Computer Vision I [EECE]
Project 2
Sharma, Shivam [sharma.s@husky.neu.edu]
Shah, Romil [shah.romil@husky.neu.edu]

Project 2

IMAGE MOSAICING

Computer Vision I [EECE]
Project 2
Sharma, Shivam [sharma.s@husky.neu.edu]
Shah, Romil [shah.romil@husky.neu.edu]

TABLE OF CONTENTS:

Computer Vision I [EECE]
Project 2
Sharma, Shivam [sharma.s@husky.neu.edu]
Shah, Romil [shah.romil@husky.neu.edu]

1. Abstract

A technique for mosaicing and image warping using Corner detection and Correlations is proposed in this project. The devised algorithm uses the Harris Corner Detection, Normalised Cross Correlation, Ransac and warping methods. The proposed algorithm is tested by variations in the critical parameters such as threshold values during Normalised Cross Correlation, Patch window size and other parameters. Corresponding results are analysed and the best result is reported upon.

2. Introduction

We start off with applying the Harris corner detector which makes use of the differential gradient along the x and y axis to detect the corners. Then we do Non maximum suppression, which is edge thinning technique, here we use it to get the sparse set of points.

Then we apply the Normalised cross correlation between two adjacent images to obtain more relevant and comparable corner points. NCC is usually used in image cases where the brightness of image is varying due to light exposures.

Then move on to calculate the Homography between the images using the Ransac algorithm. We then map the inliers (relevant) and outlier points in the original images. Then we select the relevant corner points and warp the second image with respect to the corresponding relevant corners in the previous image. We then mosaic the second image onto the previous image with respect to the relevant points.

3. Description of algorithm

The proposed algorithm is implemented by first applying the images with the Harris corner detector. This is applied with the use of two derivative filters along x and y, which computes the image derivatives along x and y respectively. These are used in the calculation of image corners. The sum of the derivatives along xx, yy and xy is calculated using Gaussian filter convolutions, where the filter's window size was decided after several experiments. Now the obtained sums are arranged to obtain the H matrix. We use this matrix to compute the Harris R function. We then move on to obtain the corner points through non-maximum suppression. We created our own function for non-maximum suppression to obtain the sparse set of corner features.

Now we compute the mapping between the two adjacent images using Normalised Cross Correlation. For this we consider the patch around the obtained corners from the previous step. We normally cross correlate the patches applied on the adjacent images. So each patch in Image 1 is correlated with each of the patches around the corners in the other image. Now we decide the threshold to filter out the lesser valued results between obtained cross correlated values. After applying the threshold we obtain a set of corresponding corner values in both the images which are high in correlation. These are the useful values to be used further.

Now we compute the Homography between the two images using the RANSAC algorithm. Here we select randomly 4 points in the image one and their corresponding numbered corner points in the next image. Now we calculate the distance between the corresponding two points between the 2 images. Now we set a threshold on the distance to distinguish between the outlier points and inlier points. If the calculated distance is less than the threshold then we store the points as inliers and if it is greater than the set threshold then we know that it is set of outlier points. This is another filter approach for getti9ng much more relevant points.

We now warp image based on the obtained inliers points. We select the best set of 4 inlier coordinates for each which covers the complete overlapping of the two images. We now calculate affine transformation matrix for the selected points. Based upon the affine transformation matrix we calculate the warped next image with respect the previous image. We now mosaic the initial image and the warped latter image with respect to the appropriate corner points.

4. Experiments

The devised algorithm gives decent results with just depicting appropriately warped with respect to the previous image. This was tested with various combinations of the parameters described above – Threshold values in NCC, Gaussian filter size and threshold values of distance.

1. **Threshold values in NCC –**
   The threshold values for the NCC were used as 0.6 , 0.7 and 0.8 . We obtained the best results with with the threshold value 0.8.

2. **Window size of the Gaussian Filter during corner detection–**
   Three values were experimented upon and checked which gave the better result. The values of size of Gaussian filter tested are – [5x5], [9x9], [15x15]. We obtained best results with the size of 5x5 Gaussian filter.

