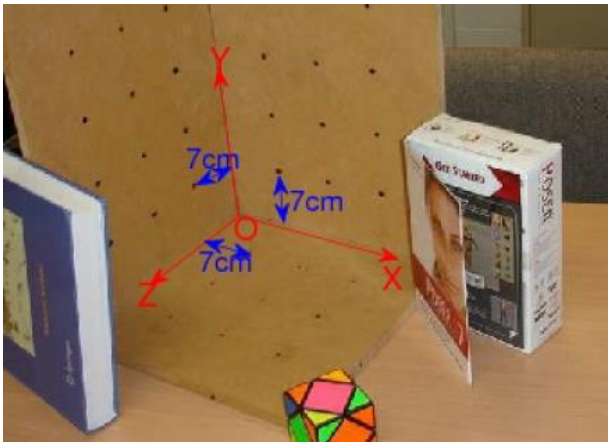
C-Lab-3 Report u5752771

Task1

The purpose of task 1 is camera matrix calibration. Given at least 6 pairs of known XYZ world coordinates and UV image coordinates, use DLT method to compute the camera calibration matrix. After that, the world coordinates are projected to the image plane, the mean square error are calculated for comparison. First of all, world coordinates are collected by reading locations on given 3D frame shown on RHS. Then the corresponding image coordinates in image plane are collected by command ginput. Alternatively, edge filter can be applied to collect the 2D coordinates. 12 pairs of coordinates are collected in this case for over-determined purpose.



Second, those pairs of coordinates are covert to homogeneous coordinates, which makes it invariant to scale and capable of performing linear transform.

After that, direct linear transformation (DLT) are performed. To solve $\mathbf{x}_i = \mathbf{P}\mathbf{X}_i$, if scale is not concerned, the expression is equivalent to

$$\mathbf{x}_i \times \mathbf{P}\mathbf{X}_i = 0$$

Thus

$$\left.\begin{aligned} \mathbf{x}_i &= (x_i, y_i, w_i)^T \\ \mathbf{P}\mathbf{X}_i &= \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \mathbf{p}_3^T \end{bmatrix} \mathbf{X}_i = \begin{pmatrix} \mathbf{p}_1^T \mathbf{X}_i \\ \mathbf{p}_2^T \mathbf{X}_i \\ \mathbf{p}_3^T \mathbf{X}_i \end{pmatrix} \end{aligned}\right\} \mathbf{x}_i \times \mathbf{P}\mathbf{X}_i = \begin{pmatrix} y_i \mathbf{p}_3^T \mathbf{X}_i - w_i \mathbf{p}_2^T \mathbf{X}_i \\ w_i \mathbf{p}_1^T \mathbf{X}_i - x_i \mathbf{p}_3^T \mathbf{X}_i \\ x_i \mathbf{p}_2^T \mathbf{X}_i - y_i \mathbf{p}_1^T \mathbf{X}_i \end{pmatrix}$$

Using DLT to factorize to unknowns, the matrix can be writen as

$$\begin{bmatrix} 0^T & -w_i \mathbf{X}_i^T & y_i \mathbf{X}_i^T \\ w_i \mathbf{X}_i^T & 0^T & -x_i \mathbf{X}_i^T \\ -y_i \mathbf{X}_i^T & x_i \mathbf{X}_i^T & 0^T \end{bmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix} = \mathbf{A}_i \mathbf{p} = 0$$

It is invariant up to scale, which means only 2 out of 3 equations are linearly independent, thus

