Homework 8

ECE 661
Aziza Satkhozhina
asatkhoz@purdue.edu

Goals

The goal of this homework is to estimate intrinsic and extrinsic camera parameters using Zhang's calibration algorithm.

Detecting corners

Corners are detected using Hough transform. The steps in detecting the corners are the following:

1) First step is use Canny edge detector to find the edges. I used threshold value 0.7 in Canny detector.

2) Then, Hough Transform was applied. I used imbuilt Matlab function hough with RhoResolution parameter set to 0.5. The I found the highest 25 peaks using Threshold value 25. From the peaks, I constructed hough lines filling the gap between unconnected lines that are less than 150 pixels away from each other and with minimum length of 70.

3) However, since there are 20 squares on the calibration image, only 18 lines are needed out of which 10 are horizontal and 8 are vertical lines. To eliminate extra lines, I first classified each of 25 lines into horizontal or vertical line using their slope. Then for each class I left lines that have fewest intersections with other lines within that class since for each class we want lines that are almost parallel and do not intersect inside the image.

4) Once 18 lines are identified, their intersections were found. I used homogeneous coordinates to find the intersections.
$l_1 = x_1 \times x_2$, $l_2 = x_3 \times x_4$, and $x_{int} = l_1 \times l_2$, where x_1, x_2 and x_3, x_4 are points that are on lines 1 and 2 respectively.

5) Then, detected corners were labeled using the numbering scheme that I decided beforehand for consistency. The corners were labeled from top to bottom and left to right.

Calibration

After the corners were detected, the next step was to find the homography between world coordinates and image pixels. I measured corner coordinates using the rules where I decided that the top left corner has corrdinates of (0,0). One side of a square and a distance between the squares in calibration pattern were 25mm. I used these coordinates to establish Homographies between those coordinates and corner coordinates of each image in the dataset. So, I found H where $xIM = H \cdot xW$ for each image. H was found using nullspace of matrix A as in previous homeworks. The solution for nullspace was found to be an eigenvector that corresponded to the smallest eigenvalues after using SVD.

Zhang's algorithm was applied after homographies were found. Zhang's calibration algorithm aims to find intrinsic and extrinsic parameters of the camers which are represented by 3×3 matrix K and extrinsic paramterers R and t. The algorithm assumes that there is a calibration pattern on $Z = 0$ and multiple pictures of calibration pattern are taken from different viewpoints. It is based on the face that the pixtures of absolute conic are independent of the viewpoint, therefore it is given by $w = K^{-1}K$. We can get $\vec{h}_1^T w \vec{h}_1 = \vec{h}_2^T w \vec{h}_2$ and $\vec{h}_1^T w \vec{h}_2 = 0$, where w is the image of absolute conic and vectors $\vec{h}_i$ are columns of matrix H. We can define matrix w as a vector $b = (w_{11}, w_{12}, w_{22}, w_{13}, w_{23}, w_{33})$ and define $\vec{v}_{ij} = (h_{i1}h_{j1}; h_{i1}h_{j2} + h_{i2}h_{j1}; h_{i2}h_{j2}; h_{i3}h_{j1} + h_{i1}h_{j3}; h_{i3}h_{j2} + h_{i2}h_{j3}; h_{i3}h_{j3})$. We can derive $\vec{v}_{12}^T b = 0$ and $(\vec{v}_{11} - \vec{v}_{22})^T b = 0$. Since $\vec{v}$ consists of only values of matrix H, we can stack up the resulting equaions using n images from the dataset. b is the nullspace solution and can be found using SVD. Nullspace solution is the eigenvector that corresponds to the smallest eigenvalue after using SVD.

Even though we know found the image of the absolute conic, our goal is to find parameters of the camera. Instrinsic parameters make the following matrix $K = \begin{pmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}$. The parameters can be found with the following equations:

$$x_0 = \frac{w_{12} - w_{13} - w_{11}w_{23}}{w_{11}w_{22} - w12w_{12}}$$

$$\lambda = w_{33} - \frac{w_{13}^2 + x_0(w_{12}w_{13} - w_{11}w_{23}}{w_{11}}$$

$$\alpha_x = \sqrt{\frac{\lambda}{w_{11}}}$$

$$\alpha_y = \sqrt{\frac{\lambda w_{11}}{w_{11}w_{22} - w_{12}^2}}$$

$$s = -\frac{w_{12}\alpha_x^2\alpha_y}{\lambda}$$

$y_0 = \frac{sx_0}{\alpha_y} - \frac{w_{13}\alpha_x^2}{\lambda}$

Note that intrinsic parameters are the same for all images. Therefore they found once. However extrinsic parameters are different for each image depending on the viewpoint of the camera. Therefore they are calculated for each image separately.

$R = [\vec{r}_1 \vec{r}_2 \vec{r}_3]$ and $K^{-1}[\vec{h}_1 \vec{h}_2 \vec{h}_3] = [\vec{r}_1 \vec{r}_2 \vec{t}]$

From these equations we can get

$\vec{r}_1 = scale \cdot K^{-1}\vec{h}_1$

$\vec{r}_2 = scale \cdot K^{-1}\vec{h}_2$

$\vec{r}_3 = \vec{r}_1 \times \vec{r}_2$

$\vec{t} = scale \cdot K^{-1}\vec{h}_3$, where $scale = \frac{1}{||K^{-1}\vec{h}_1||}$

Refinement of calibration parameters

Since our corner detection does not give exact locations of the corners, we can use Levenberg-Marquadt optimization algorithm to further refine the results. I used Matlab's lsqnonlin function for LM. The cost function fot the optimization is the following: $cost = \sum_i \sum_j ||\vec{x}_{ij} - K[R_i t_i]\vec{x}_{M,j}||^2$, where $\vec{x}_{ij}$ is jth pixel of image i that was estimated using corner detection algorithm, $\vec{x}_{M,j}$ is world coordinate j , R_i, t_i are extrinsic parameters from image i, and K is intrinsic parameters of the camera. We stack parameters from all images together into $p = [K, R_1, t_1, R_2...]$ vector and use LM algorithm to refine these parameters.

One thing to note is that matrix R has 9 parameters, but rotation has only 3 DoF. Therefore, we use polar coordinates, and transform matrix R into Rodriguez representation.

$\vec{w} = \frac{\phi}{2sin\phi} \begin{pmatrix} r_{32} - r_{23} \\ r_{12} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}$ and $\phi = acos(\frac{trace(R)-1}{2})$.

To transform back from Rodriguez, we use

$R = I + \frac{sin(\phi)}{\phi} W_x + \frac{1-cos\phi}{\phi} W_x^2$, where $W_x = \begin{pmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{pmatrix}$

Conditioning Rotational matrix

We need to make sure that our rotational matrix R is orthogonal. Therefore we need to find R for given Q:

$min_R ||R - Q||_F^2$ where $||\cdot||_F$ is the Frobenius norm. The solution for a given problem is use SVD $[U, D, V] = svd(Q)$ and set R to UV^T.

Incorporating Radial Distortion

So far, we assumed that we have a pinhole camera. However, pinhole model breaks down for short focal-length cameras. Therefore, we need to compensate for the radial distortion that is introduced by using this model. In previous steps we predicted pixel coordinates $\hat{x} = K[R|t]xW$. We can compensate for radial distortion by using this as our predicted pixel:

$\hat{x}_{rad} = \hat{x} + (\hat{x} - x_0)[k_1 r^2 + k_2 r^4]$ and $\hat{y}_{rad} = \hat{y} + (\hat{y} - y_0)[k_1 r^2 + k_2 r^4]$,

where k_1 and k_2 are parameters for radial distortion and $r^2 = (\hat{x} - x_0)^2 + (\hat{y} - y_0)^2$.

Results

Intrinsic parameters for the given dataset, before LM:

$$K = \begin{pmatrix} 744.9667 & 0.5500 & 329.9118 \\ 0 & 745.3404 & 232.5847 \\ 0 & 0 & 1 \end{pmatrix}.$$

Intrinsic parameters for the given dataset, after LM, without radial distortion:

$$K = \begin{pmatrix} 904.9858 & 0.9448 & 332.7354 \\ 0 & 906.6334 & 237.1124 \\ 0 & 0 & 1 \end{pmatrix}.$$

Intrinsic parameters for the given dataset, after LM, with radial distortion:

$$K = \begin{pmatrix} 910.8388 & 0.8261 & 337.4781 \\ 0 & 913.0800 & 236.3815 \\ 0 & 0 & 1 \end{pmatrix}.$$

$k1 = -0.2046$ and $k2 = 0.6363$.

Intrinsic parameters for my dataset, after LM, with radial distortion:

$$K = \begin{pmatrix} 1359.1 & 2 & 659 \\ 0 & 1352.7 & 491.5 \\ 0 & 0 & 1 \end{pmatrix}.$$

$k1 = -0.0955$ and $k2 = -0.0694$.

Output images for the provided dataset

Output images for Created dataset

In the figure 15, you can notice that R is almost an identity matrix, which proves that extrinsic parameters are calculated correctly! The translation was mostly in Z direction. The measured distance was 52mm, and t-vector has 59.8mm.

Figure 1: Provided dataset: 1) Edge images. 2) Hough lines. 3) Detected corners.