3. **Window size during patch and correlation –** Sizes analysed – 45,55 & 65. The size of 45x45 gave best results of all.

4. **Threshold values of distance –**
   The threshold values for distances used are 1, 2 & 3. The value 2 for threshold of distance gave better results for inliers.

5. Values of Parameters

1. **Standard Deviation Gaussian Filter during corner detection–** 0.5

2. **Scaling factor used-**

For the Dana Hall images the scaling factor used is 1, whereas for Office images the scaling factor used is 0.5.

3. **Harris function (R) –**

For Dana Hall 1:

```
H{1}  =
   1.0e+03 *
    3.4986    2.8353
    2.8353    3.4155
H{2}=
   1.0e+03 *
    2.9006    2.3498
    2.3498    2.8286
H{3}=
   1.0e+03 *
    3.0853    2.4253
    2.4253    2.8794
```

For Dana Hall 2:

```
H{1}=
   1.0e+03 *
    3.2485    2.5755
    2.5755    3.0839
H{2} =
   1.0e+03 *
    2.1503    1.7716
    1.7716    2.1988
H{3} =
  745.6634  375.3863
  375.3863  262.4124
```

4. **Homography Matrix Calculated–**

In two of the iterations for a good result we have received the value of homography matrix as -

In the Dana hall 1 image set between images 1 and 2:

$$H= \begin{matrix} 0.8064 & -0.0516 & -0.0004 \\ -0.0274 & 0.9185 & -0.0001 \\ 127{,}1409 & 11.7986 & 1.0000 \end{matrix}$$

In the Dana hall 1 between images 2 and 3:

$$H= \begin{matrix} 0.9054 & -0.0187 & -0.0002 \\ -0.0153 & 0.9598 & -0.0000 \\ 70.6664 & 2.2799 & 1.0000 \end{matrix}$$

In the Dana hall 2 image set between images 1 and 2:

$$H= \begin{matrix} 0.7502 & -0.0771 & -0.0004 \\ -0.0231 & 0.8887 & -0.0000 \\ 104.3801 & 19.8177 & 1.0000 \end{matrix}$$

In the Dana hall 2 between images 2 and 3:

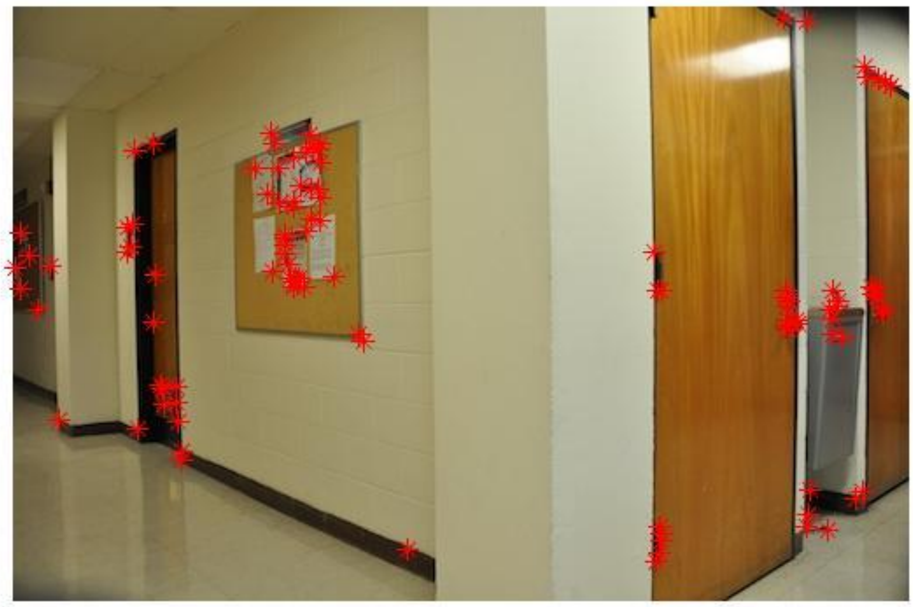
$$H= \begin{matrix} 0.6330 & -0.1144 & -0.0007 \\ -0.0137 & 0.8800 & -0.0000 \\ 148.4368 & 17.0229 & 1.0000 \end{matrix}$$