C-Lab-3 Report u5752771

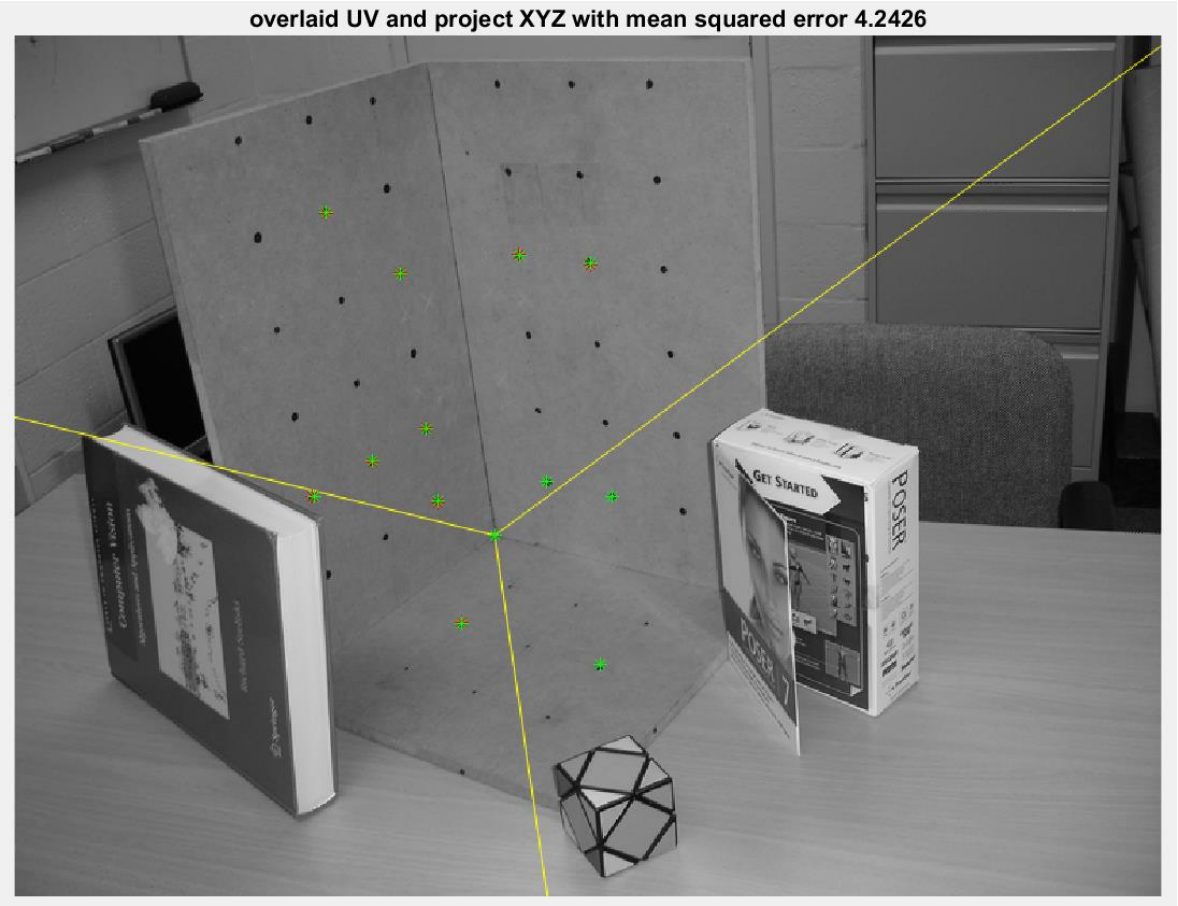
$$\begin{bmatrix} 0^T & -w_i\mathbf{X}_i^T & y_i\mathbf{X}_i^T \\ w_i\mathbf{X}_i^T & 0^T & -x_i\mathbf{X}_i^T \end{bmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix} = \mathbf{A}_i\mathbf{p} = 0$$

To solve for P, the trivial solution Ap = 0 is not intersting, alternatively we minimized $\|\mathbf{Ap}\|$. Additional constrain is to avoid Ap = 0 but $\|\mathbf{p}\| = 1$ is needed, so solution is given by 1D nullspace from singular value decomposition (SVD). By SVD method, the V matrix is ordered by the decending eigen value in diagonal matrix S. The solution P is given by taking the last column vector, which corresponding to the smallest eigen value.

Finally P is normalized and reshaped to 3 by 4 marix. For the first iamge in image set, the camera matrix is

4.4488	-2.0168	-5.9544	322.0481
0.1222	-7.3521	1.3461	334.5792
-0.0038	-0.0035	-0.0053	1

By using this camera matrix, the XYZ coordinates are projected back to image plane, the result is shown below



C-Lab-3 Report u5752771

The original coordinates are shown in red markers, but they are covered by the newly projected image coordinated from XYZ world coordinates. The mean square error is 4.24 in the scale of pixel, which indicates that the camera matrix is accurate and the projection is precise. The vanish points cound be out of the image. The three yellow lines is generated by setting homogeneous coordinates of three vectors of each axis to zero. When covert the image coordinates, they extend to infinity.

