Homework 11

|  |  |
| --- | --- |
| **Meta information** | |
| Name | Savan Kiran |
| Program | Masters in Computer Science |
| Questions skipped | N/A |
| Questions substituted | N/A |
| Extra credit questions | N/A |

# PART A

In this task we use RANSAC to ignore the effects of outliers while fitting a line to the given 300 points. Here, we assume 1/4th of the points lie on a pretty good line. The number of iterations for the RANSAC algorithm is calculated as follows,

To determine inliers in every iteration, I used the best N% method where N is same as w, i.e., 1/4th of the points.

Below is the code snippet for the same,

data=importdata('line\_data\_2.txt');

figure('Name','Part A - RANSAC','NumberTitle','off');

scatter(data(:,1),data(:,2),'b');

hold on

data\_s=size(data,1);

n=2;

w=0.25;

N=w;

p=0.99;

k=ceil(log(1-p)/log(1-w^n));

lineX=[];

lineY=[];

best\_err=999999;

for i=1:k

p=randperm(data\_s,n);

pt\_dist=zeros(data\_s,2);

for j=1:data\_s

pt\_dist(j,1)=j;

pt\_dist(j,2)=distance\_point\_to\_line(data(j,:),data(p(1),:),data(p(2),:));

end

pt\_dist=sortrows(pt\_dist,2);

bestN=floor(N\*data\_s);

in\_pts=zeros(bestN,2);

for j=1:bestN

in\_pts(j,1)=data(pt\_dist(j,1),1);

in\_pts(j,2)=data(pt\_dist(j,1),2);

end

[lx, ly, err]=model(in\_pts);

if err < best\_err

lineX=lx;

lineY=ly;

best\_err=err;

end

end

plot(lineX,lineY,'r');

disp(['RANSAC line error: ' num2str(best\_err)]);

The scatter plot of points and the fitted line is shown in Figure 1.

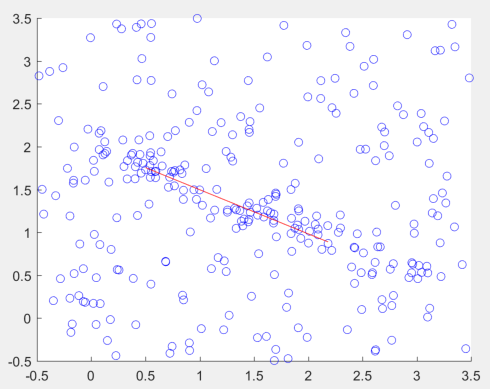


Figure 1. Line fitted (shown in red) using RANSAC algorithm

The error of the fitted line based on the inliers is - 0.13243.

# PART B

In this task, we implement the DLT method to compute homography. We first experiment with 4, 5 and 6 randomly generated points inside [0,1]x[0,1] over 10 samples. Below is the code snippet for the same,

for k=4:6

errs=zeros(10,1);

for i=1:10

errs(i,1)=random\_dlt(k);

end

errs=errs.^2;

rms\_err=sqrt(mean(errs(:)));

disp(['k=' num2str(k) ' and rms\_err=' num2str(rms\_err)]);

end

function err=random\_dlt(k)

left=zeros(k,3);

right=zeros(k,3);

for i=1:k

left(i,1)=rand();

left(i,2)=rand();

left(i,3)=1;

right(i,1)=rand();

right(i,2)=rand();

right(i,3)=1;

end

H=homography(left, right, k);

left\_m=map(left, H);

[err err2]=rms\_error\_2(right,left\_m);

end

function H=homography(left, right, k)

U=[];

for i=1:k

l=left(i,:);

r=right(i,:);

rx=r(1);

ry=r(2);

z=[0 0 0];

temp1=[z -l ry.\*l];

temp2=[l z -rx.\*l];

temp3=[-ry.\*l rx.\*l z];

U=[U; temp1; temp2; temp3];

end

[eV,eD]=eig(U'\*U);

H=reshape(eV(:,1), [3 3])';

end

Table 1 shows the RMS errors of mapping for the ‘k’ random points using the homography found with the same points.

|  |  |
| --- | --- |
| No. of points | RMS error |
| 4 | 8.3772e-11 |
| 5 | 0.50226 |
| 6 | 3.617 |

Table 1. RMS errors for ‘k’ random points mapped using homography calculated

The RMS errors increase as number of points increase which makes sense as calculating homography includes more points, they try to uniformly map each point minimizing error. However, since the points are randomly generated, as number of points increase, the RMS error also increases.