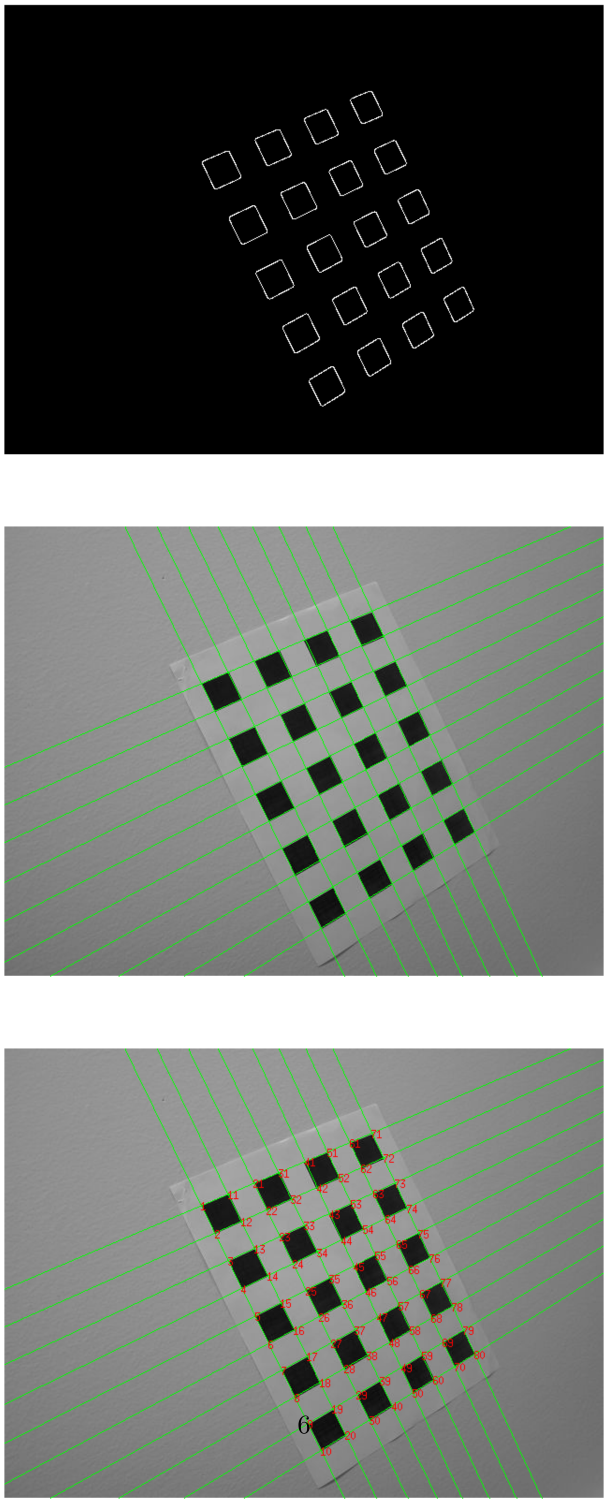
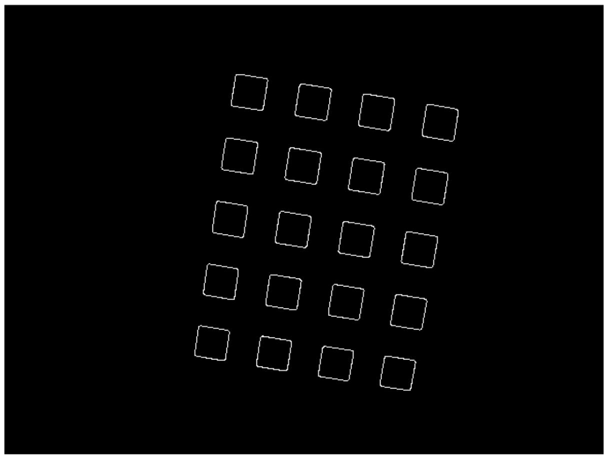
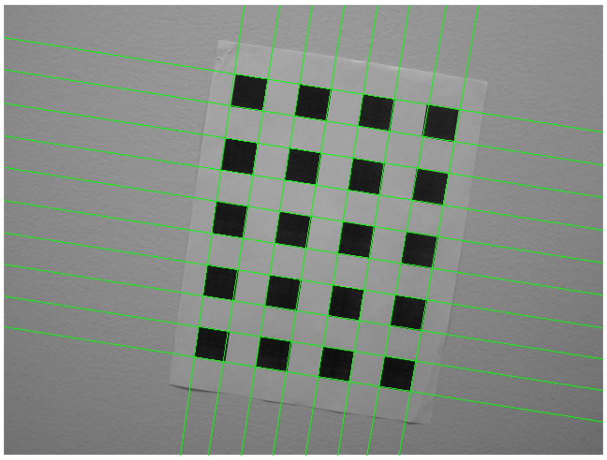


Figure 2: Provided dataset: 1) Edge images. 2) Hough lines. 3) Detected corners.





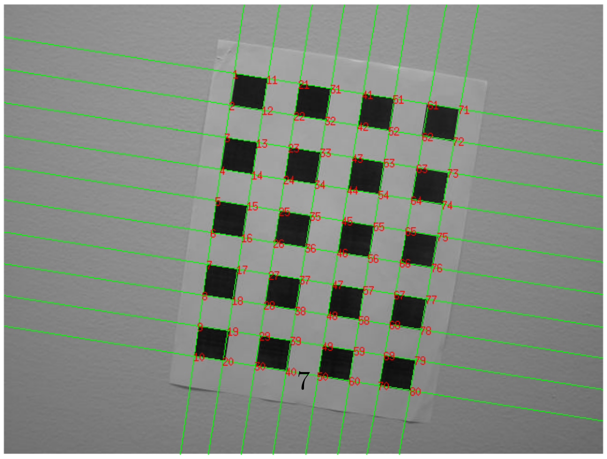


Figure 3: Provided dataset: 1) Edge images. 2) Hough lines. 3) Detected corners.

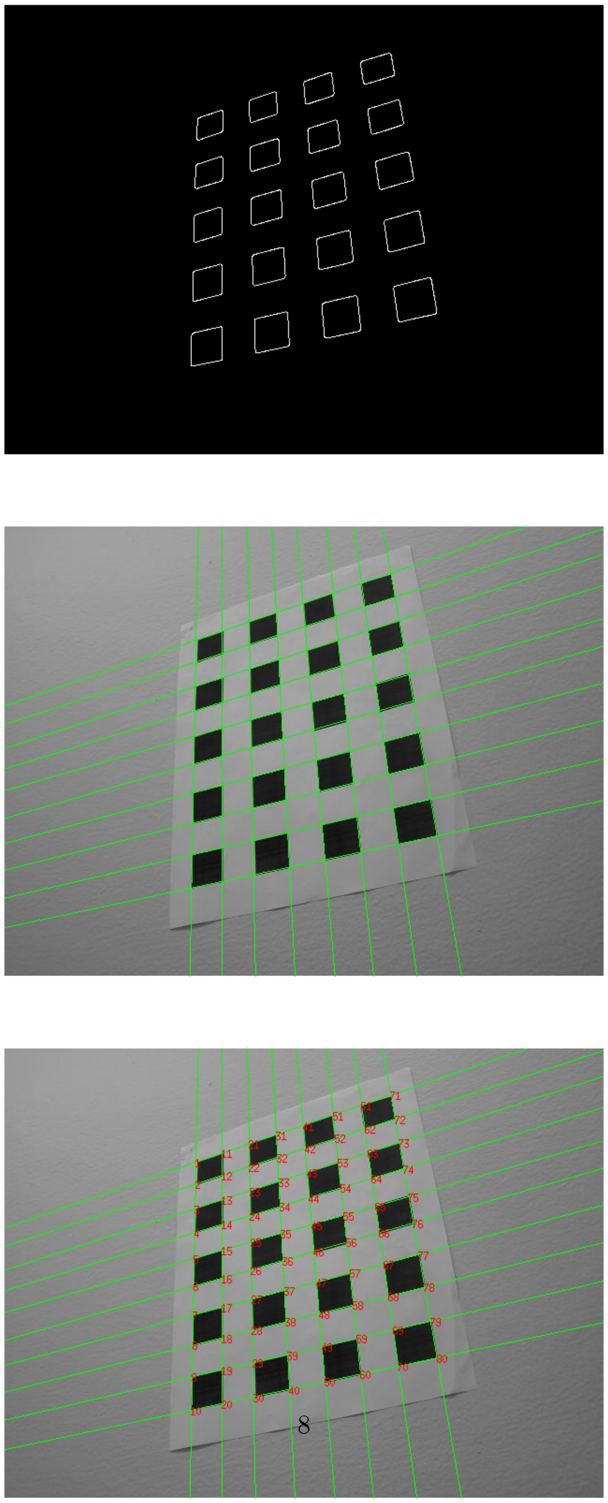


Figure 4: Provided dataset: 1) Edge images. 2) Hough lines. 3) Detected corners.

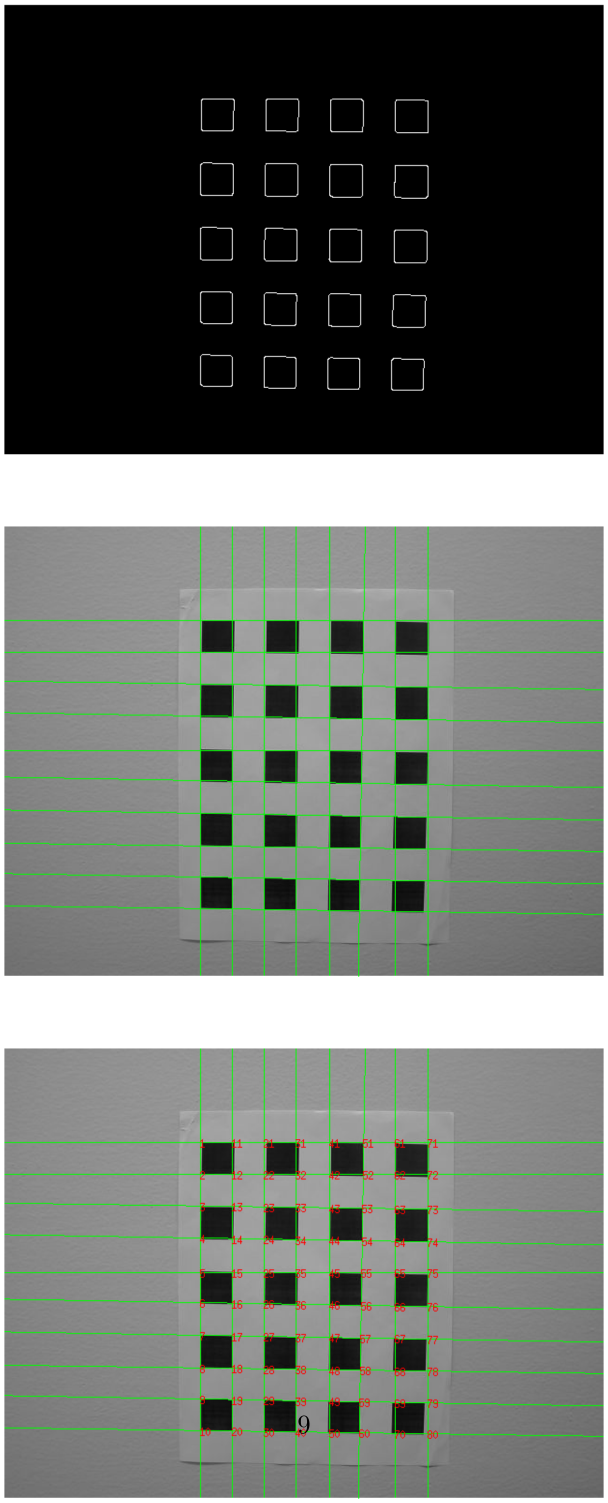


Figure 5: Extrinsic parameters for Image1: $[R|t] = \left(\begin{array}{ccc|c} 0.7987 & -0.1836 & 0.6564 & -58.4803 \\ 0.1954 & 0.9843 & 0.0329 & -124.1388 \\ -0.6530 & 0.0746 & 0.8141 & 687.6628 \end{array}\right)$.