By decompositing the camera matrix, the intrinsic camera matrix K is

7.1260	0.0409	2.9209
0	7.0729	2.4195
0	0	0.0074

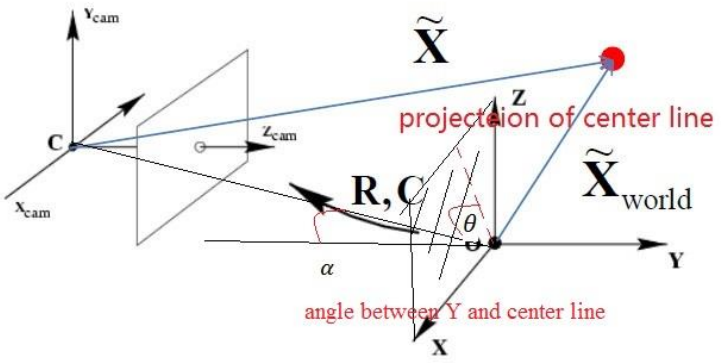
The rotation matrix R is

0.8346	-0.0858	-0.5442
0.1937	-0.8791	0.4356
-0.5157	-0.4689	-0.7170

Camera centre coordinate in 3D plane is the translation given by

78.3139
63.4874
91.0939

The goal is to calculated the α between Y axis and centre line drawn in the diagram below.



$$\sin\alpha = \cos\beta = \cos\left(\vec{n},\vec{u}\right) = \frac{|\vec{n}\cdot\vec{u}|}{|\vec{n}|\cdot|\vec{u}|}$$

Vector OC is (78.3139, 63.4874, 91.0939),

Vector Y is (0, 1, 0)

The sine of pitch angle is given by the dot product divided by the norm of OC and Y, in Matlab

```
Y_axis = [0,1,0]
temp = dot(t',Y_axis)/(norm(t')*norm(Y_axis));
pitch_angle = asin(temp)*180/pi;
```

The result of pitch angle is 27.856 degrees

C-Lab-3 Report u5752771

Code for Task1:

```
clc,clear;
load('img_uv_xyz.mat');                     %load points and image data
[C,A,V] = calibrate(img1, xyz, uv);              %camera matrix and plotting
[K, R, t] = vgg_KR_from_P(C, 1);            %C = K[R|t]
%K: camera calibration matrix 3x3, intrisics
%R: rotation matrix 3x3
%R|t or R[I|-CamCenter]: camera extrinsics
%C = K[R | ?RCamCenter]= [M| ?MCamCenter]
focal_length = K(1,1);                      %from intrisics matrix K
Y_axis = [0,1,0];                           %vector of y axis
temp = dot(t',Y_axis)/(norm(t')*norm(Y_axis));  %sine of the ptich angle
pitch_angle = asin(temp)*180/pi;%pitch angle calculation from combined rotation matrix
```

```
function [C,A,V] = calibrate(img1, xyz, uv)
xyz_T = xyz';                                   %tanspose of data
uv_T = uv';                                     %tanspose of data
uv_T = padarray(uv_T,[1,0],1,'post');           %pad to homogeneous coordinates
xyz_T = padarray(xyz_T,[1,0],1,'post');         %pad to homogeneous coordinates
num_of_samples = size(xyz_T,2);                 %number of poionts in xyz and uv
c_size = 12;                                    %size of camera calibration matrix
A = [];                                         %empty matrix far cat
for i = 1:num_of_samples                        %for each point
    xi = uv_T(1,i);
    yi = uv_T(2,i);
    wi = uv_T(3,i);
    Xi = xyz_T(:,i);
    Ai = [
            0, 0, 0, 0,    -wi * Xi',   yi * Xi' ;
            wi * Xi',     0, 0, 0, 0,   -xi * Xi'
            ];                                  %DLT matrix
    A = [ A ; Ai ];
end;
[~,~,V] = svd(A);                               %svd decomposition
c = V(:,c_size);                                %get the last column
c = c/c(c_size);                                %normalized
C = reshape(c,4,3)';                            %respape to camera matrix
temp = [0 0 0 7;                                %vanishing point ploting, add extra world coordiantes
        0 0 7 0;
        0 7 0 0;
        1 0 0 0;
    ];
xyz_T = cat(2, xyz_T, temp);                    %cat the extra world coordinates
uv_T_ver = C*xyz_T;                             %XYZ projection
for i = 1: size(uv_T_ver,2)                     %convert to non-homogeneous coordinate
uv_T_ver(1,i) = uv_T_ver(1,i)/uv_T_ver(3,i);
uv_T_ver(2,i) = uv_T_ver(2,i)/uv_T_ver(3,i);
uv_T_ver(3,i) = uv_T_ver(3,i)/uv_T_ver(3,i);
end
uv_ver = round(uv_T_ver(1:2,:)');               %round to display, and display the vanishing line
img1 = insertShape(img1,'Line',[uv_ver(end-3,:) uv_ver(end-2,:); ...
                                uv_ver(end-3,:) uv_ver(end-1,:); ...
                                uv_ver(end-3,:) uv_ver(end,:); ...
                                                                ]);
imshow(img1);
hold on
plot(uv(:,1), uv(:,2), 'r*');
hold on
distance = sum(sqrt(sum((uv_ver(1:end-4,:) - uv) .^ 2)));   %element wise mean squared error
str = sprintf('overlaid UV and project XYZ with mean squared error %0.5g',distance);
plot(uv_ver(:,1), uv_ver(:,2), 'g*');title(str);
```