In the next part, we use the data from assignment 9. For each pair of images (slide and video frame), we will manually collect 8 matching points using mouse clicking. Table 2 shows the points for each pair.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Image 1 | | Image 2 | | Image 3 | |
| Frame | Slide | Frame | Slide | Frame | Slide |
| 218,28,1 | 132,32,1 | 169,33,1 | 182,31,1 | 70,15,1 | 56,70,1 |
| 270,29,1 | 230,33,1 | 140,46,1 | 128,52,1 | 58,83,1 | 45,139,1 |
| 216,57,1 | 126,94,1 | 141,65,1 | 133,90,1 | 178,163,1 | 160,222,1 |
| 219,74,1 | 136,128,1 | 76,135,1 | 24,216,1 | 227,5,1 | 301,58,1 |
| 170,116,1 | 45,215,1 | 247,33,1 | 303,32,1 | 331,135,1 | 302,191,1 |
| 260,108,1 | 212,198,1 | 75,46,1 | 21,51,1 | 130,61,1 | 112,115,1 |
| 179,93,1 | 62,166,1 | 92,120,1 | 51,188,1 | 188,113,1 | 167,172,1 |
| 160,68,1 | 27,115,1 | 159,137,1 | 162,216,1 | 35,132,1 | 24,191,1 |

Table 2. 8 matching points found manually on slide and frame images

We select points corresponding to the letters as they’re easy to match on the counter part as against points in empty space.

We now take a subset of 4 points from the 8 as input to computing homography. With the computed homography, we map all 8 points (from the slide) to the video frame. We can see that the 4 points used to calculate homography perfectly overlap when mapped on the video frame whereas other 4 points are close by. Below images depicts this.

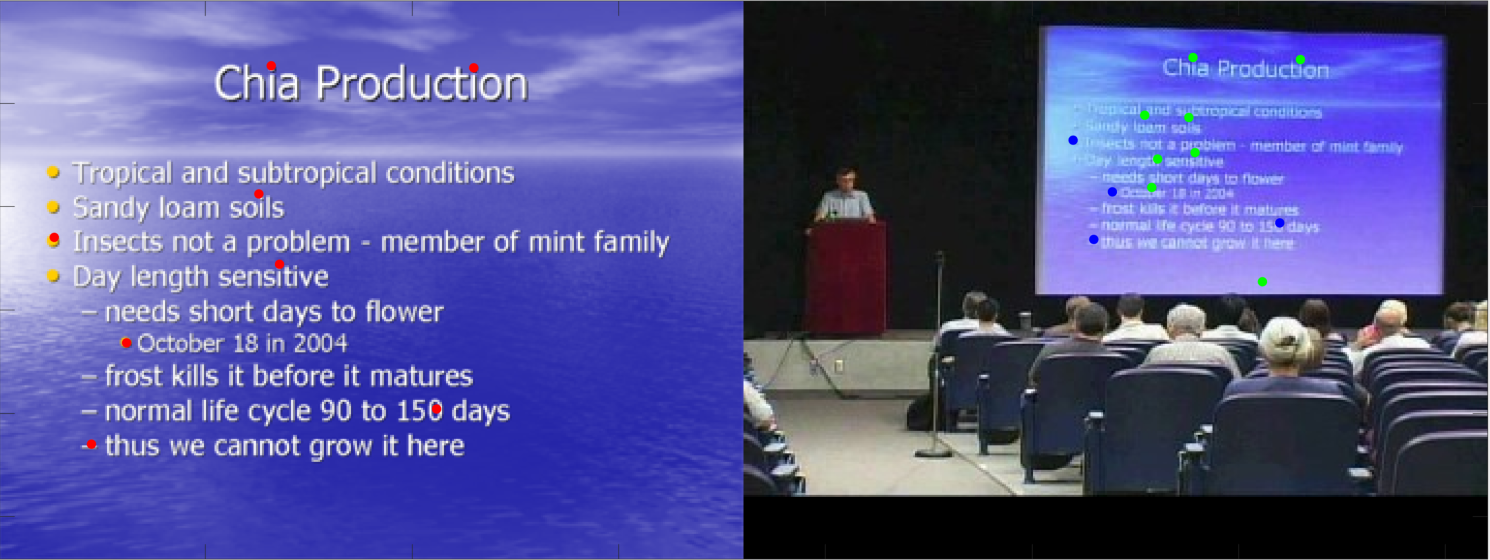


Figure 2. The red and blue points are 8 matching points whereas mapped points are shown in green

