ENGN4528 Clab-3 Report

Junlin Han

U6835134

June 27, 2020

1 Task 1 3D-2D Camera Calibration

1.1 Calibrate function

We want to solve this equation below,and find the value from p00 to p32.Following the DLT algorithm,we first build A from the input points then apply SVD decomposition to get C.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & p_{22} \\ p_{30} & p_{31} & p_{32} \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

I use DLT algorithm instead of normalized DLT algorithm with n=6,six 2-D calibration points.The data including image,2-D calibration points matrix,3-D calibration points matrix are stored in mydata.mat.

Direct Linear Transformation (DLT)

Objective

Given $n \geq 6$ 2D-to-3D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{X}_i\}$, determine the 3x4 projection matrix $\mathbf{P}$ such that $\mathbf{x}_i = \mathbf{P}\mathbf{X}_i$

Algorithm

(i) For each correspondence $\{\mathbf{x}_i \leftrightarrow \mathbf{X}_i\}$ compute $\mathbf{A}_i$. Usually only the two first rows are needed.

(ii) Assemble n 2x12 matrices $\mathbf{A}_i$ into a single $2n$x12 matrix $\mathbf{A}$

(iii) Compute the SVD of $\mathbf{A}$. The solution for $\mathbf{p}$ is the last column of $\mathbf{V}$

(iv) Normalize $\mathbf{p}$ to 1, and rearrange to obtain $\mathbf{P}$

My calibrate function is listed here, I also projects the XYZ coordinates back into image coordinates using the calibration matrix and Lines from the origin to the vanishing points in the X, Y and Z directions

```
function C = calibrate(im, XYZ, uv)
m = length(uv(:,1));
A = zeros(m*2,12);
for i = 1:m
   %get the 3D vertices and their 2D corresponding points
   X = XYZ(i,1);
   Y = XYZ(i,2);
   Z = XYZ(i,3);
   u = uv(i,1);
   v = uv(i,2);
   %assemble into A
   %odd row
   A(i*2-1,:) = [0,0,0,0,X,Y,Z,1,-v*X,-v*Y,-v*Z,-v];
   %even row
   A(i*2,:)=[X,Y,Z,1,0,0,0,0,-u*X,-u*Y,-u*Z,-u];
end
%apply SVD decomposition,p is same size as A
[¬,¬,V] = svd(A);
%solution,last column
C = V(:,12);
%size 12*1
C = C/C(12);
C = reshape(C,[4 3]);
C=C';
XYZ=[XYZ,ones(m,1)];
xy = C*XYZ';
xy(1,:) = xy(1,:)./xy(3,:);
xy(2,:) = xy(2,:)./xy(3,:);
xy=xy(1:2,:);
xy=xy';
distance = (xy - uv).^2;
distance = sum(sum(distance));
n = length(uv(1,:));
%calculate mean sqaure error
distance =distance/(m*n);
disp(['Mean Square Error: ',num2str(distance)]);
imshow(im);
hold on;
plot(uv(:,1),uv(:,2),'ro');
plot(xy(:,1),xy(:,2),'g+');
hold off;
%draw vanishing line
end
```

This part of the code performs basic DLT algorithm for calculating Camera Matrix

C according to 3D-2D corresponded coordinates and image as input.It also shows the Mean Square Error and plot result. Code for drawing Lines from the origin to the vanishing points is a little long so I won't display it here. The basic procedure is similar to what we did, choosing 6 points for each direction and project them back into image coordinates using calibration matrix. I'm not sure where is the precisely vanishing points thus I only extend lines till the end of board.

1.2 Display the image

In this task, I'm using stereo2012a.jpg.The 'ro' points (red circle) in figure1 are the points I choose,stored as uv.The 'g+' points (green plus) are the points obtain by projecting the XYZ coordinates back into image coordinates.
From figure1,we can observe that the plot points are very close,Four green plus are exactly inside the corresponding red circles.Two green plus (lower right) can be seen slightly overlap with the red circle by zooming figure1.This is due to manually set of 2-D points.Normalizing the data improves the DLT quality since normalization can minimize the error.The green line shows X direction,yellow line shows Y direction and purple line shows Z direction.For each direction line,I choose six points with same direction in 3-D coordinates and project them back into image coordinates using calibration matrix.Then I get 6 pairs of x,y coordinates and I choose one of them thus I can do some arithmetic to get one more point (point3 in code) near the vanishing points.I use plot to plot the line between origin and point3.