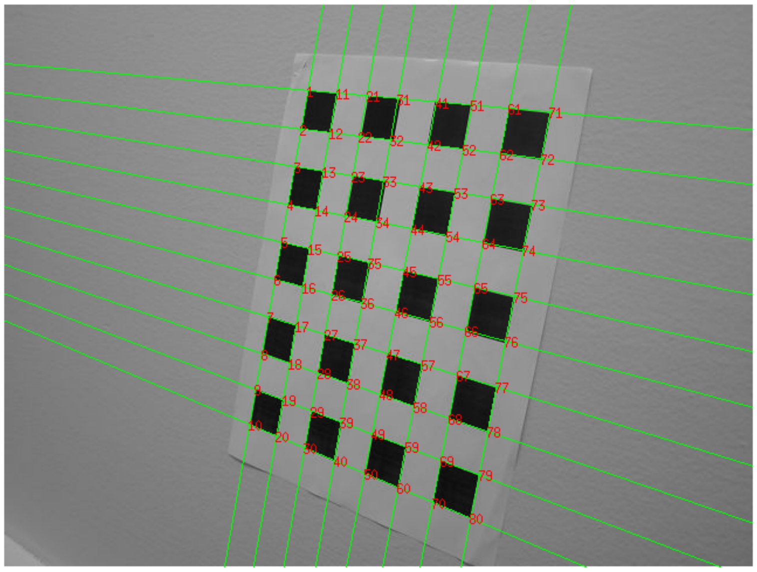


Figure 6: Extrinsic parameters for Image2: $[R|t] = \left(\begin{array}{ccc|c} 0.8161 & -0.0402 & 0.6620 & -49.4124 \\ -0.0006 & 0.9976 & -0.0791 & -97.0220 \\ 0.6632 & 0.0682 & 0.8140 & 628.1957 \end{array}\right)$.

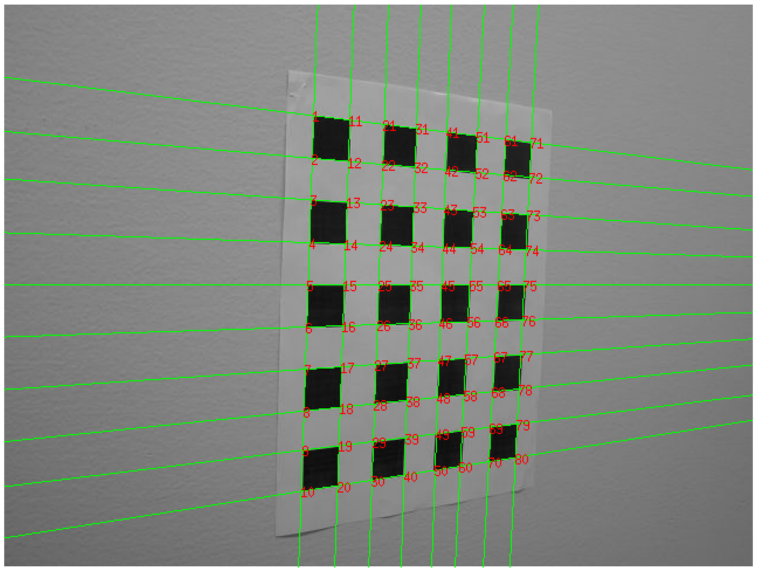


Figure 7: Extrinsic parameters for Image3: $[R|t] = \left(\begin{array}{ccc|c} 0.8793 & 0.1664 & -0.5626 & -75.6946 \\ -0.1980 & 0.9861 & -0.0453 & -84.8582 \\ 0.5523 & 0.1165 & 0.8887 & 640.6205 \end{array}\right)$.

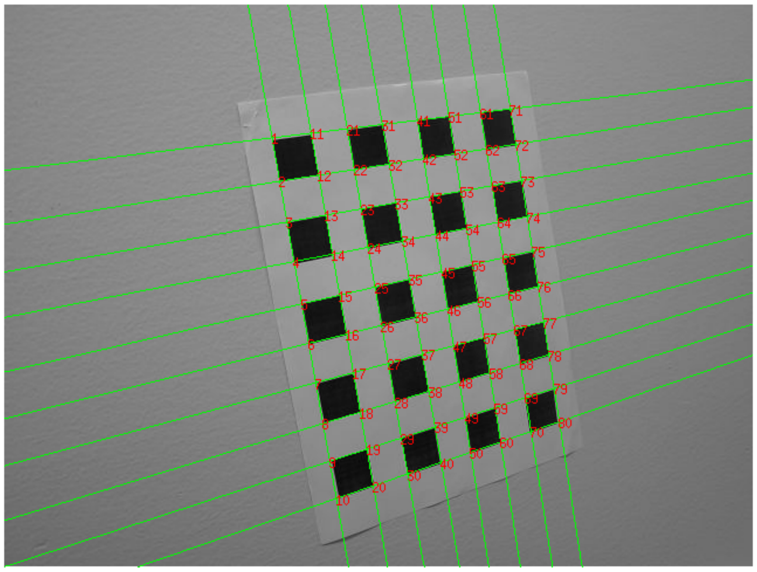


Figure 8: Extrinsic parameters for image 4: $[R|t] = \left(\begin{array}{ccc|c} 0.8419 & 0.4412 & -0.4584 & -98.4077 \\ -0.4242 & 0.9260 & 0.1255 & -53.5081 \\ 0.4742 & 0.0322 & 0.9142 & 716.8074 \end{array}\right)$.

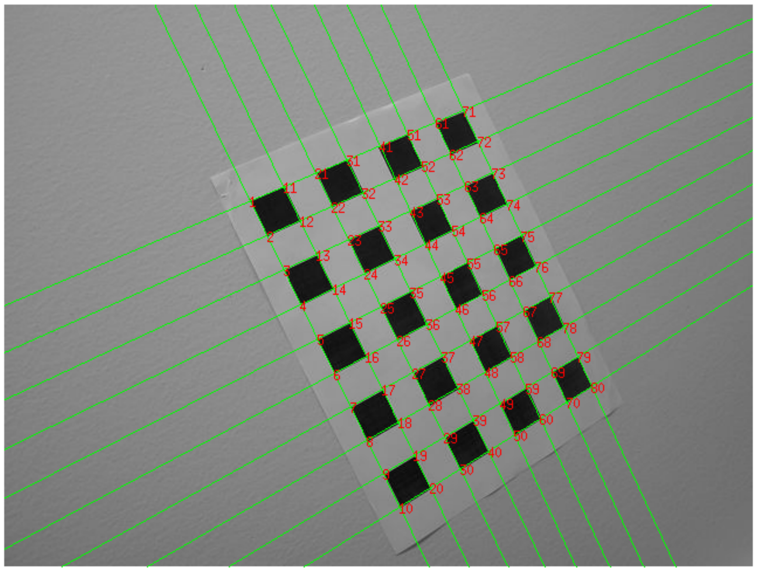
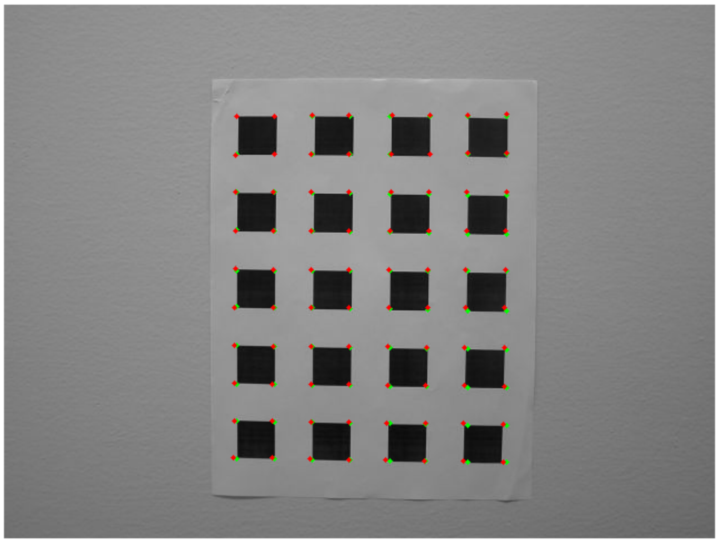
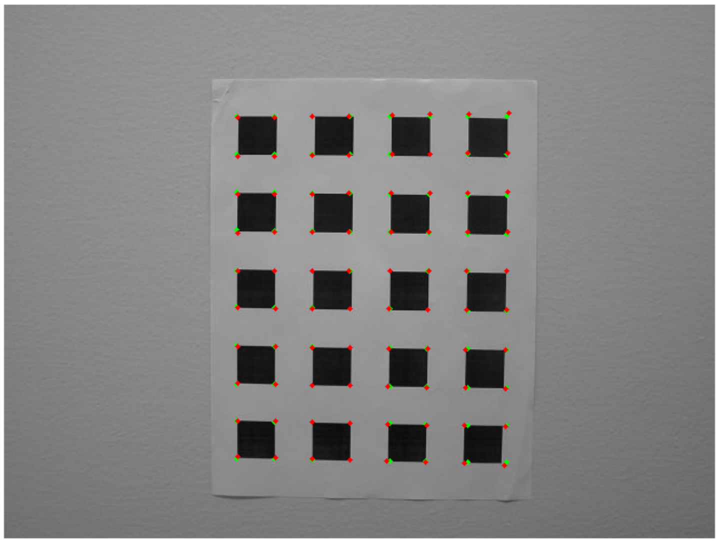


Figure 9: Provided dataset corner reprojection. Image1. Green points are the original corners and red points are projected corners.