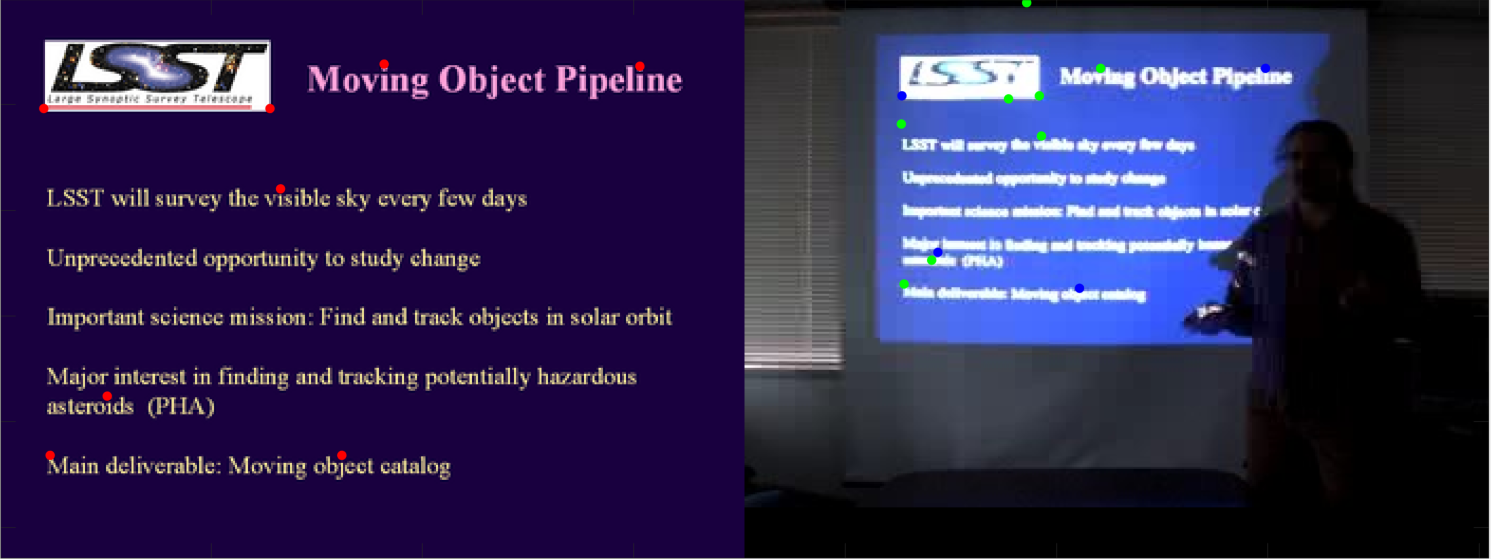


Figure 3. The red and blue points are 8 matching points whereas mapped points are shown in green