5. **Windows Matrix** – checked between 45, 55 and 65

6. Results:

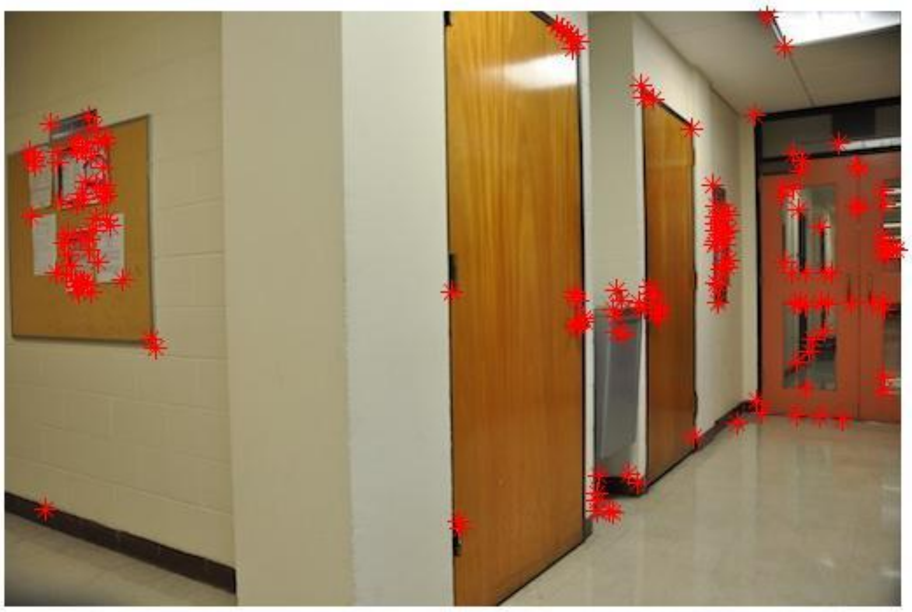
6.1. DanaHall1:

(a) Picture 1 after Harris Corner Detector

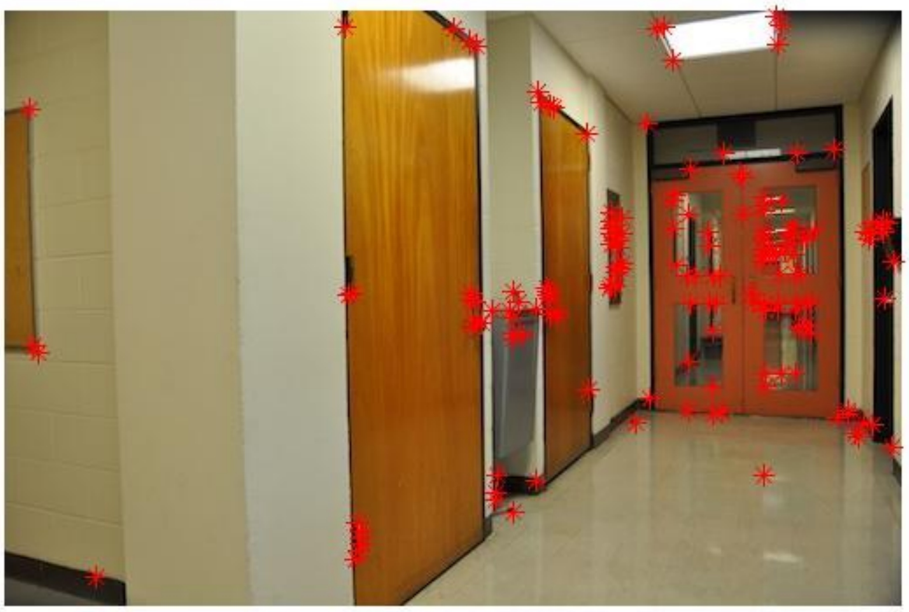


(b) Picture 2 after Harris Corner Detector

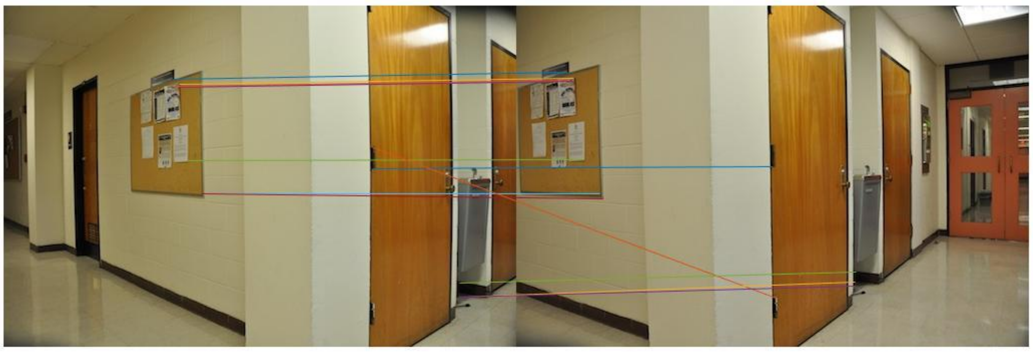
Sharma, Shivam [sharma.s@husky.neu.edu]
Shah, Romil [shah.romil@husky.neu.edu]



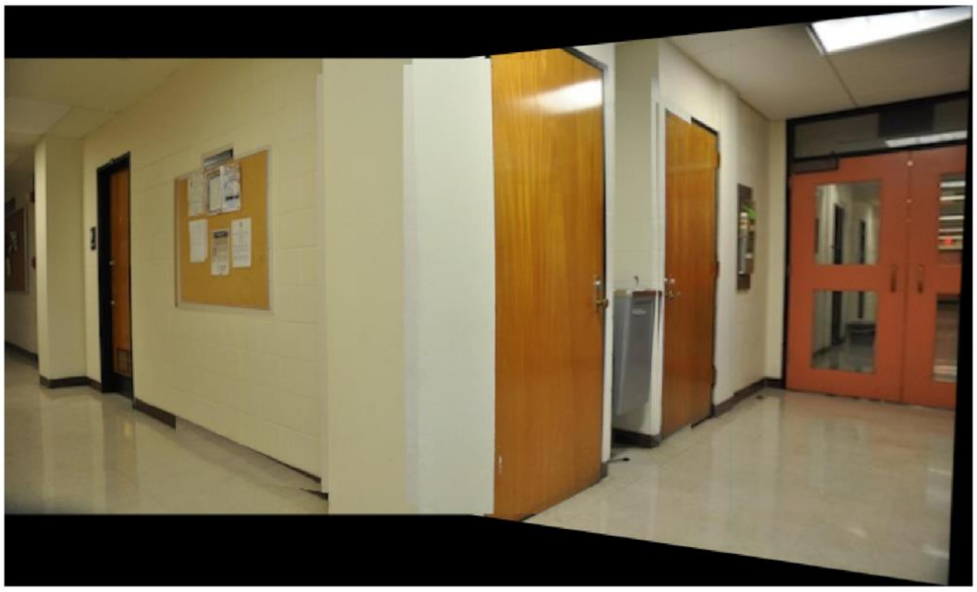
(c) Picture 3 after Harris Corner Detector



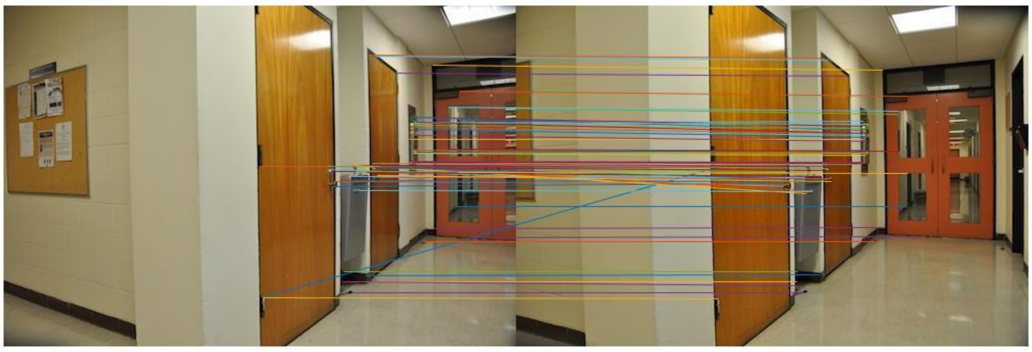
(d) Mapped images 1 & 2 with inlier and outlier distinction