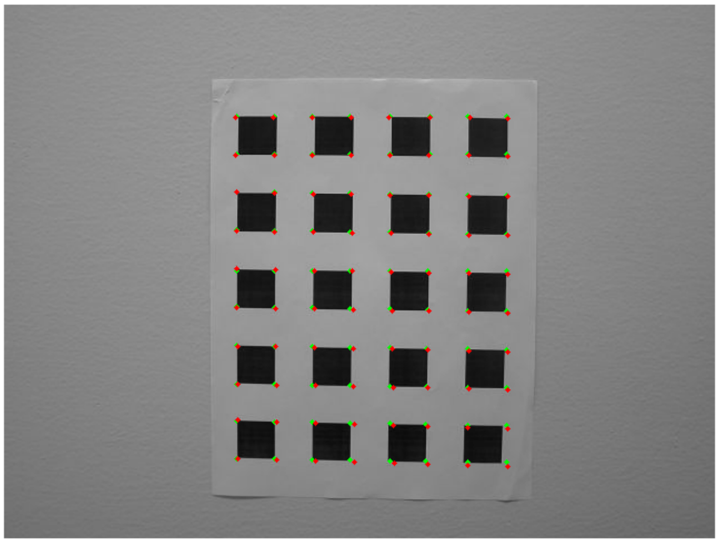


(a) Before LM mean error = 1.1464, variance of error = 0.7386

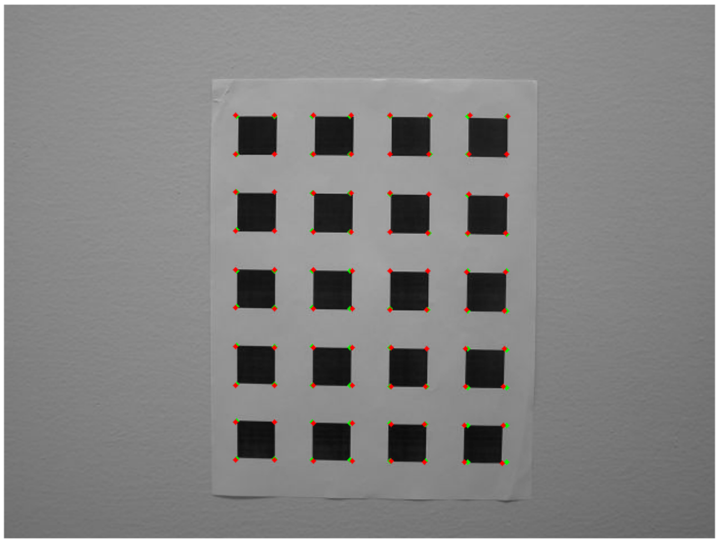


(b) After LM mean error = 0.8301, variance of error = 0.4798

Figure 10: Provided dataset corner reprojection. Image2. Green points are the original corners and red points are projected corners.

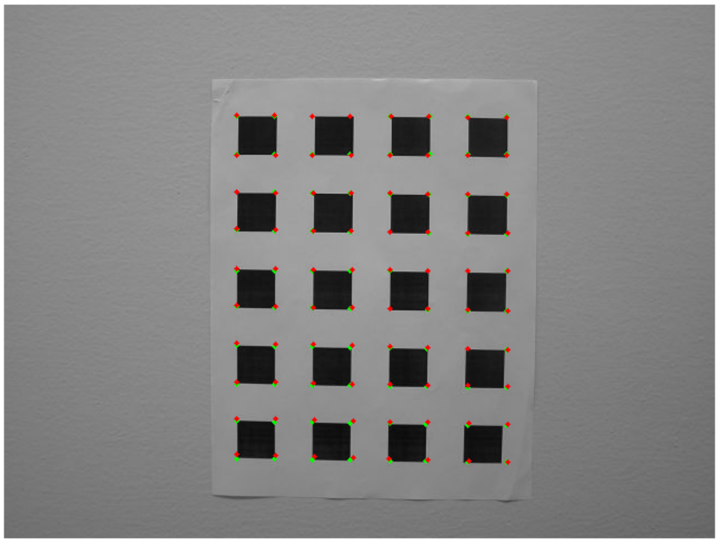


(a) Before LM mean error = 1.2354, variance of error = 0.7119



(b) After LM mean error = 0.7575, variance of error = 0.4302

Figure 11: Provided dataset corner reprojection. Image3. Green points are the original corners and red points are projected corners.


(a) Before LM mean error = 2.0047, variance of error = 1.9417

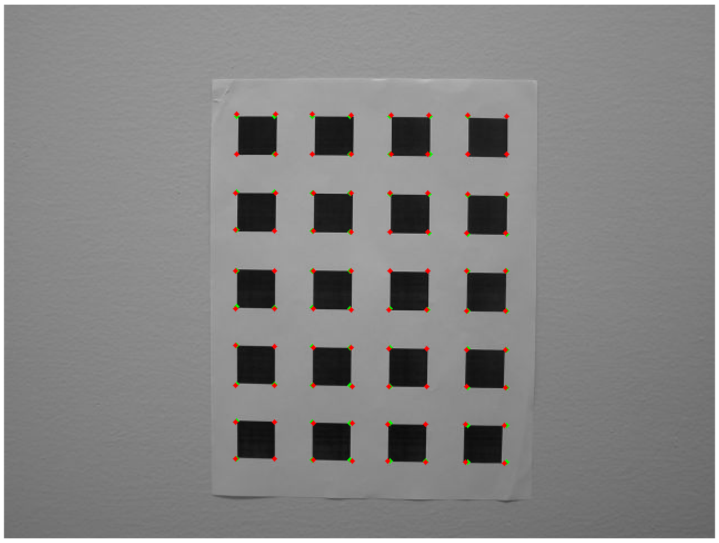
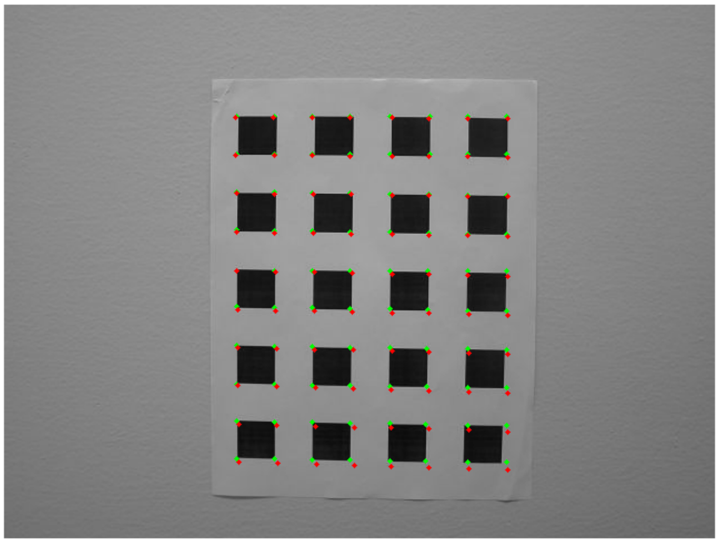

(b) After LM mean error = 0.7987, variance of error = 0.4699

Figure 12: Provided dataset corner reprojection. Image4. Green points are the original corners and red points are projected corners.


(a) Before LM mean error = 0.9370, variance of error = 0.4920

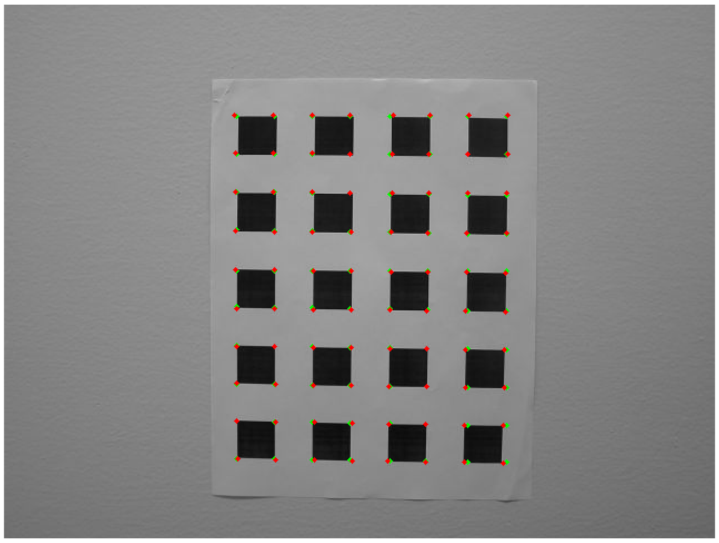
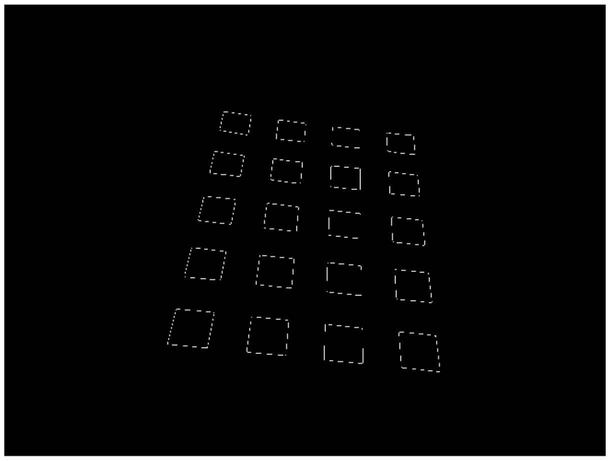

(b) After LM mean error = 0.6376, variance of error = 0.2379

Figure 13: My dataset: 1) Edge images. 2) Hough lines and Detected corners.



