Project 1

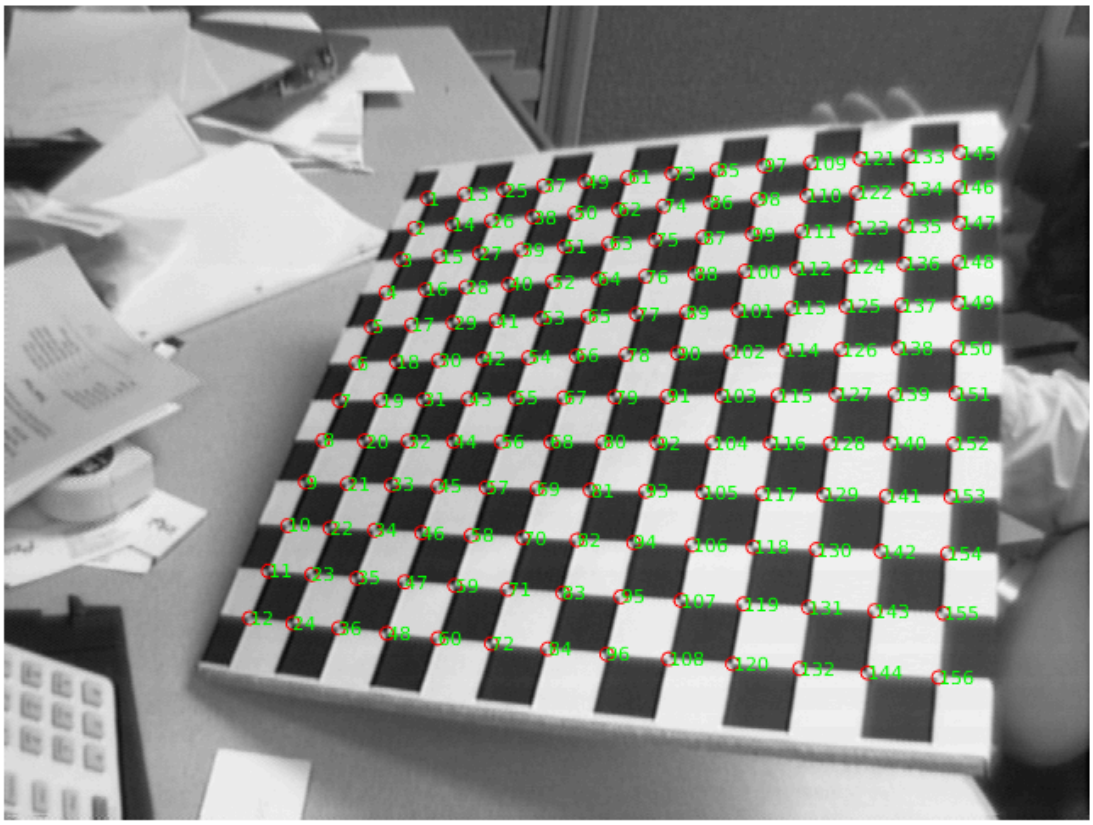
This is project 1 for Computer Vision and Pattern Recognition exam.