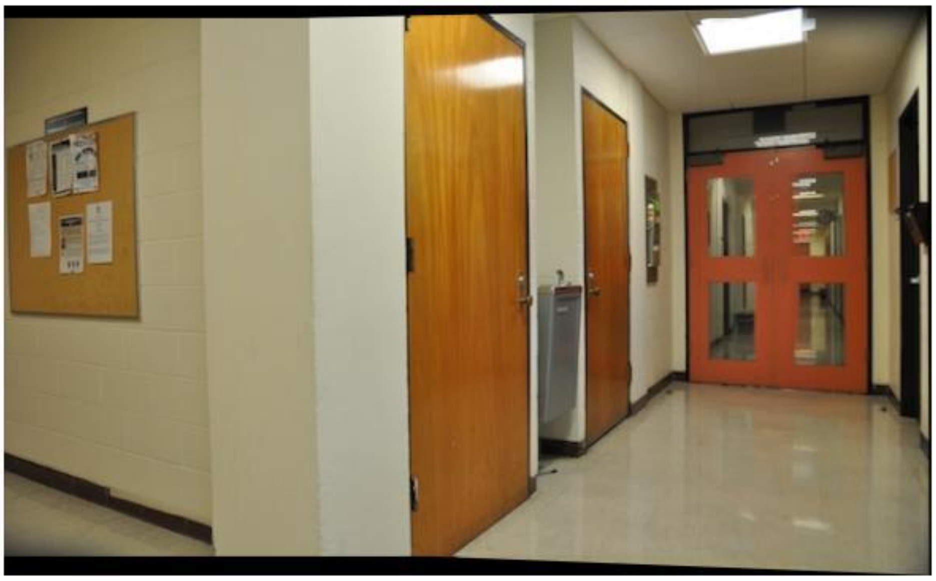
(e) Mosaiced Images 1 &2



(f) Mapped image 2 & 3 with inlier and outlier distinction



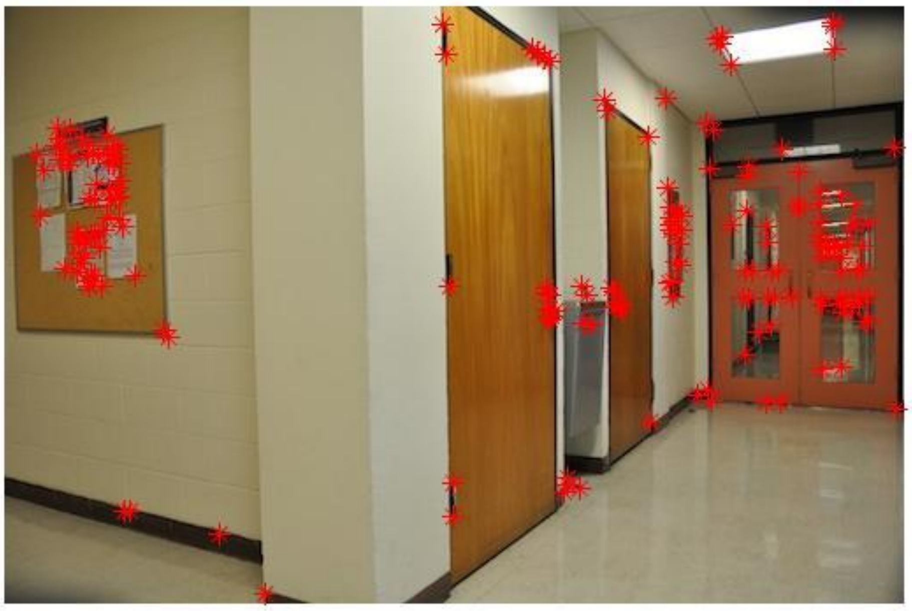
(g) Mosaiced images 1, 2 & 3