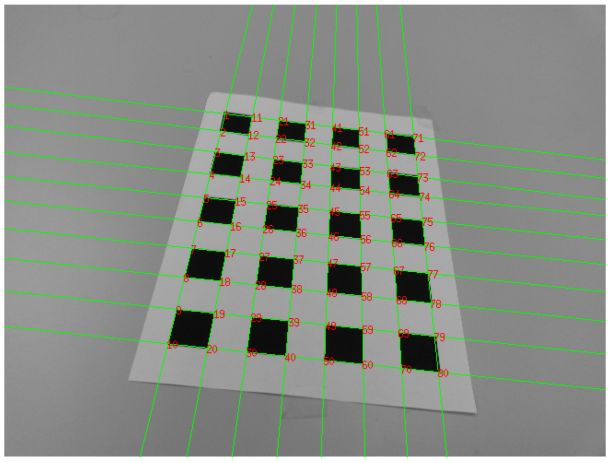
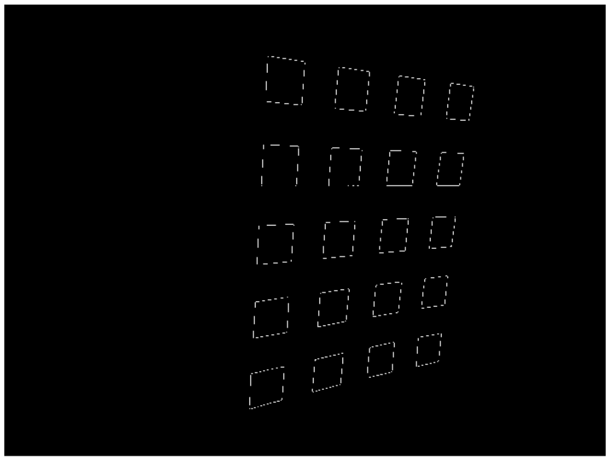


Figure 14: My dataset: 1) Edge images. 2) Hough lines and Detected corners.



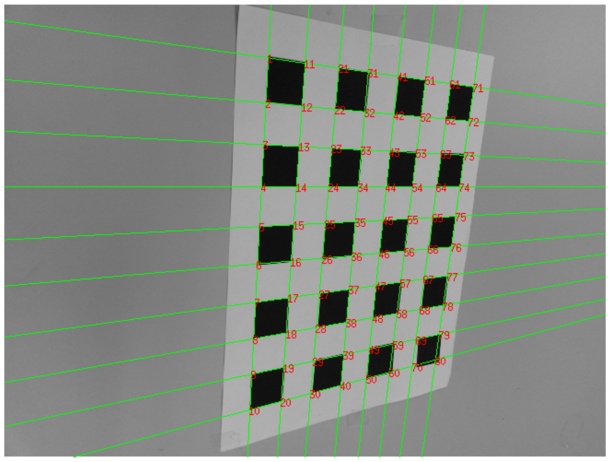


Figure 15: Extrinsic parameters for Image2: $[R|t] = \left(\begin{array}{lll|l} 1 & -0.007 & -0.0155 & -84.4837 \\ 0.0075 & 0.9967 & 0.1861 & -106.4158 \\ 0.0152 & -0.1861 & 0.9967 & 598.3001 \end{array}\right)$.

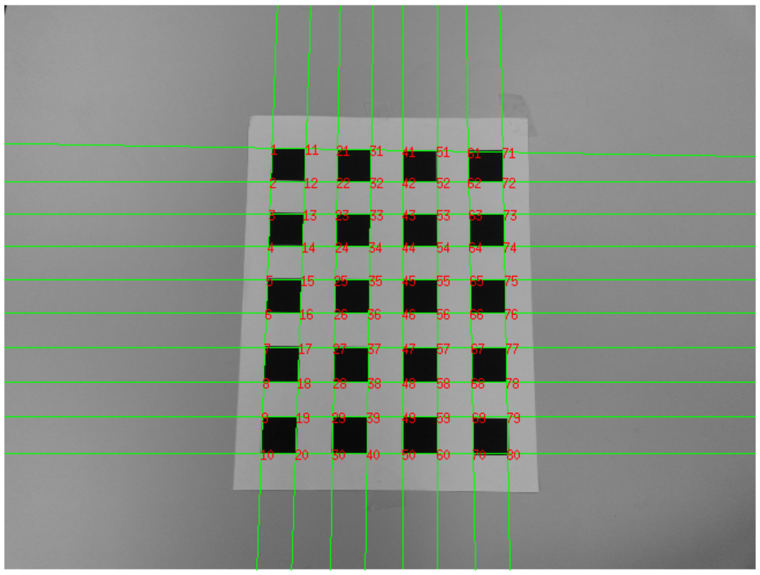


Figure 16: Extrinsic parameters for Image3: $[R|t] = \left(\begin{array}{ccc|c} 0.9989 & 0.0353 & -0.0790 & -95.7507 \\ -0.0295 & 0.9905 & 0.2583 & -101.6290 \\ 0.0813 & -0.2576 & 0.9898 & 526.3244 \end{array}\right)$.

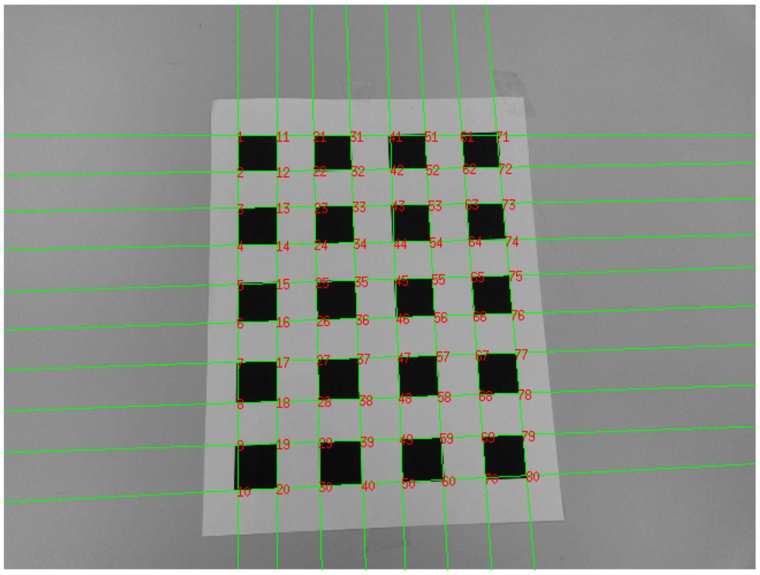


Figure 17: Extrinsic parameters for image 4: $[R|t] = \left(\begin{array}{lll|l} 0.9986 & -0.0106 & -0.0784 & -79.3633 \\ 0.0255 & 0.9592 & 0.4190 & -116.4901 \\ 0.0749 & -0.4196 & 0.9579 & 520.9872 \end{array}\right)$.

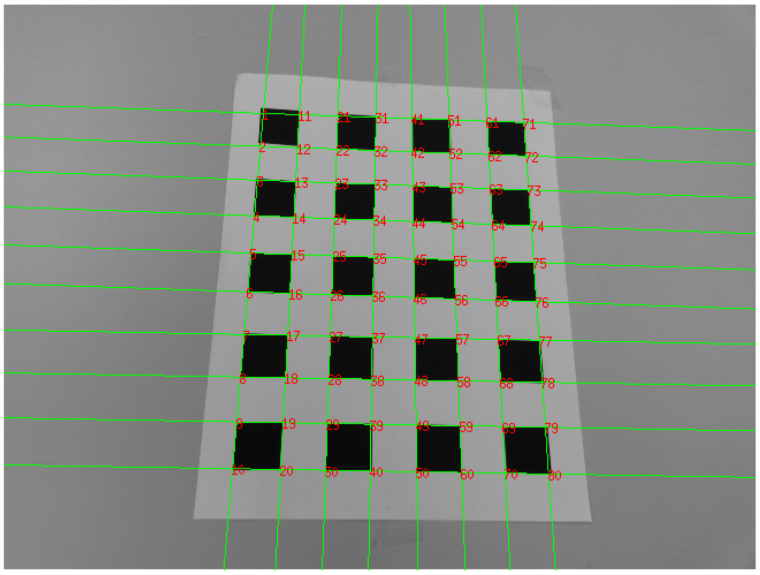
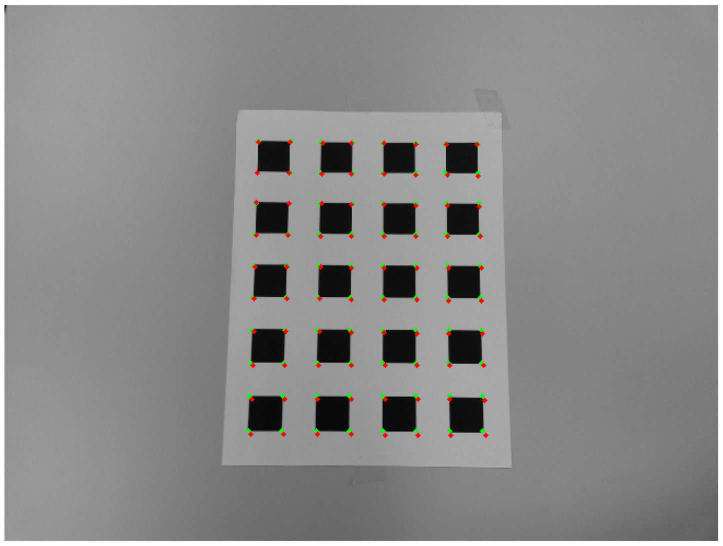
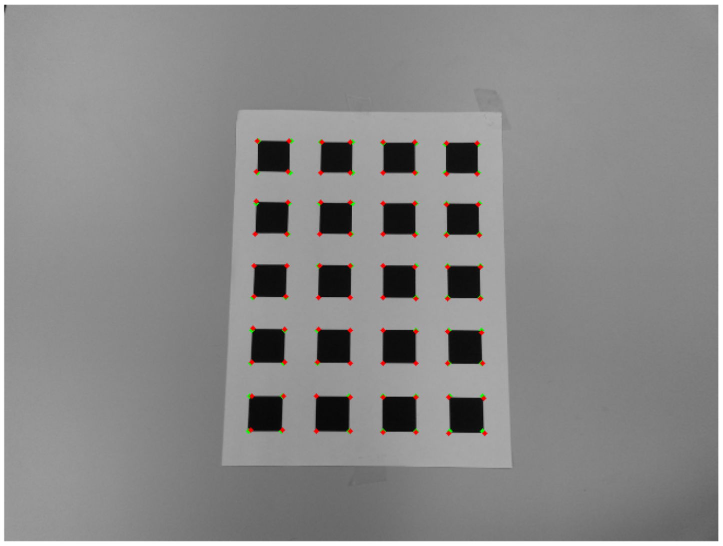


Figure 18: Provided dataset corner reprojection. Image1. Green points are the original corners and red points are projected corners.