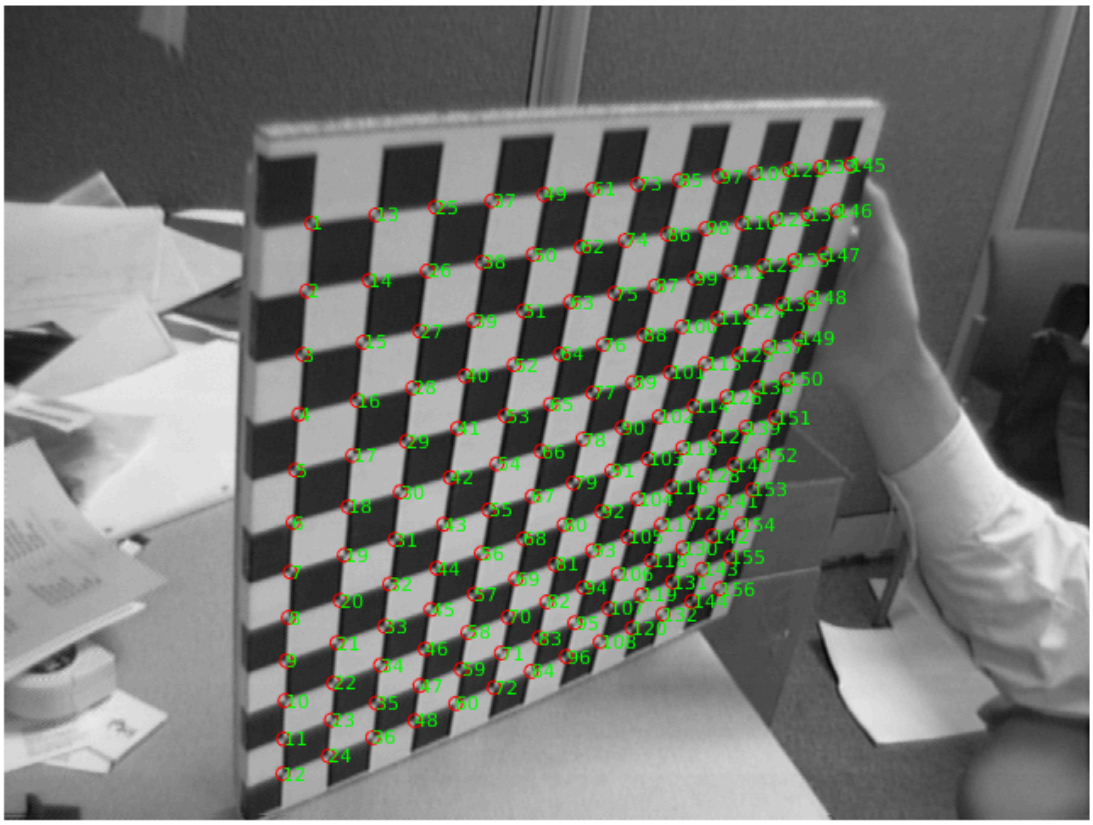
Detect and visualize checkerboard points

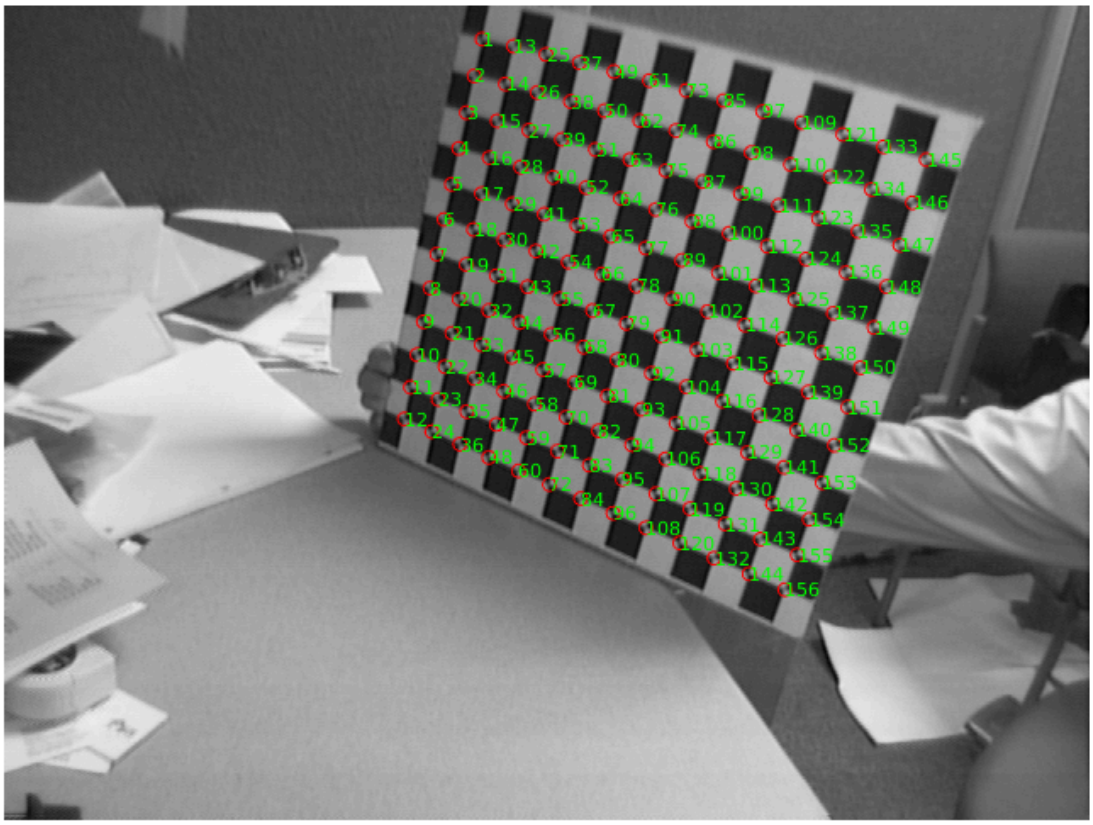
To estimate homographies, we need *a set of correspondences (at least four) between 3D points lying on the checkerboard patten and 2D points in the image*. The size of the squares of the pattern is 30mm. Let's visualize the detected points by superimposing text and markers to the image.