1.3 List the 3x4 camera calibration matrix C

The result of my calibration matrix is shown here.I also attach my mean square error here.

```
C =
   -6.4360  -19.6284   -1.7183  310.8657
   -8.2098  -13.5315   -2.0243  262.2702
   -0.0220   -0.0691    0.0089    1.0000
Mean Square Error: 3.9605
```

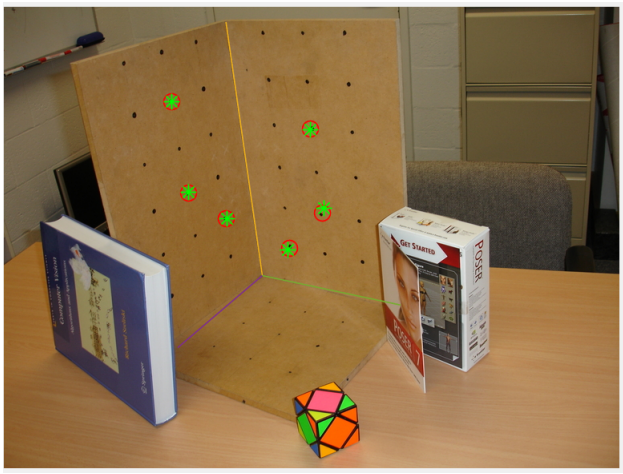

Figure 1: Plot result on stereo2012a

1.4 Decompose the C matrix into K, R, t, such that $C = K[R|t]$

I use given code $vgg - KR - from - P.m$ to do decomposition.
Code are shown here.

```
C=calibrate(I,XYZ,uv);
disp('C = ');
disp(C)
[K, R, t] = vgg_KR_from_P(C);
disp('K = ');
disp(K);
disp('R = ');
disp(R);
disp('t = ');
disp(t);
```

Results of K,R,t are shown here.

```
K =
   38.1497   43.5723  277.5720
         0   73.6860  205.4600
         0         0    1.0000
R =
   -0.6624    0.2992    0.6868
    0.6862   -0.1255    0.7165
```

```
    0.3005    0.9459   -0.1222
t =
   11.4192
   11.5846
    5.8114
```

From the lecture, P is defined as

$$P = K[R|t] = K[R| - RC] = [M| - MC]$$

In the provided codes, t is defined as

```
if nargout > 2
  t = -P(:,1:N)\P(:,end);
end
```

Actually, this is 'C' in the lecture slide. So it's not precisely the same.Due to this reason,our focal length is a little weird.

1.5 What is the focal length of the camera

From the K,intrinsic matrix we can find focal length

$$\mathbf{K} = \begin{pmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

Where fx,fy are focal length.From my K matrix we can obtain fx=38.1497,fy=73.6860 in cm.We know $focal = \sqrt{f_x^2 + f_y^2}$,so we can get focal length=82.97cm.That's quite a weird answer.
I think our camera matrix C is correct from the visual analysis and MSE error,we plot the nearly same points in figure1.As far as I'm concerned, a plausible reason is that the 'K', 'R', 't' in the provided codes are not exactly same as our lecture slide as I mentioned in 1.4.

1.6 What is the pitch angle of the camera with respect to the X-Z plane in the world coordinate system?

We can calculate pitch angle using rotation matrix R.

```
R =
   -0.6624    0.2992    0.6868
    0.6862   -0.1255    0.7165
    0.3005    0.9459   -0.1222
```

A single rotation matrix can be formed by multiplying the yaw, pitch, and roll rotation matrices to obtain [1]

$R(\alpha,\beta,\gamma) = R_z(\alpha)R_y(\beta)R_x(\gamma)$

$$= \begin{matrix} cos(\alpha)cos(\beta) & cos(\alpha)sin(\beta)sin(\gamma) - sin(\alpha)cos(\gamma) & cos(\alpha)sin(\beta)cos(\gamma) + sin(\alpha)sin(\gamma) \\ sin(\alpha)cos(\beta) & sin(\alpha)sin(\beta)sin(\gamma) + cos(\alpha)cos(\gamma) & sin(\alpha)sin(\beta)sin(\gamma) - cos(\alpha)sin(\gamma) \\ -sin(\beta) & cos(\beta)sin(\gamma) & cos(\beta)cos(\gamma) \end{matrix}$$

So we can get this equation:

$\beta = tan^-1(-R_{31}/\sqrt{R_{32}^2 + R_{33}^2})$ So we can substitute values in our rotation matrix R thus we get $\beta = -18.3 degree$.