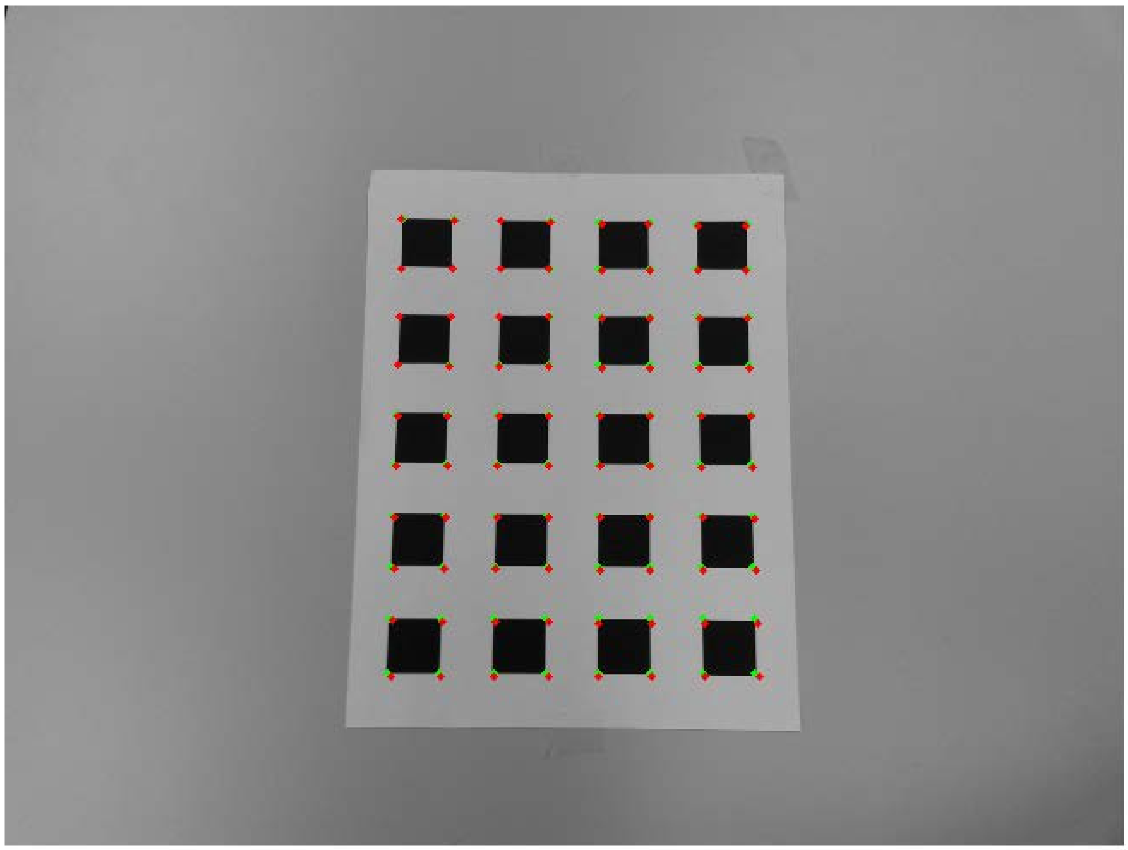


(a) Before LM mean error = 2.1984, variance of error = 3.2451



(b) After LM mean error = 1.296, variance of error = 1.0705

Figure 19: Provided dataset corner reprojection. Image1. Green points are the original corners and red points are projected corners.



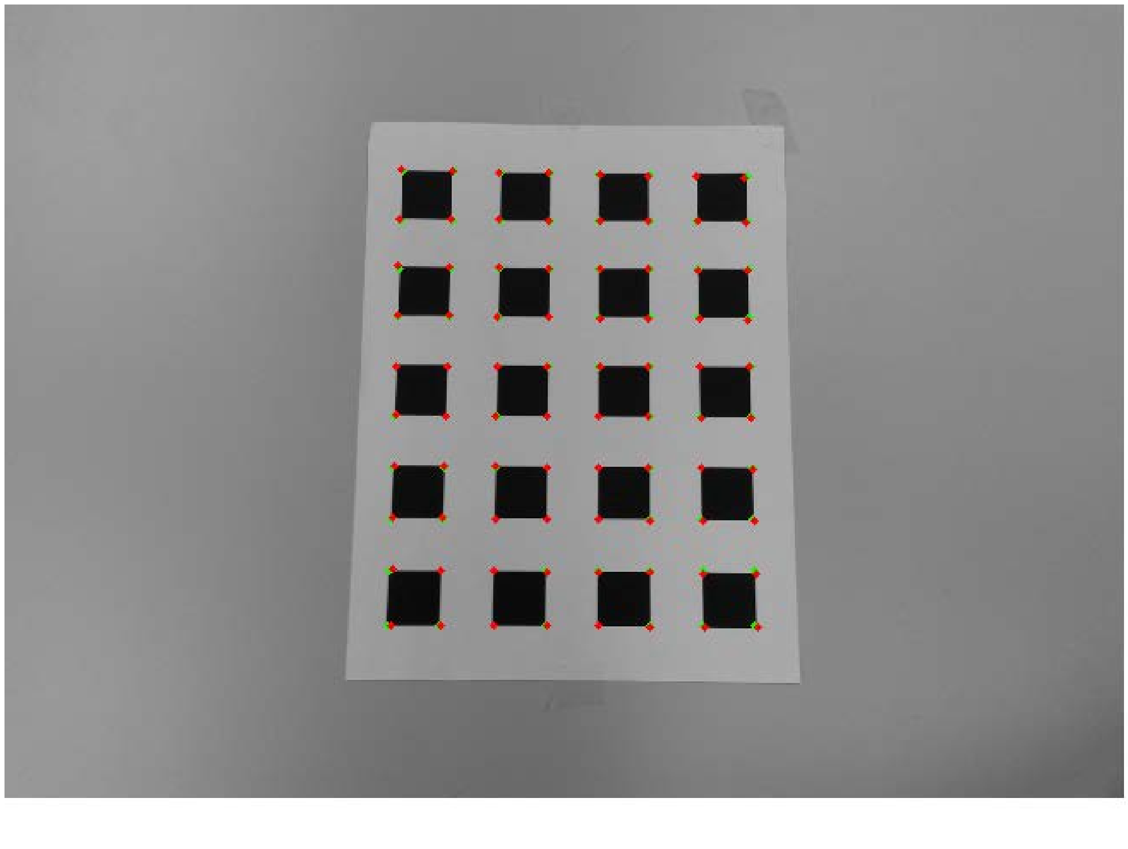
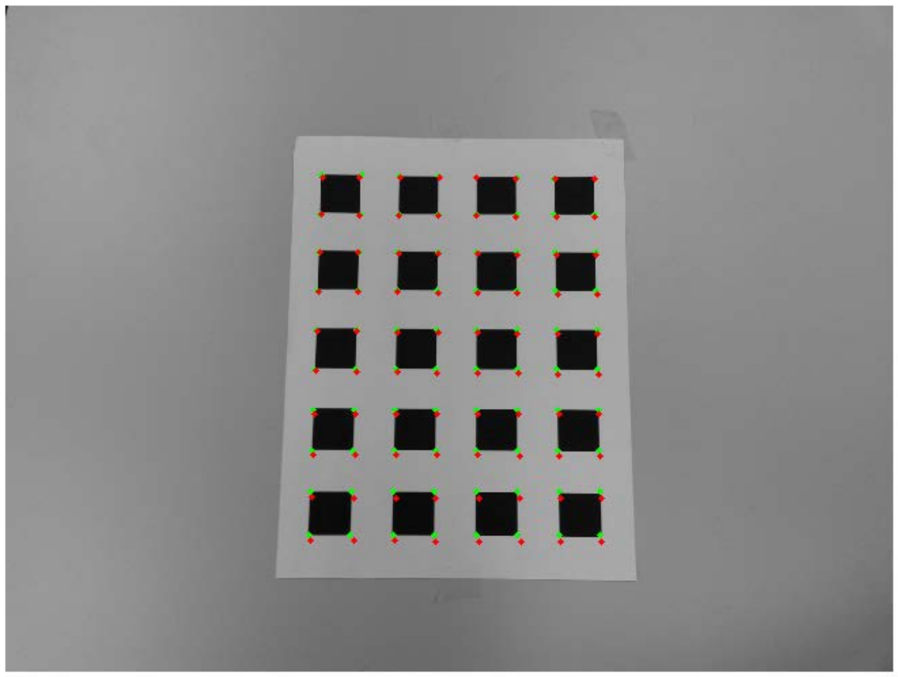
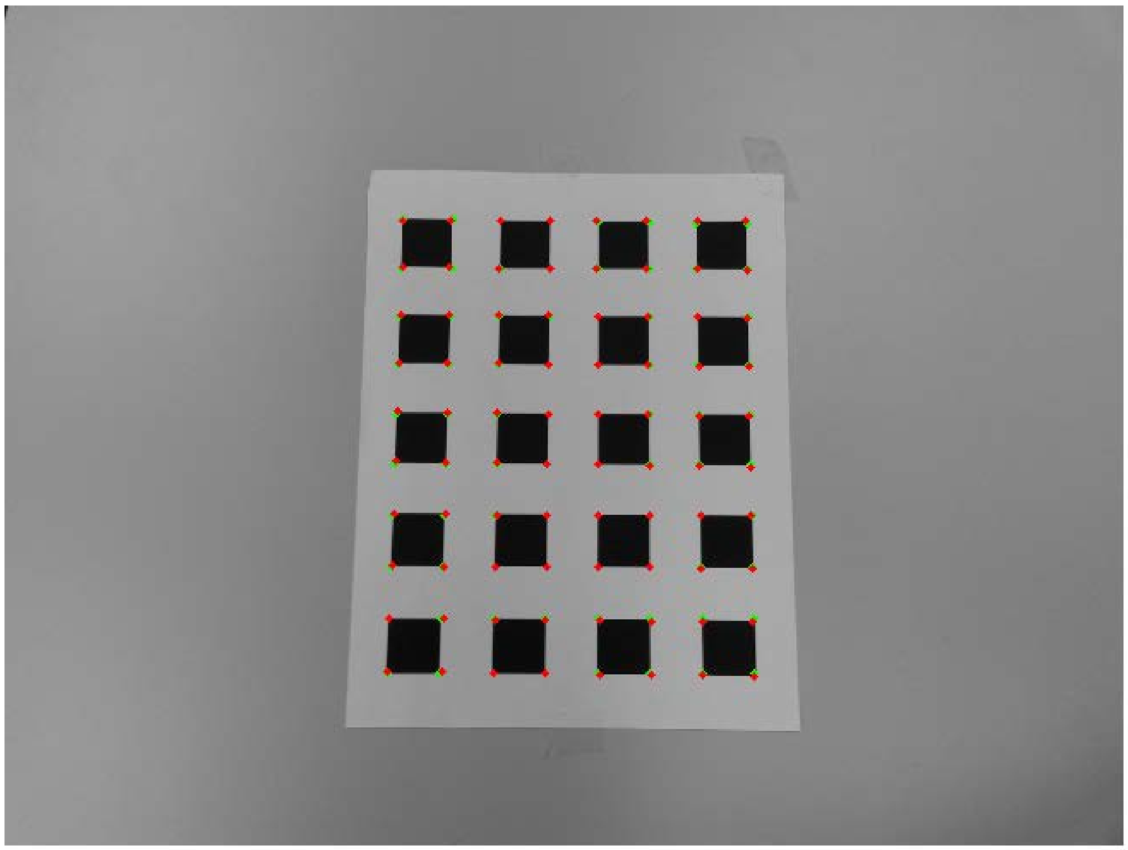