C-Lab-3 Report u5752771

Task 2:

The purpose of task 2 is to perform homography transformation between two images.

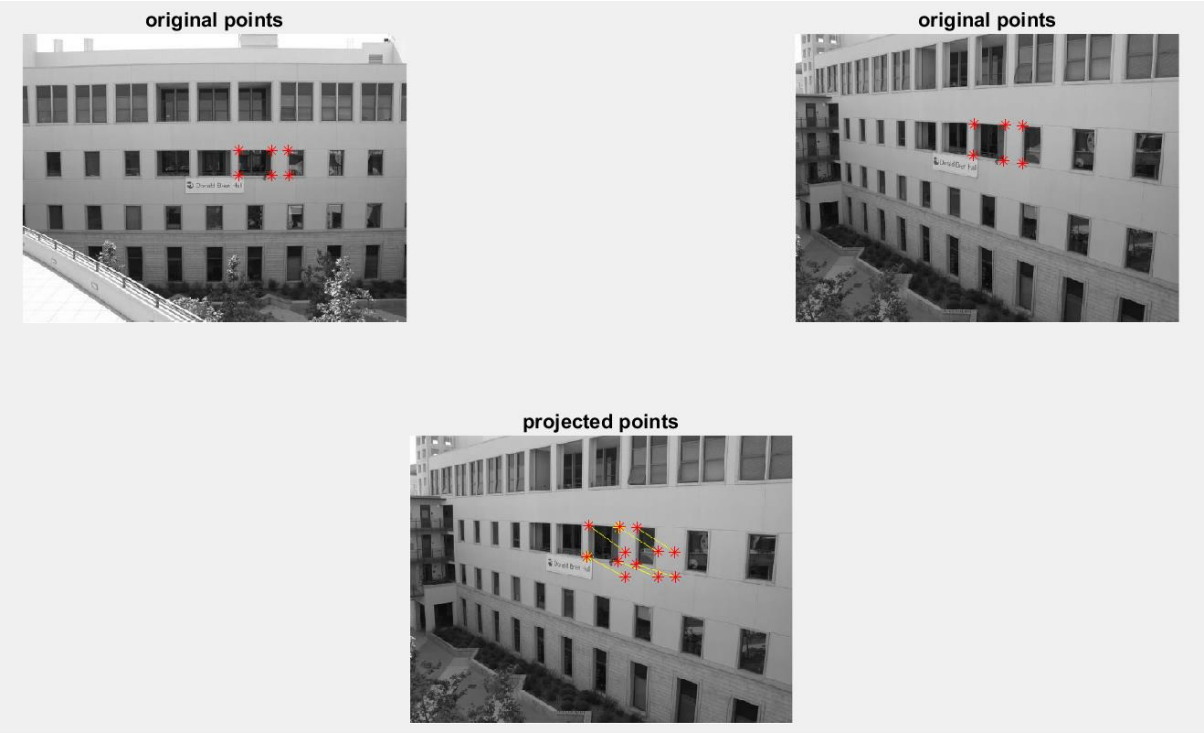
First of all, corresponding sets of points in both images are collected in same manner.