2 Two-View DLT based homography estimation

2.1 List your source code for homography estimation and display the two images and the location of six pairs of selected points

We first find 6 pairs of corresponding points.I use Matlab inbuilt function **ginput** to do so.I collect 6 points from left image first.After getting 12 points,I store them as mydata2.mat.

```
left=imread('Left.jpg');
right=imread('Right.jpg');
get data,comment ifload mydata2.
subplot(1,2,1);
imshow(left);
title('Left Image');
hold on;
xy_left= ginput(6);
plot(xy_left(:,1),xy_left(:,2),'g*');
hold off;
subplot(1,2,2);
imshow(right);
title('Right Image');
hold on;
```

```
xy_right= ginput(6);
plot(xy_right(:,1),xy_right(:,2),'r*');
hold off;
x1=xy_left(:,1);
y1=xy_left(:,2);
x2=xy_right(:,1);
y2=xy_right(:,2);
```

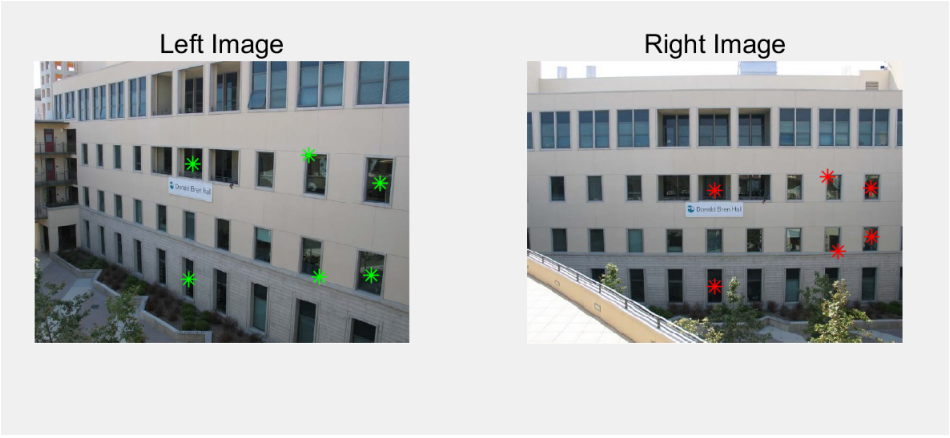

Figure 2: Six pairs of points of given images

Figure2 shows the location of six pairs of points.
The DLT algorithm for 2D homograph matrix is quite similar to the DLT algorithm for camera matrix.I also follow the algorithm in lecture slide W6B.
We want to solve this equation below.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

We can write it as two eqations.

$$x'_i * (h_{20} * x_i + h_{21} * y_i + h_{22}) = h_{00} * x_i + h_{01} * y_i + h_{02}$$

$$y'_i * (h_{20} * x_i + h_{21} * y_i + h_{22}) = h_{10} * x_i + h_{11} * y_i + h_{12}$$

Then,we want to get the eigenvector that belongs to the smallest eigenvalue of matrix A.We can assemble matrix A with this form.

$$[A] = \begin{bmatrix} 0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \\ x_j & y_j & 1 & 0 & 0 & 0 & -x'_j x_j & -x'_j y_j & -x'_j \\ where & i & represent & odd & row & ,j & represent & even & row \\ ... & & & & & & & & \end{bmatrix}$$

Also,this is the way we assemble matrix A for Task 1. My source code is given here.

```
function H = homography(u2Trans, v2Trans, uBase, vBase)
n = length(u2Trans);
% The length of u2Tran, v2Tran, uBase, vBase must be the same.
%build matrix A with size 2n * 9.
A = zeros(2*n,9);
%Form the matrix A;
for i = 1:n
    x1 = uBase(i);
    y1 = vBase(i);
    x2 = u2Trans(i);
    y2 = v2Trans(i);
    %odd row
    A(2*i-1,:) = [0,0,0,x2,y2,1,-y1*x2,-y1*y2,-y1];
    %even row
    A(2*i,:) =  [x2,y2,1,0,0,0,-x1*x2,-x1*y2,-x1];
end
%apply SVD decomposition,V is same size as A
[¬,¬,V] = svd(A);
%solution,last column
H = V(:,9);
%reshape H and normalize
H = H/H(9);
H = reshape(H,[3 3]);
end
```

2.2 List the 3x3 camera homography matrix H

The result of H is shown here.

```
H =
    0.2736   -0.1048   -0.0012
   -0.0160    0.8667    0.0001
   83.5206  -25.5179    1.0000
```

2.3 Warp the left image according to the calculated homography

I warp my left image with my H matrix and matlab inbuilt function ***projective2d*** and ***imwarp***.Result is shown in figure3.
From figure3,we can observe that wrap image and left image share similar perspective which proves our algorithm works well.
My code for this part is here.