We now load four images, detect the interest points and store the images `I` and the points `XYpixel` in the structure array `imageData`.

```
clear all
%wd = '/home/ginevracoal/MEGA/Università/DSSC/semester_3/ComputerVision-PatternRecognition/zhan
%cd(wd)

iimage=[1 4 7 10];

%creating imagedata objects
imageData = create_imagedata(iimage);

% the show images
for ii=1:length(iimage)
  hnd=figure; %when creating a figure, a handle is returned, that can be used for accessing pro
  imshow(imageData(ii).I)
  hold on

  for jj=1:size(imageData(ii).XYpixel,1)
        x=imageData(ii).XYpixel(jj,1);
        y=imageData(ii).XYpixel(jj,2);
        plot(x,y,'or')

        hndtxt=text(x,y,num2str(jj));
        set(hndtxt,'fontsize',8,'color','green');
  end
end
```





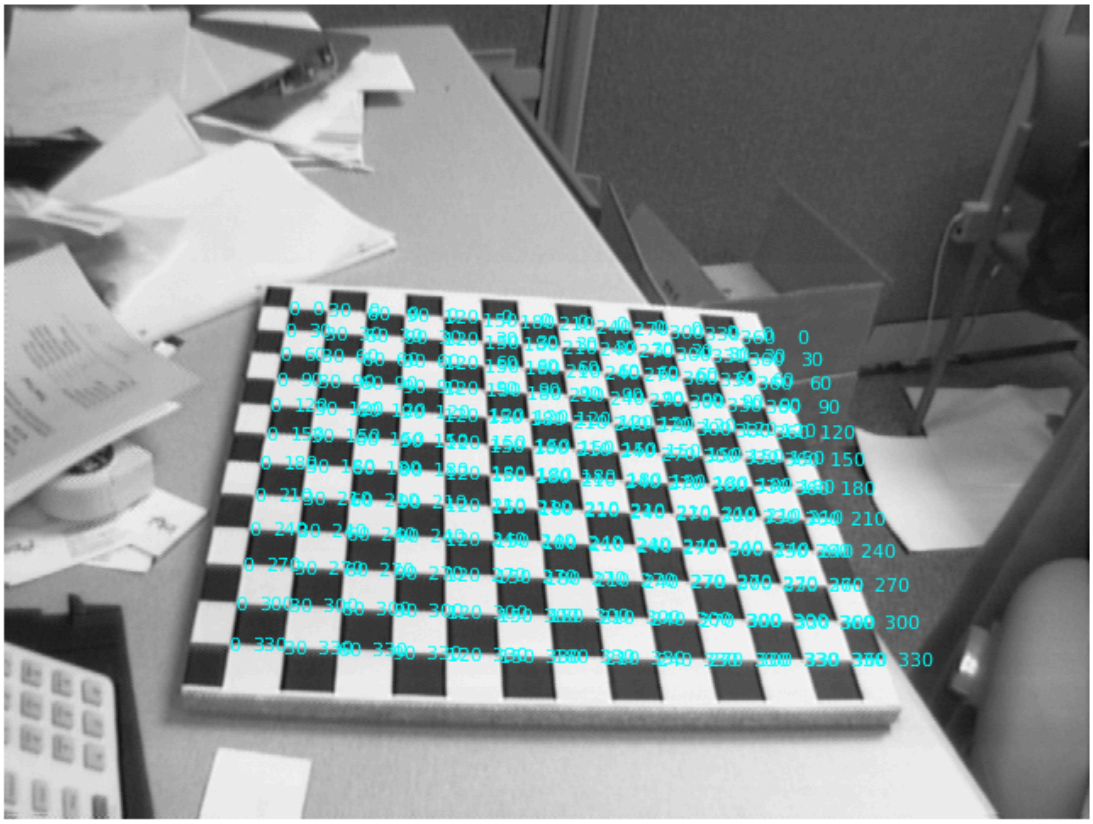


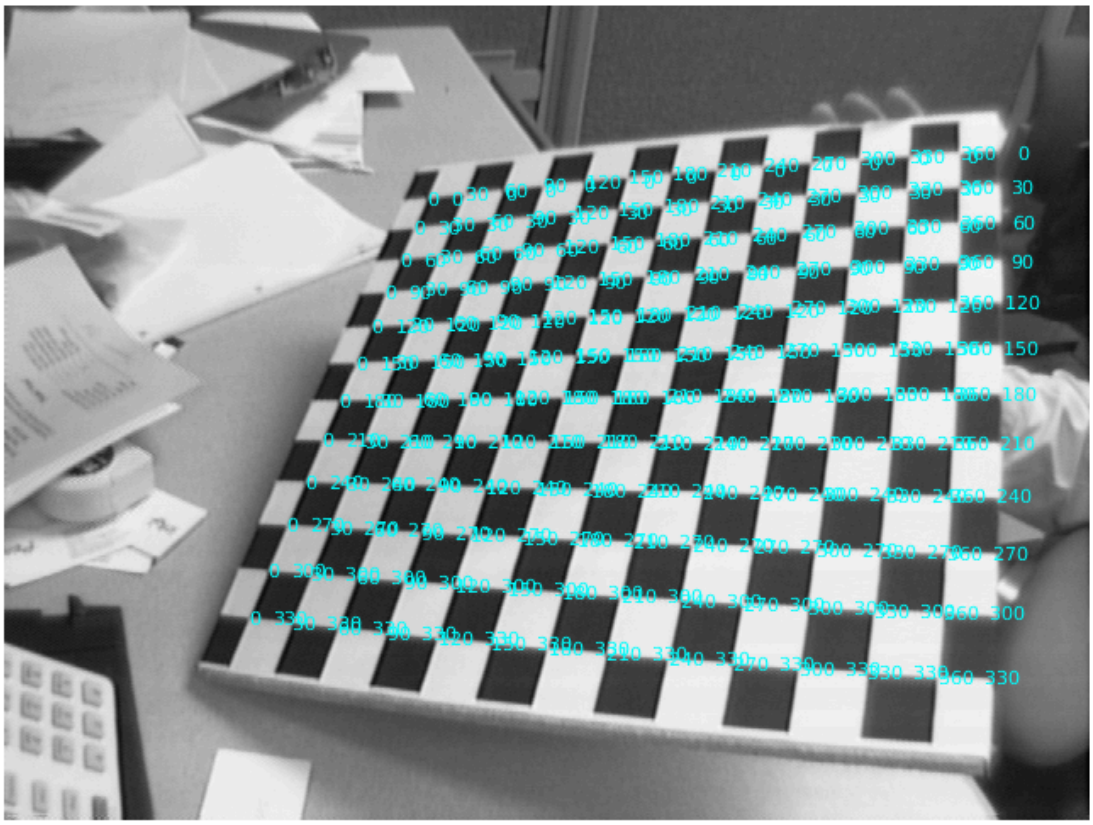
Establish initial correspondences

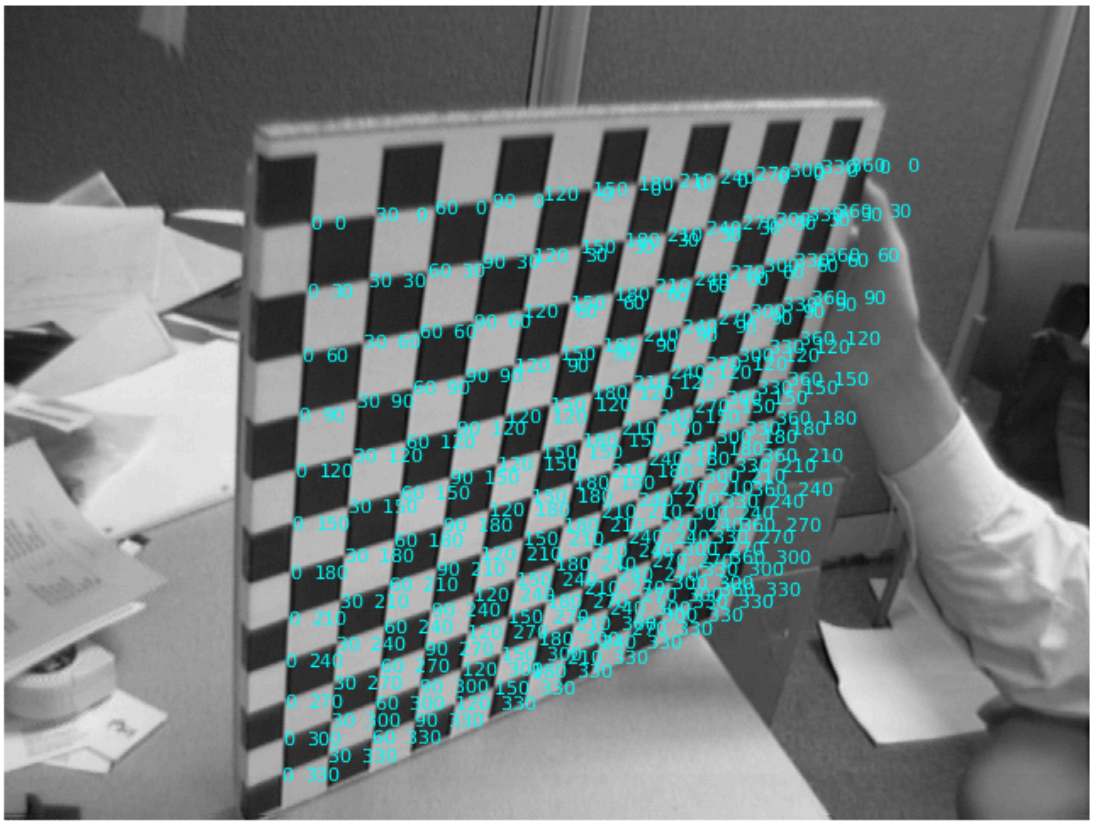
Now, given that the square size is 30mm and that the detected points are ordered, we can establish correspondences. This is step zero of the algorithm, where the pixel values are the ones just defined.