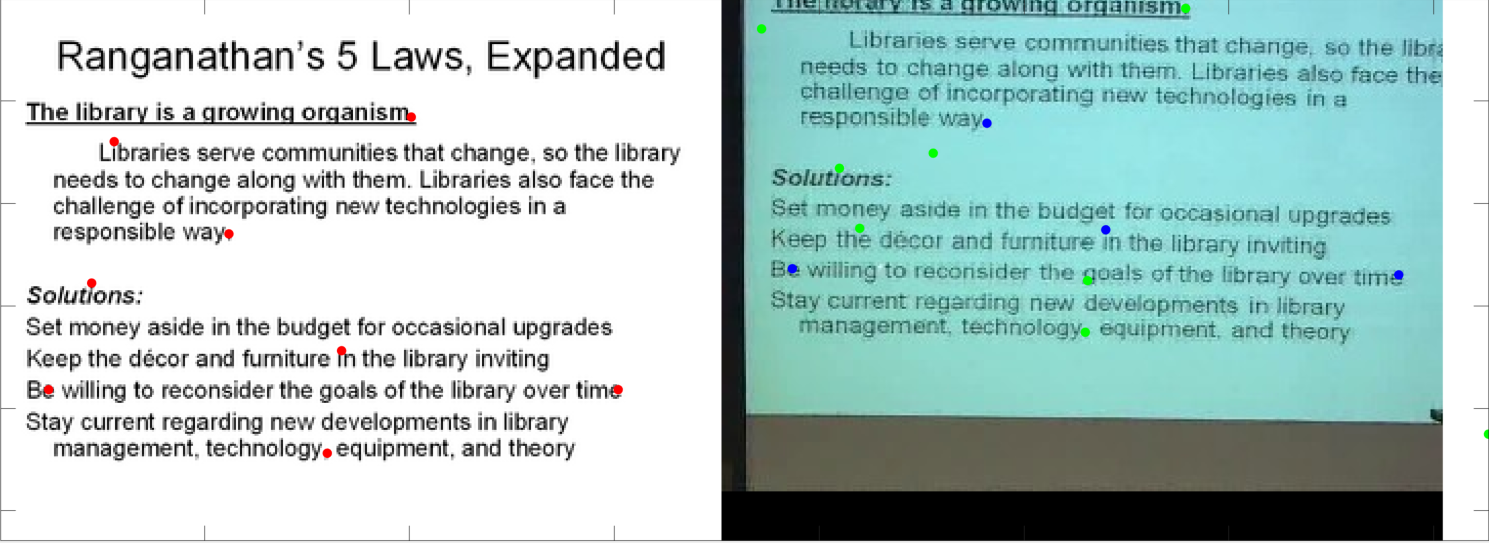


Figure 4. The red and blue points are 8 matching points whereas mapped points are shown in green

Below is the code snippet for the same,

function map\_points(im1\_s, im1\_f, left1, right1)

H1=homography(left1, right1, 4);

left1\_m=map(left1, H1);

% display points

end

function left\_m=map(left, H)

left\_m=[];

for i=1:size(left,1)

temp=(H\*left(i,:)')';

temp=[temp(1,1)/temp(1,3) temp(1,2)/temp(1,3) 1];

left\_m=[left\_m; temp];

end

end

# PART C

We will now use RANSAC algorithm to improve homography calculation in key point matching. In assignment 9, we extracted many features in the slides and video frame images. We then used the nearest neighbor to match to the best feature. However, in the video frame, the slide is at certain angle to the camera which leads to wrong coordinates of the features. We will try to re-map the coordinates by calculating the homography between the pair of images using their features. We will use RANSAC to calculate homography by removing the influence of the outliers

We follow the simple approach explained in the lecture slides.

Repeat many time

Randomly select enough matches to fit homograhy (i.e., 4)  
 Compute homography using DLT  
 Using that homography, measure error on best – 50% (N)

Output best one found

RANSAC helps. In assignment 9, a part of the feature matches we found were lying outside the slide in the video frame, whereas, now, only a couple of them lie outside. Visually, the matches seem to be more accurate. If we have good feature matches, considering bigger N for homography calculation is useful as it can reduce the error.