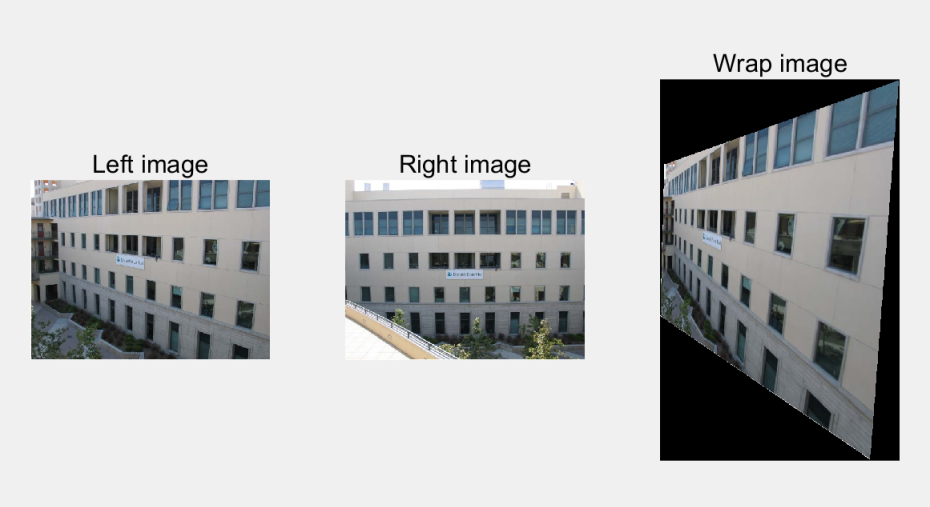

Figure 3: Warp result

```
tform = projective2d(H);
wrap= imwarp(left,tform);
figure;
subplot(1,3,1);
imshow(left);
```

```
title('Left image');
subplot(1,3,2);
imshow(right);
title('Right image');
subplot(1,3,3);
imshow(uint8(wrap));
title('Wrap image');
```

2.4 Study the factors that affect the rectified results

For this part,I'd like to discuss 3 factors that may influence the results.

2.4.1 The number of points we choose

We know that at list 4 pairs of points are needed to apply DLT algorithm.Matrix H has 8 degrees of freedom and each pair of points will give two linear independent equations. Thus, we need at least 4 pairs of points.
But will 4 pairs of points work well?If I choose points with similar positions as we did before,we can check it by comparing the results.
From figure4 and figure5,we can conclude that 4 pairs are enough to perform two-view DLT algorithm.The results of warp images are just same by human eyes.