Figure 20: Provided dataset corner reprojection. Image1. Green points are the original corners and red points are projected corners.





```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% close all; warning off;
% %define and initialize world coordinates that were measured with the
ruler
xW = zeros(80,2);
for j  = 1:8
    for i = 1:10
        xW((j-1)*10+i,:) =  [(j-1)* 25 (i-1)*25];
    end
end

nimg = 20;
rad_dist = 1;

allH = []; V = []; xIM = [];
for k =1:20
    filename = strcat('Dataset2/Pic_',int2str(k),'.jpg');
    %calculate the coordinates of the corners in the image
    [imcoord] = get_corners(filename);
    xIM{k} = imcoord;
    %solve Ah = 0, and find h which is the homography
    A =
getA(xW(:,1),xW(:,2),double(imcoord(:,1)),double(imcoord(:,2)));
    [U,D,T] = svd(A);
    h = T(:,9);
    H = [h(1:3)'; h(4:6)'; h(7:9)'];
    allH{k} = H;
    %store V matrix for calculating the intrinsic parameters
    v12 = getv(H,1,2);
    v11 = getv(H,1,1);
    v22 = getv(H,2,2);
    V = [V
        v12
        (v11-v22)];
end

[U,D,T] = svd(V);
b = T(:,6); %B11 B12 B22 B13 B23 B33

%intrinsic parameters
y0 = (b(2)*b(4)-b(1)*b(5))/(b(1)*b(3)-b(2)^2);
lambda = b(6)-(b(4)^2+y0*(b(2)*b(4)-b(1)*b(5)))/b(1);
alphax = sqrt(lambda/b(1));
alphay = sqrt(lambda*b(1)/(b(1)*b(3)-b(2)^2));
s = -b(2)*alphax^2*alphay/lambda;
x0 = s*y0/alphay-b(4)*alphax^2/lambda;

%extrinsic parameters
K = [alphax s x0; 0 alphay y0; 0 0 1];
p = zeros(1,5+6*nimg);
p(1:5) = [alphax s x0 alphay y0];
if(rad_dist)
```

```
    p = zeros(1,7+6*nimg);
    p(1:5) = [alphax s x0 alphay y0];
    p(6:7) = [0 0];
    cnt = 7;
else
    p = zeros(1,5+6*nimg);
    p(1:5) = [alphax s x0 alphay y0];
    cnt = 5;
end

ydata=[];
K_inv = inv(K);
R_beforeLM = [];
R_afterLM = [];
t_beforeLM = [];
t_afterLM = [];
for k = 1:nimg
    H = allH{k};
    t = K_inv*H(:,3);
    mag = norm(K_inv*H(:,1));
    if(t(3)<0)
        mag = -mag;
    end
    r1 = K_inv*H(:,1)/mag;
    r2 = K_inv*H(:,2)/mag;
    r3 = cross(r1,r2);
    R = [r1 r2 r3];
    t = t/mag;
    [U,D,V] = svd(R);
    R = U*V';
    R_beforeLM{k}=R;
    t_beforeLM{k}=t;

    phi = acos((trace(R)-1)/2);
    w = phi/(2*sin(phi))*([R(3,2)-R(2,3) R(1,3)-R(3,1) R(2,1)-
R(1,2)])';
    p(cnt+1:cnt+3) = w;
    p(cnt+4:cnt+6) = t;
    cnt = cnt + 6;
    y=xIM{k};
    y=y';
    ydata=[ydata y(:)'];
end
x = xW';
xdata = x(:)';
options = optimoptions('lsqcurvefit','Algorithm','levenberg-
marquardt');
p1 = lsqnonlin(@myfun1,p,[],[],options,xdata,ydata,rad_dist,nimg);

alphax = p1(1);
s = p1(2);
x0 = p1(3);
```

```
alphay = p1(4);
y0 = p1(5);
K1 = [alphax s x0; 0 alphay y0; 0 0 1];

if(rad_dist)
    k1 = p1(6);
    k2 = p1(7);
    cnt = 7;
else
    cnt = 5;
end

for k = 1:nimg
    w = p1(cnt+1:cnt+3);
    t_afterLM{k} = p1(cnt+4:cnt+6)';
    cnt = cnt + 6;
    wx = [0 -w(3) w(2); w(3) 0 -w(1); -w(2) w(1) 0];
    phi = norm(w);
    R_afterLM{k} = eye(3)+sin(phi)/phi*wx + (1-cos(phi))/phi*wx^2;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function err = myfun1(p,xW,xIM,rad_dist,nimg)

alphax = p(1);
s = p(2);
x0 = p(3);
alphay = p(4);
y0 = p(5);
K = [alphax s x0; 0 alphay y0; 0 0 1];

if(rad_dist == 1)
    k1 = p(6);
    k2 = p(7);
    K1 = [alphax 0 x0; 0 alphay y0; 0 0 1];
    cnt = 7;
else
    cnt = 5;
end

xproj = zeros(1,nimg*160);
n1=1;

for k = 1:nimg
    w = p(cnt+1:cnt+3);
    t = p(cnt+4:cnt+6)';
    cnt = cnt + 6;
    wx = [0 -w(3) w(2); w(3) 0 -w(1); -w(2) w(1) 0];
    phi = norm(w);
    R = eye(3)+sin(phi)/phi*wx + (1-cos(phi))/phi*wx^2;
        n2=1;
    for i = 1:80
        x = K*[R t]*[xW(n2:n2+1) 0 1]';
        xproj(n1:n1+1) = [x(1)/x(3) x(2)/x(3)];
        if(rad_dist == 1)
            xp = [xproj(n1:n1+1)  1];
            xworld = inv(K1)*xp';
            r2 = xworld(1)^2 + xworld(2)^2;
            xp1  = xworld(1) + xworld(1)*(k1*r2+k2*r2^2);
            xp2  = xworld(2) + xworld(2)*(k1*r2+k2*r2^2);
            x =  K1*[xp1 xp2 1]';
            xproj(n1:n1+1) = [x(1)/x(3) x(2)/x(3)];
        end
        n1 = n1+2;
        n2 = n2+2;
    end
end

err = xIM - xproj;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x ] = intersect(line1, line2)

A = [line1.point1 1];
B = [line1.point2 1];

l1 = cross(A,B);

A = [line2.point1 1];
B = [line2.point2 1];

l2 = cross(A,B);

x = cross(l1,l2);

x = double([x(1)/x(3) x(2)/x(3)]);

%use homogenous coordinates
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [A] = getA(xW,yW,xIM,yIM)

A = [];
for i = 1:length(xW)
     B = [xW(i)   yW(i)   1   0       0       0   -xW(i)*xIM(i)   -
yW(i)*xIM(i) -xIM(i);
    0      0      0   xW(i)   yW(i)   1   -xW(i)*yIM(i)   -
yW(i)*yIM(i) -yIM(i)];

   A = [A; B];

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function v = getv(H,i,j)

v = [H(1,i)*H(1,j), H(1,i)*H(2,j)+H(2,i)*H(1,j), H(2,i)*H(2,j),
H(3,i)*H(1,j)+H(1,i)*H(3,j) ,H(3,i)*H(2,j)+H(2,i)*H(3,j),
H(3,i)*H(3,j)];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [] = reproject(R,t,K,xIM,k)
%fixed image is image 11 for provided dataset

filename = strcat('Dataset2/Pic_',int2str(2),'.jpg');
img = rgb2gray(imread(filename));
img = imresize(img,0.4);
r = R{2};
Pfixed = K*[r(:,1:2) t{2}]; %4x3
xtrue = xIM{2};

%K is fixed
r = R{k};
P = K*[r(:,1:2) t{k}];
x0 = K(1,3);
y0 = K(2,3);

xim = xIM{k};
xim = [xim ones(size(xim,1),1)];
xyz = inv(P)*xim';
xest = Pfixed*xyz;
xest = xest';
figure
imshow(img)
for i  = 1:80
    xest(i,:) = xest(i,:) / xest(i,3);
    hold on
    plot(uint64(xtrue(i,1)),uint64(xtrue(i,2)),'g.','MarkerSize',12);
     hold on
    plot(uint64(xest(i,1)),uint64(xest(i,2)),'r.','MarkerSize',12);
end
xest = xest(:,1:2);
hold off
err = abs(xtrue(:)-xest(:));
mean(err)
var(err)

%**********************************************
% if(rad_dist)
%     K1 = K; K1(1,2) = 0;
%     for i = 1:80
%          xp = [xest(i,:)  1];
%          xworld = inv(K1)*xp';
%          r2 = xworld(1)^2 + xworld(2)^2;
%           xp1  = xworld(1) + xworld(1)*(k1*r2+k2*r2^2);
%           xp2  = xworld(2) + xworld(2)*(k1*r2+k2*r2^2);
%           x =  K1*[xp1 xp2 1]';
%           xest(i,:) = [x(1)/x(3) x(2)/x(3)];
%      end
% end
%
% figure
% imshow(img)
% for i  = 1:80
%      hold on
%      plot(uint64(xest(i,1)),uint64(xest(i,2)),'r.','MarkerSize',15);
% end