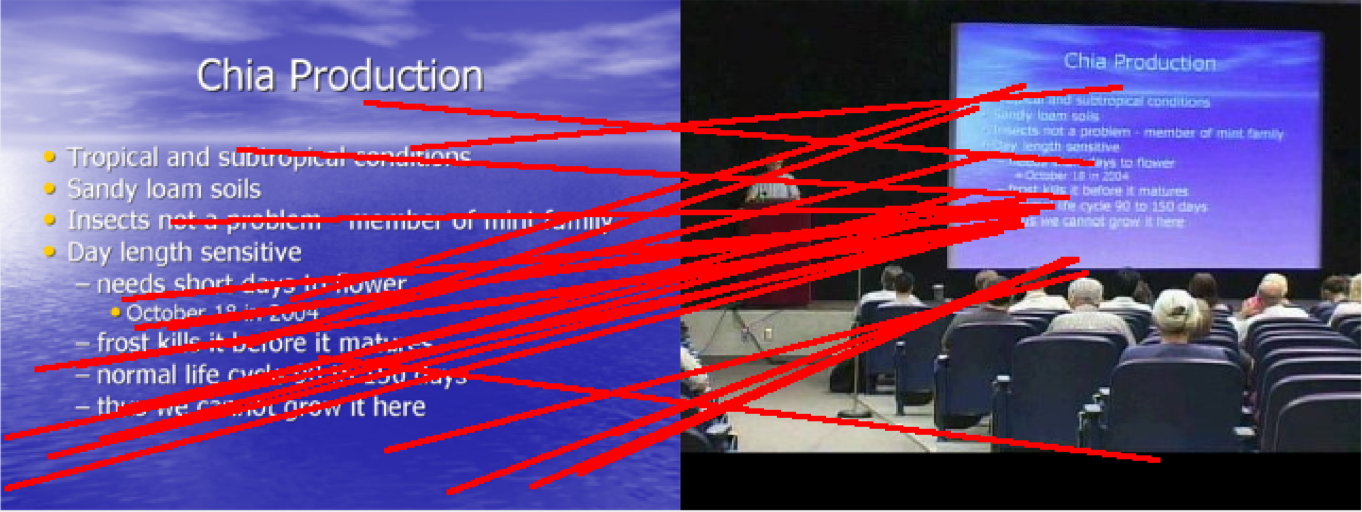


Figure 5. Image 1 – features matched using homography and RANSAC

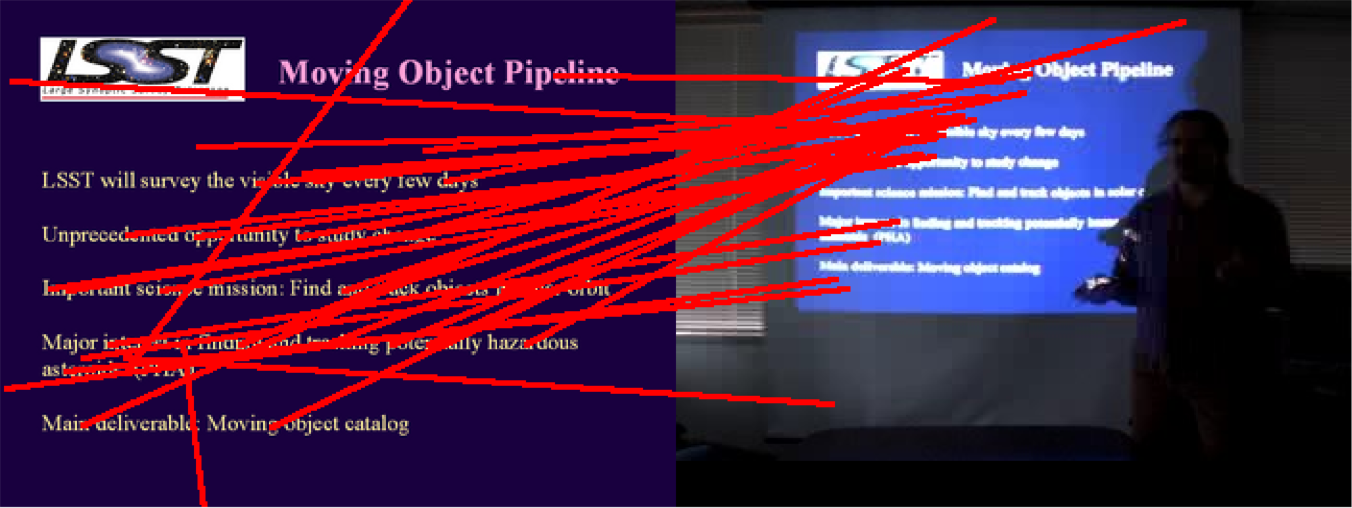


Figure 6. Image 2 – features matched using homography and RANSAC

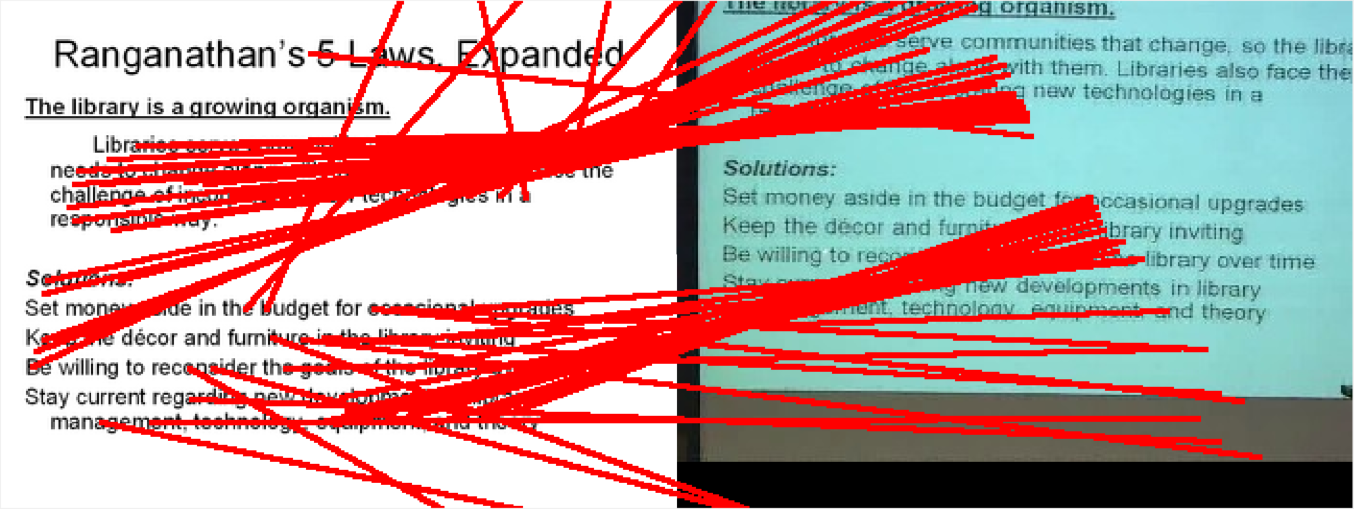


Figure 7. Image 3 – features matched using homography and RANSAC

Below is the code snippet for the same,

function homo\_ransac(frame\_pgm, frame\_color, slide\_pgm, slide\_color)

euclidean=0;lowe\_opt=1;ratio=0.96;

[match, ff, fs]=find\_matches(frame\_pgm, slide\_pgm, lowe\_opt, ratio, euclidean);

data\_s=size(match,1);

n=4;w=0.25;N=0.5;p=0.99;

k=ceil(log(1-p)/log(1-w^n));

bestH=[];

best\_err=999999;

for i=1:k

rand\_match=randperm(data\_s,n);

left=zeros(n,3);

right=zeros(n,3);

for j=1:n

left(j,1)=fs(2,match(rand\_match(j),2));

left(j,2)=fs(1,match(rand\_match(j),2));

left(j,3)=1;

right(j,1)=ff(2,match(rand\_match(j),1));

right(j,2)=ff(1,match(rand\_match(j),1));

right(j,3)=1;

end

H=homography(left,right,n);

left=zeros(data\_s,3);

right=zeros(data\_s,3);