Second, homography matrix is calculated by DLT followed by SVD method. The only difference is the size of homography is 3 by 3, while camera matrix is 3 by 4 due to the presence of 3D coordinates. By slightly change the size of A matrix, the rest operations are the same.

The result is given below, the first two images are the original images where the corresponding sets of points are collected to calculate the homography matrix. In the third image, points from first image are marked, then projection is performed, newly projected coordinates are also marked on the third image. Those two sets of markers are linked by yellow lines for visualization.

The calculated homography matrix is

0.2028	-0.0214	90.4076
-0.1355	0.7901	-9.0089
-0.0014	3.8842e-05	1



Code for Task2:

```
clc,clear;
img1 = imread('Left.jpg');                    %load iamge
img2 = imread('Right.jpg');
img1 = rgb2gray(img1);
img2 = rgb2gray(img2);
load('clab4_2_xy_data.mat');                  %load xy coordinates
[H, xy2_T] = DLT(xy1,xy2);                    %DLT transform of image2 to image1, retunrn H matrix
```

C-Lab-3 Report u5752771

```
xy1_T_ver = H*xy2_T;                                %coordinates projection form image2 to image1
for i = 1: size(xy1_T_ver,2)                        %convert to non-homogeneous coordinate
xy1_T_ver(1,i) = xy1_T_ver(1,i)/xy1_T_ver(3,i);
xy1_T_ver(2,i) = xy1_T_ver(2,i)/xy1_T_ver(3,i);
xy1_T_ver(3,i) = xy1_T_ver(3,i)/xy1_T_ver(3,i);
end
xy1_ver = round(xy1_T_ver(1:2,:)');
figure;                                             %visulization
subplot(2,2,1);imshow(img2);title('original points');hold on;
plot(xy2(:,1), xy2(:,2), 'r*');
subplot(2,2,2);imshow(img1);title('original points');hold on;
plot(xy1(:,1), xy1(:,2), 'r*');
img1 = insertShape(img1,'Line',[xy2(:,:) xy1_ver(:,:)]);
subplot(2,2,3:4);imshow(img1);title('projected points');hold on;
plot(xy2(:,1), xy2(:,2), 'r*');hold on;
plot(xy1_ver(:,1), xy1_ver(:,2), 'y*');hold on;
plot(xy1(:,1), xy1(:,2), 'r*');
```

```
function [H xy2_T] = DLT(xy1, xy2)
xy1 = round(xy1);                                   %round the raw xy data from giput
xy2 = round(xy2);                                   %round the raw xy data from giput
xy1_T = xy1';                                       %transpose matrix
xy2_T = xy2';                                       %transpose matrix
xy2_T = padarray(xy2_T,[1,0],1,'post');             %convert to homogeneous
xy1_T = padarray(xy1_T,[1,0],1,'post');             %convert to homogeneous
num_of_samples = size(xy1_T,2);                     %number of points in raw data
c_size = 9;                                         %size of homography matrix
A = [];                                             %empty matrix for cat
for i = 1:num_of_samples
    xi = xy1_T(1,i);
    yi = xy1_T(2,i);
    wi = xy1_T(3,i);
    Xi = xy2_T(:,i);
    Ai = [
            0, 0, 0,    -wi * Xi',   yi * Xi' ;
            wi * Xi',     0, 0, 0,   -xi * Xi'
            ];
    A = [ A ; Ai ];                                 %DLT matrix
end;
[U,D,V] = svd(A);                                   %svd decomposition of A
h = V(:,c_size);                                    %take the last column
h = h/h(c_size);                                    %normalized to zero
H = reshape(h,3,3)';                                %reshape to homography matrix
end
```