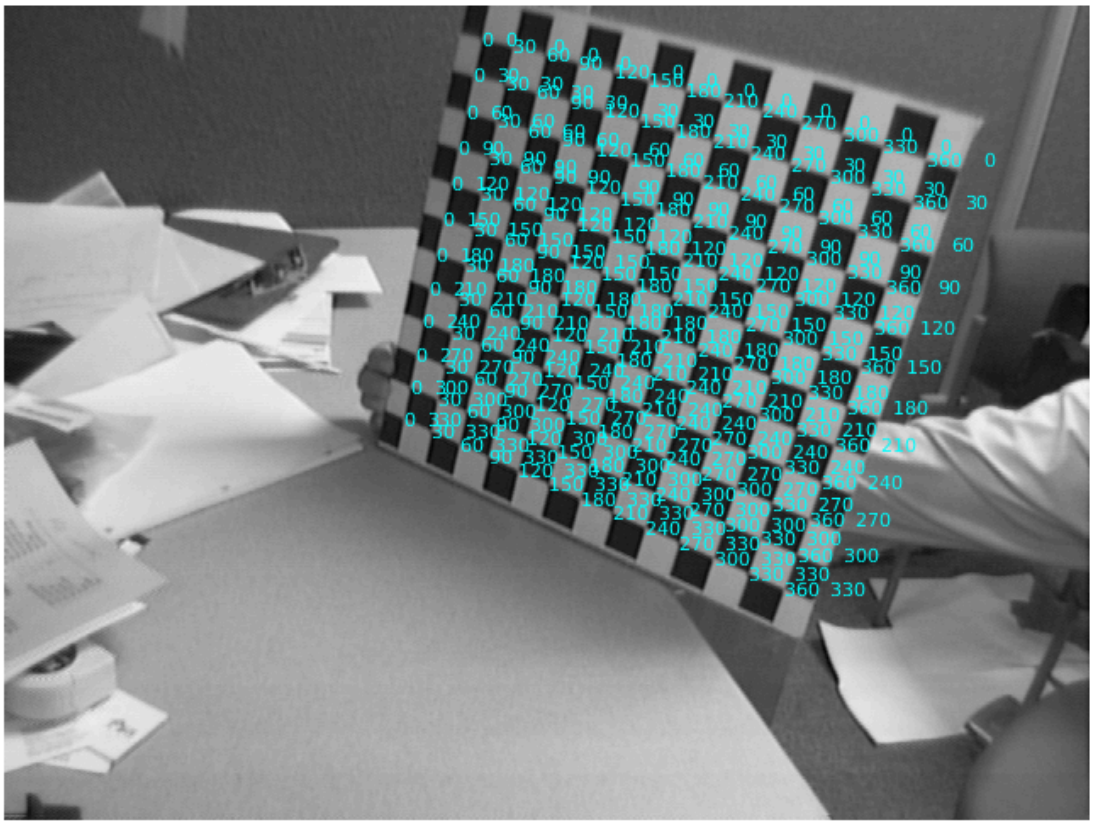
```
[imageData] = establish_correspondences(imageData,iimage);


% print the images and the associated pixel values
for ii=1:length(iimage) % for all the images
  figure
  imshow(imageData(ii).I) % display them

  XYpixel=imageData(ii).XYpixel;
  XYmm=imageData(ii).XYmm;

  for jj=1:length(XYpixel)
      [row,col]=ind2sub([12,13],jj); %linear index to row,col

      Xmm=imageData(ii).XYmm(jj,1);
      Ymm=imageData(ii).XYmm(jj,2);

      hndtxt=text(XYpixel(jj,1),...
```

```
            XYpixel(jj,2),...
            num2str([Xmm Ymm]));
        set(hndtxt,'fontsize',8,'color','cyan');

    end

end
```









Estimating homographies

Now, the field `XYmm` contains the coordinates in millimeters along the checkerboard and the field `XYpixel` the coordinates of the corresponding pixel in the image. Thus, the correspondences have been established.

Remember that to estimate the homography we need to estimate the 3x3 matrix H in the following:

$$w\begin{bmatrix}u\\v\\1\end{bmatrix} = P\begin{bmatrix}x\\y\\0\\1\end{bmatrix} = [p_1\ p_2\ p_3\ p_4]\begin{bmatrix}x\\y\\0\\1\end{bmatrix} = \underbrace{[p_1\ p_2\ p_4]}_{\doteq H}\underbrace{\begin{bmatrix}x\\y\\1\end{bmatrix}}_{\doteq m}$$

where u, v are the pixel coordinates and x, y the mm coordinates.

By partitioning H row-wise as

$$H = \begin{bmatrix}h_1^\top\\h_2^\top\\h_3^\top\end{bmatrix}$$

we obtain two homogeneous equations for each correspondence:

$$\begin{bmatrix} m^\top & 0^\top & -um^\top \\ 0^\top & m^\top & -vm^\top \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = 0.$$

We need to stack the equations and get a homogeneous linear system of equations such as:

$$Ax = 0$$

that we can solve in the least squares sense under the constraint that $x \neq 0$.