for j=1:data\_s

left(j,1)=fs(2,j);

left(j,2)=fs(1,j);

left(j,3)=1;

right(j,1)=ff(2,j);

right(j,2)=ff(1,j);

right(j,3)=1;

end

left\_m=map(left,H);

[err err2]=rms\_error\_2(right,left\_m);

err\_s=zeros(data\_s,2);

for j=1:data\_s

err\_s(j,1)=j;

err\_s(j,2)=err2(j);

end

sortrows(err\_s,2);

bestN=floor(N\*data\_s);

left=zeros(bestN,3);

right=zeros(bestN,3);

for j=1:bestN

left(j,1)=fs(2,match(err\_s(j,1),2));

left(j,2)=fs(1,match(err\_s(j,1),2));

left(j,3)=1;

right(j,1)=ff(2,match(err\_s(j,1),1));

right(j,2)=ff(1,match(err\_s(j,1),1));

right(j,3)=1;

end

H=homography(left,right,bestN);

left\_m=map(left,H);

[err err2]=rms\_error\_2(right,left\_m);

if err < best\_err

bestH=H;

best\_err=err;

end

end

left=zeros(data\_s,3);

right=zeros(data\_s,3);

for j=1:data\_s

left(j,1)=fs(2,j);

left(j,2)=fs(1,j);

left(j,3)=1;

right(j,1)=ff(2,j);

right(j,2)=ff(1,j);

right(j,3)=1;

end

left\_m=map(left,bestH);

% show mapping

end

# PART E

Couldn’t complete the stitching images in time. Will do it as part of Assignment 12.