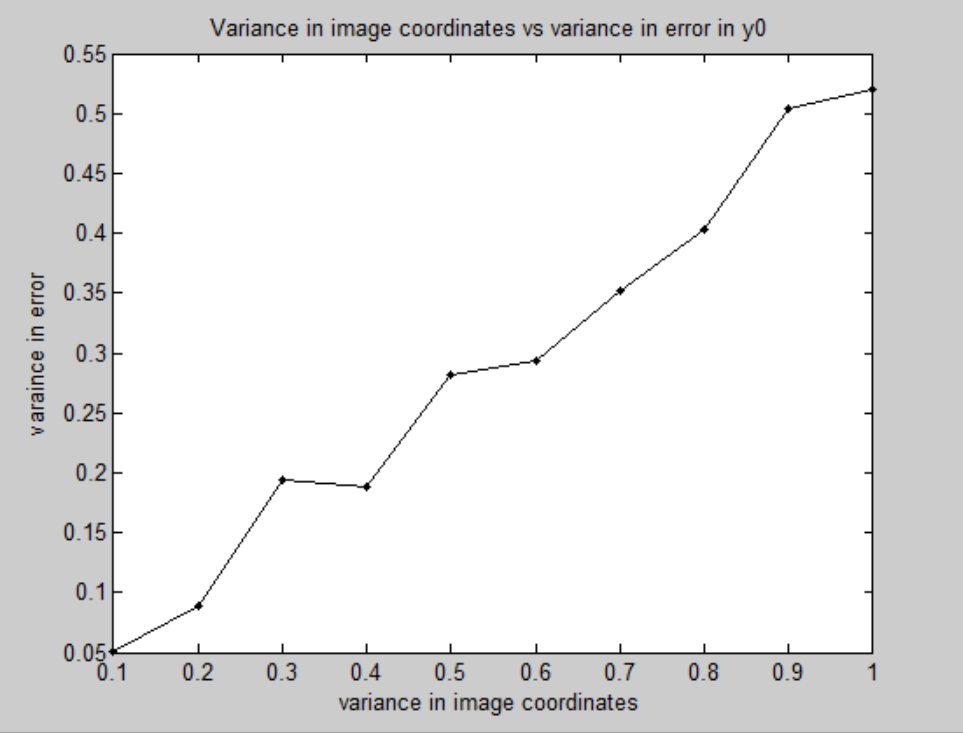
The following figure show plots for 'Variance in image coordinates vs variance in error in PARAMETERS