% xest = xest(:,1:2);
% hold off
% err = abs(xtrue(:)-xest(:));
% mean(err)
% var(err)
% %*******************************************8*
%
%



end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [corner] = get_corners(filename)

gr_truth = imresize(imread(filename),0.4);

gr_truth_gray = rgb2gray(gr_truth);
gr_truth_edge = edge(gr_truth_gray,'canny',0.7);%for provided dataset
0.7
figure
imshow(gr_truth_edge)

[H, T, R] = hough(gr_truth_edge,'RhoResolution',1); %for provided
dataset 0.5

P = houghpeaks(H,25,'Threshold',10); %for provided dataset 25 and 15
lines = houghlines(gr_truth_edge,T,R,P,'FillGap',350,'MinLength',100);
%for provided dataset 150 and 70

% figure, imshow(gr_truth_gray), hold on
line_param = zeros(length(lines),2); %slope, y-intersect,
hor = []; ver = [];
% figure
% imshow(gr_truth_gray)
for k = 1:length(lines)
   xy = [lines(k).point1; lines(k).point2];
   %find the equation of the line y = mx + b
   %find slope m
   line_param(k,1) = (xy(1,2)-xy(2,2))/(xy(1,1)-xy(2,1));
%    plot_line(lines,k,size(gr_truth_edge));
   if(abs(line_param(k,1))>1.5)
       ver = [ver k];
   else
       hor = [hor k];
   end

   if(abs(line_param(k,1)) == inf)
       line_param(k,2) = inf;
   else
       line_param(k,2) = xy(1,2) - line_param(k,1)*xy(1,1);
   end
end

corner = [];
for i = 1:length(lines)
    n_c{i} = [];
end
%*********************************************************************
***
%get rid of extra lines
lines_hor = lines(hor);
ehor = zeros(1,length(hor));
for i= 1:length(lines_hor)
    for j = i+1:length(lines_hor)
        [pt]= intersect(lines_hor(i), lines_hor(j));
        if(pt(1)>1 && pt(1)<size(gr_truth,2) && pt(2)>1 &&
pt(2)<size(gr_truth,1))
            ehor(i) =ehor(i)+ 1;
            ehor(j) = ehor(j)+1;
        end
    end
end
lines_ver = lines(ver);
ever = zeros(1,length(ver));
for i= 1:length(lines_ver)
    for j = i+1:length(lines_ver)
        [pt]= intersect(lines_ver(i), lines_ver(j));
        if(pt(1)>1 && pt(1)<size(gr_truth,2) && pt(2)>1 &&
pt(2)<size(gr_truth,1))
            ever(i) = ever(i) +1;
            ever(j) = ever(j) +1;
        end
    end
end

[ever ind1] = sort(ever,'ascend');
[ever ind2] = sort(ehor,'ascend');
lines = lines([hor(ind2(1:10)) ver(ind1(1:8))]);
```

```
%*********************************************************************
***
%plot the lines
figure
imshow(gr_truth_gray)
for k = 1:length(lines)
   xy = [lines(k).point1; lines(k).point2];
   %find the equation of the line y = mx + b
   %find slope m
   line_param(k,1) = (xy(1,2)-xy(2,2))/(xy(1,1)-xy(2,1));
   if(abs(line_param(k,1)) == inf)
       line_param(k,2) = inf;
       hold on
       y = 1:size(gr_truth,1);
       x = xy(1,1)*ones(1,length(y));
       plot(x,y,'Color','green')
   else
       line_param(k,2) = xy(1,2) - line_param(k,1)*xy(1,1);
       f = @(x)  line_param(k,1)*x + line_param(k,2);
       x = 1:size(gr_truth,2);
       y =  uint64(f(x));
       hold on
       plot(x,y,'Color','green');
   end
end


%*********************************************************************
***

%find the corners
for i= 1:length(lines)
    for j = i+1:length(lines)
        [pt]= intersect(lines(i), lines(j));
        if(pt(1)>1 && pt(1)<size(gr_truth,2) && pt(2)>1 &&
pt(2)<size(gr_truth,1))
            corner = [corner; pt ];
%             hold on
%             plot(pt(1),pt(2),'r*')
            n_c{i} = [n_c{i} size(corner,1)];
            n_c{j} = [n_c{j} size(corner,1)];
        end
    end
end

%make sure you have 80 corners
if(size(corner,1)~=80)
    disp(length(lines))
    disp(size(corner,1));
    fprintf('Error with the number of corners\n');
end
```

```
%label the corners same way as I did the
%if n_l(i) is 10 then line i is vertical , if 8 then it's horizontal
%neded to sort the linesâ€" Ð                                    Ñ, â€" Ð³Ð¾Ð²Ð¾Ñ
Ð¿Ñ€ÐµÐ´Ð¿Ð¾Ð»Ð°Ð³Ð°ÑŽ, Ñ‡Ñ‚Ð¾ Ñ‚Ð¾Ñ‚ ÐºÐ¾Ñ‚Ð¾Ñ€Ñ‹Ð¹ Ñ        Ð²Ð¾ÐµÐ¹
Ð³Ð¾Ð»Ð¾Ð²Ð¾Ð¹ Ñ€Ð°Ð·Ð±Ð¸Ð» Ð»Ð¾Ð±Ð¾Ð²Ð¾Ðµ Ñ                Ñ‚ÐµÐ°Ð»Ð¾, Ð
Ð¿Ð¾Ð»ÑƒÑ‡Ð¸Ñ‚ÑŒ Ð¾ÐºÐ¾Ð»Ð¾ 5 Ð»ÐµÑ‚ Ð·Ð° Ð¿Ð¾Ð²Ñ€ÐµÐ¶Ð´ÐµÐ½Ð¸Ðµ
Ñ‡ÑƒÐ¶ÐµÐ³Ð¾ Ð¸Ð¼ÑƒÑ‰ÐµÑ                             Ñ‚Ð²Ð° Ð¸ Ð¿Ð¾Ð¿Ñ‹Ñ‚Ð°Ñƒ
Ð²Ð¾Ñ€Ð¾Ð²Ñ                                                  Ñ‚Ð²Ð°... Ð
Ð² ÐºÑƒÑ                                       Ñ‚Ñ‹, Ð¼Ð¾Ð¶ÐµÑ‚, Ð¸ 8 Ñ
Ð¿Ð¾Ð¿Ñ‹Ñ‚ÐºÑƒ ÑƒÐ±ÐµÐ¶Ð°Ñ‚ÑŒ Ñ                                 Ð¼ÐµÑÑ‚Ð°

hor = [];
ver = [];

for i = 1:length(lines)
    if(length(n_c{i}) ==8)
        hor = [hor i];
    else
        ver = [ver i];
    end
end

xs = zeros(length(ver),1); %for each vertical line sort
for i = 1:length(ver)
    %sort according to smallest x
    ind = n_c{ver(i)}; %these are corners that are on that line
    xs(i) = min(corner(ind,1)); %this is the smallest y for that
vertical line
end

[d ind] = sort(xs,'ascend'); %sort vertical lines according to the
smalles x
ver = ver(ind); %vertical lines are sorted

labels = zeros(80,1);
cnt = 0;
ys = zeros(10,1); %for each vertical line sort
for  i = 1:length(ver)
    ind = n_c{ver(i)}; %these are corners that are on that line
    ys = corner(ind,2);
    [d sind] = sort(ys,'ascend');
    for j = 1:length(sind) %1 to 10
        cnt =cnt + 1;
        labels(cnt) = ind(sind(j));
    end
end

corner = corner(labels,:);
```

```
%
%*********************************************************************
***
% %plot the lines
% figure
% imshow(gr_truth_gray)
% for k = 1:length(lines)
%    xy = [lines(k).point1; lines(k).point2];
%    %find the equation of the line y = mx + b
%    %find slope m
%    line_param(k,1) = (xy(1,2)-xy(2,2))/(xy(1,1)-xy(2,1));
%    if(abs(line_param(k,1)) == inf)
%        line_param(k,2) = inf;
%        hold on
%        y = 1:size(gr_truth,1);
%        x = xy(1,1)*ones(1,length(y));
%        plot(x,y,'Color','green')
%    else
%        line_param(k,2) = xy(1,2) - line_param(k,1)*xy(1,1);
%        f = @(x)  line_param(k,1)*x + line_param(k,2);
%        x = 1:size(gr_truth,2);
%        y =  uint64(f(x));
%        hold on
%        plot(x,y,'Color','green');
%    end
% end


%*********************************************************************
***

for i = 1:length(labels)
    hold on
    text(corner(i,1),corner(i,2),int2str(i),'Color','r');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```