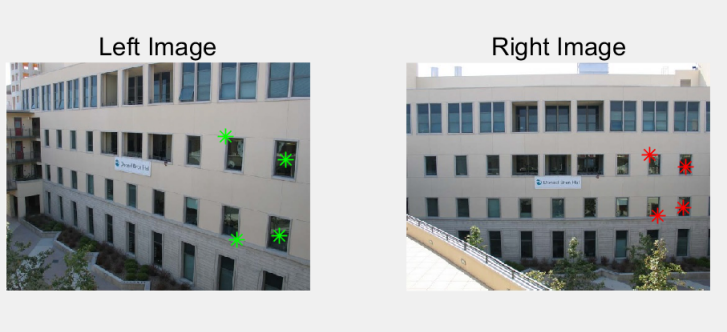

Figure 4: Four pairs of points of given image

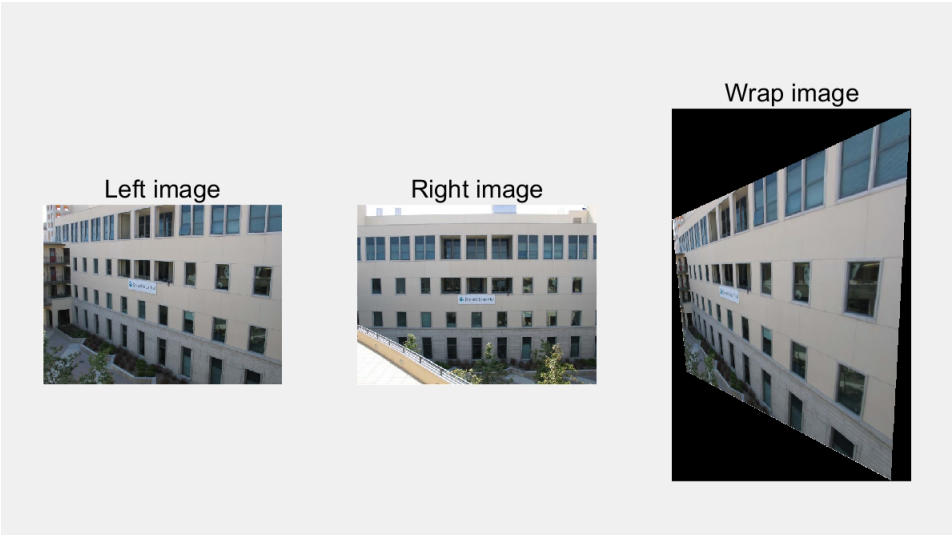

Figure 5: Warp result of 4 pairs of points

2.4.2 The region of selected points

In figure4,I choose 4 points all on right region in given image.So for this part,I'd like to choose 4 points all in left region.
We can observe from figure5 and figure7,the results are significantly different.If we choose more points in left region,the wrap image will tend to have larger length in left side and vice versa.

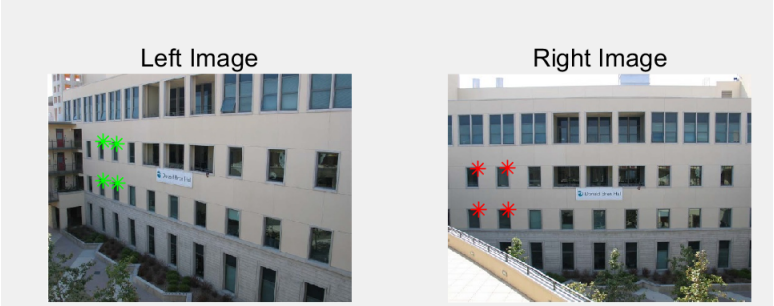

Figure 6: Four pairs of points in left region

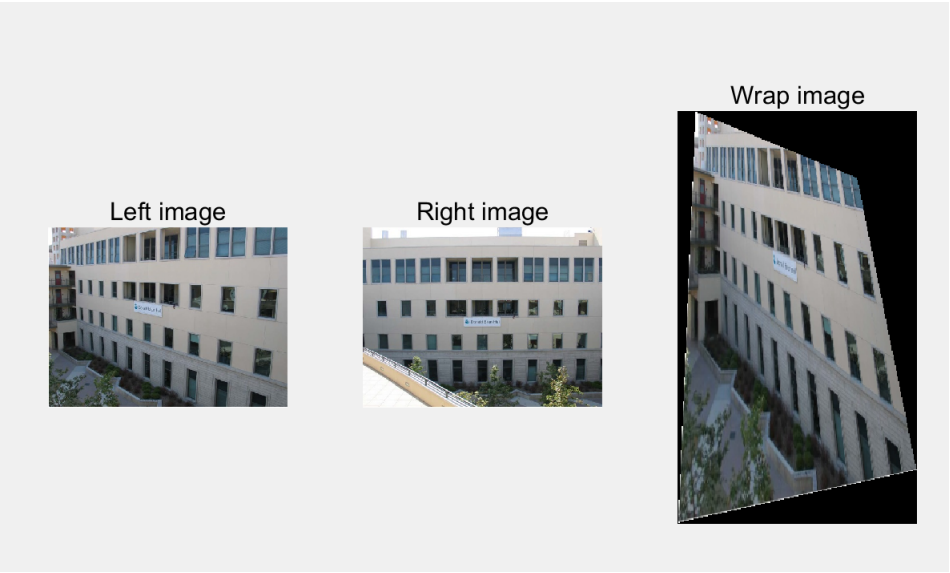

Figure 7: Warp result of 4 pairs of points in left region

2.4.3 distance between selected points

We know that the region of selected points can impact the result.So in this part,I will choose 4 points in the middle region of image with small distances,and 4 points also in the middle region of image with larger distances. They will roughly share the same central point.

From figure8 and figure9,we can observe that the distances between points are

Figure 8: Four pairs of points in small distances

obvious different while the central of 4 points are same.