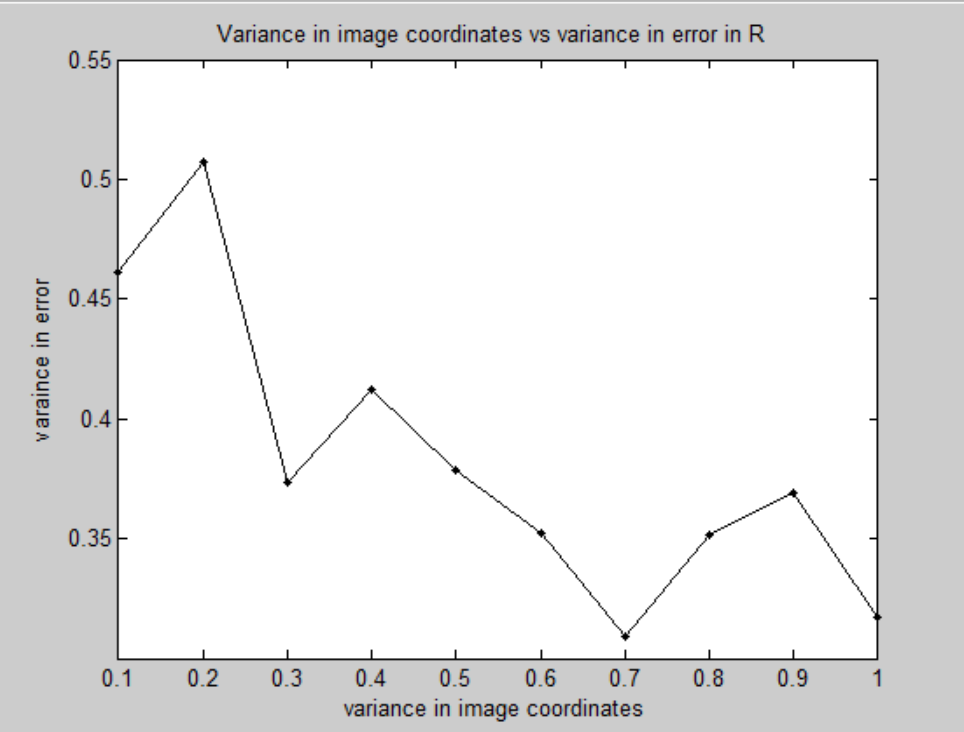


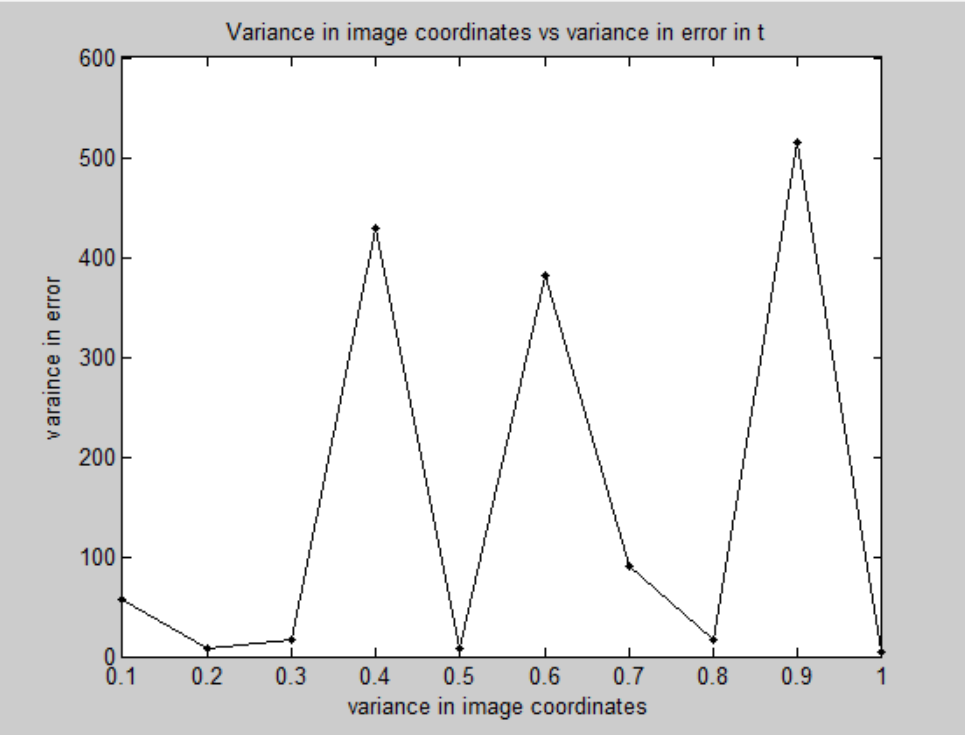






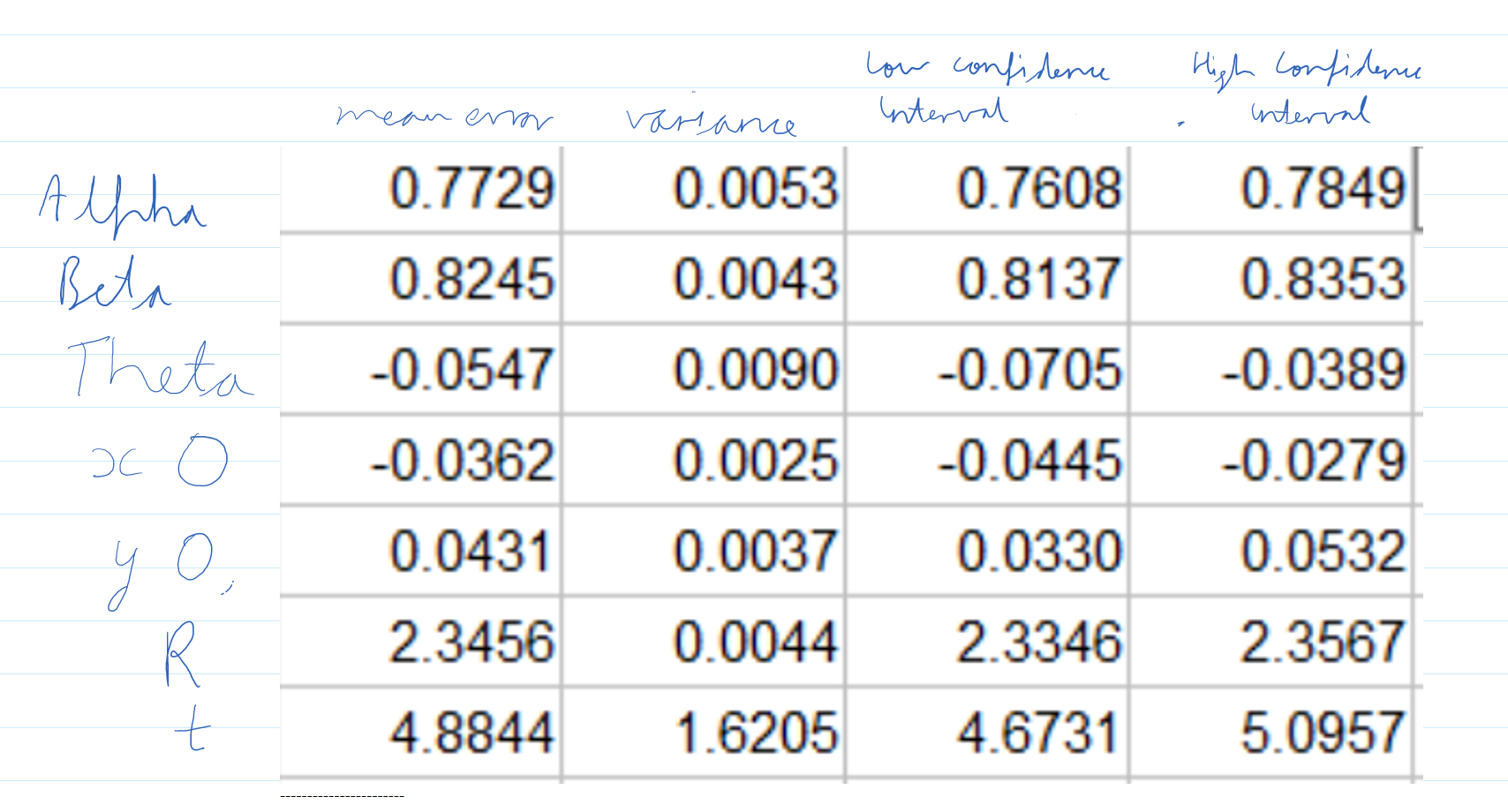




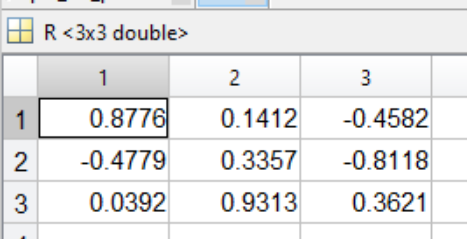


**Section 5: Analysis:**

**The following is the summazry of error in calibration:**



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

**R =** 

**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. However this is an anomaly since Intuitively, if error goes up in the input data, then error should go up in the output data.  The alpha estimate should not get better as the image noise gets worse.
* 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
* Order of sensitivity for mean error: x0,y0,theta,alpha, beta, R, t. This means for mean error is most for t and least for x0
* Order of sensitivity for variance in error: x0,y0,beta,R,alpha,theta,. This means for variance in error is most for t and least for x0. Beta, R amd alpha almost show same values.
* I conclude that t is worst affected when mean error and variance 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