The solution is the right singular value of $A = U\Sigma V^\top$ associated with the smallest singular value, i.e. the rightmost column of V.

```
% estimate the homographies H, given the set of correspondences between mm
% and pixels

for ii=1:length(iimage)
    imageData(ii).H=estimate_homography(imageData(ii));
end
```

Find the intrinsic K

- Given H, for each image I calculate the corresponding V
- solve the linear 8x6 system Vb=0
- find the matrix B
- use Cholesky factorization to compute K

```
K = compute_intrinsic(imageData, iimage);
```

Find the extrinsics R and t

Finally, for each image I can find the extrinsics R and t, by using the corresponding homography.

Here I am also checking the values of the computed matrices against the homography H.

```
for ii=1:length(iimage)
    [imageData(ii).R,...
     imageData(ii).t] = compute_extrinsics(imageData(ii), K);
end
```

Reprojection error

Choose one of the calibration images and compute the total reprojection error for all the grid points (adding a figure with the reprojected points);

```
% here I chose image 1
idx=1;
```

```
[imageData(idx).est_proj,...
 imageData(idx).rep_error] = estimate_projections(imageData(idx), K);

% show the reprojection error
rep_error = imageData(idx).rep_error
```

```
rep_error = 6.5864
```

```
% plot the reprojected points
figure
im = imageData(idx);
imshow(im.I);
hold on
for jj=1:size(im.XYpixel)
    plot(im.XYpixel(jj,1),im.XYpixel(jj,2),'or') %o for circle and r for red
    plot(im.est_proj(jj,1),im.est_proj(jj,2),'og')
end
```

Radial distortion compensation

Add radial distortion compensation to the basic Zhang's calibration procedure.

```
% estimate radial distortion parameters k1 and k2
k = estimate_dist_param(imageData(idx), K);
```

Compute the total reprojection error with radial distortion compensation and make a comparison to the one without compensation.

```
% perform  radial distortion compensation on a chosen image
imageData = iterative_radial_compensation(imageData, iimage, K, k, idx);
```

```
rep_error = 5.8015
error_difference = 0.7849
rep_error = 5.7918
```

```
% show the result of radial compensation
figure
im = imageData(idx);
imshow(im.I)
hold on
for ii=1:size(im.XYpixel,1)
    plot(im.XYpixel(ii,1),im.XYpixel(ii,2),'or') %o for circle and r for red
    plot(im.est_proj(ii,1),im.est_proj(ii,2),'og')
end
```

Superimposing a 3D cylinder to each image

Superimpose an object to the calibration plane, in all the images employed for the calibration.

```
for ii=1:length(iimage)
    figure
    imshow(imageData(ii).I)
    P = K*[imageData(ii).R imageData(ii).t]; % 3x4

    hold on % superimpose something

    vtheta=0:0.01:2*pi; % take equally spaced points
    centerX=150; % these are mm of course
    centerY=150;
    radius=100;
    % discretize the circle by taking some coordinates on it
    vX=centerX+radius*cos(vtheta);
    vY=centerY+radius*sin(vtheta);

    % projections from plane z=0
    vZ=zeros(1,length(vtheta));
    homogeneous = [vX; vY; vZ; ones(1,length(vtheta))];
    proj_hom=P*homogeneous;
```

```
    proj=[ proj_hom(1,:)./proj_hom(3,:); ...
           proj_hom(2,:)./proj_hom(3,:)];

    f=fill(proj(1,:),proj(2,:),'g');
    set(f,'facealpha',.5)

    % projections from plane z!=0
    hold on

    vZ=repmat(5e6,1,length(vtheta));
    new_homogeneous = [vX; vY; vZ; ones(1,length(vtheta))];
    new_proj_hom = P*new_homogeneous;
    new_proj = [new_proj_hom(1,:)./new_proj_hom(3,:);...
                new_proj_hom(2,:)./new_proj_hom(3,:)];

    f2=fill(new_proj(1,:),new_proj(2,:),'r');
    set(f2,'facealpha',.5)
end
```