We can observe from figure 10 and figure 11 that the warp results are almost

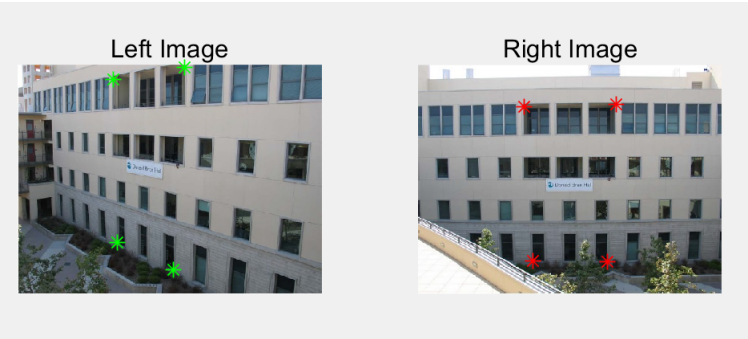

Figure 9: Four pairs of points in large distances

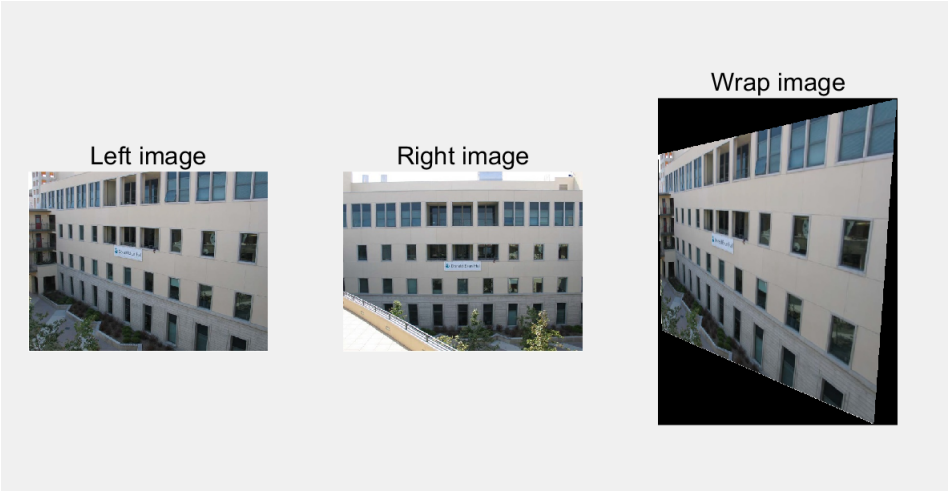

Figure 10: Four pairs of points in small distances,warp result

same.Hence we can conclude that the distance between selected points is irrelevant to warp result.

References

[1] Steven M.LaValle. *Planning algorithms*. Yaw, pitch, and roll rotations,chapter 3.2.Cambridge University Press.

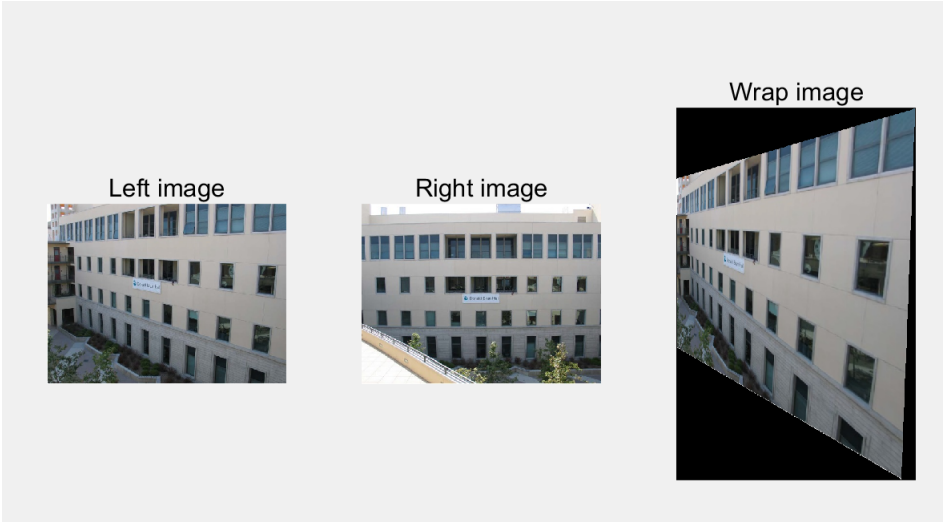

Figure 11: Four pairs of points in large distances,warp result