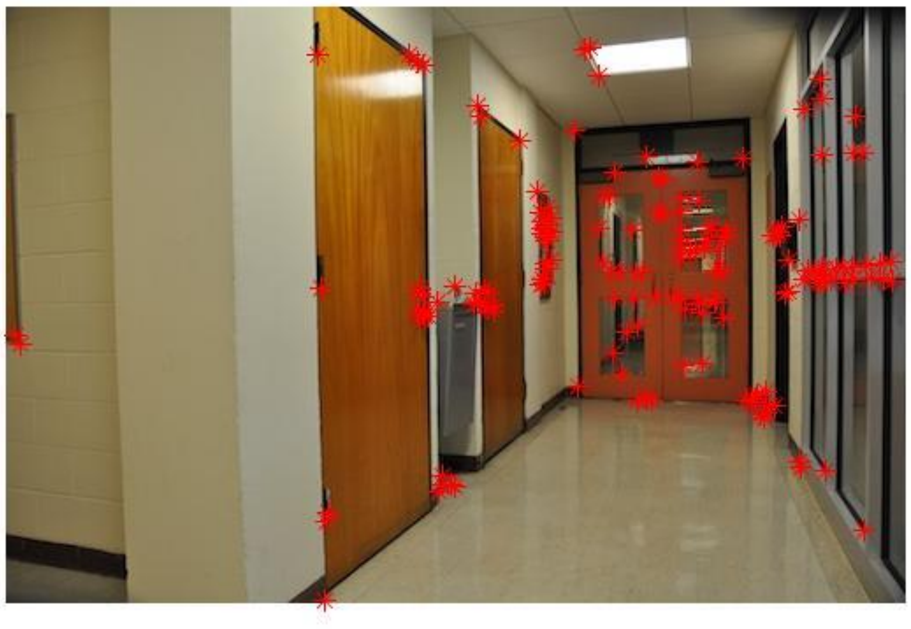
6.2. Danahall2

(a) Picture 1 after Harris Corner Detector

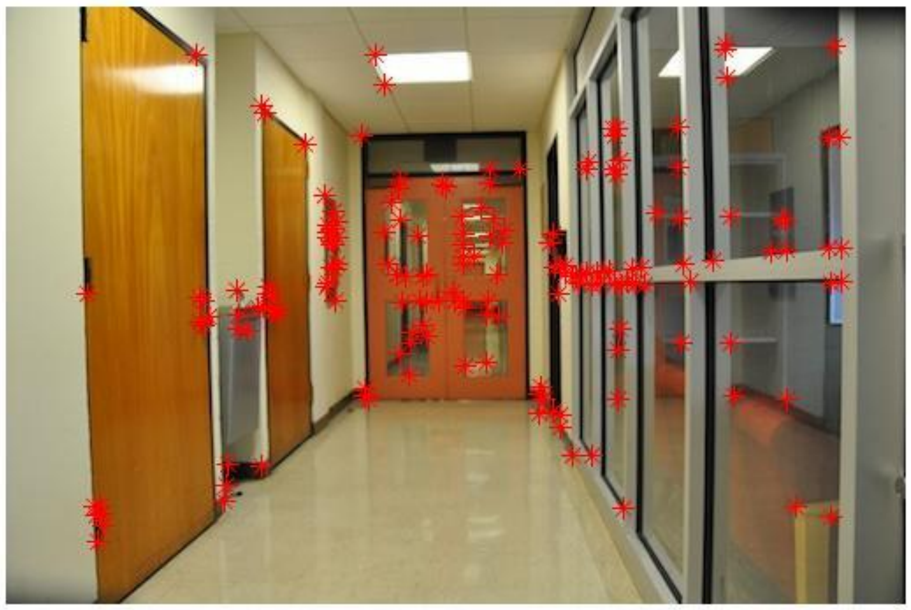
Sharma, Shivam [sharma.s@husky.neu.edu]
Shah, Romil [shah.romil@husky.neu.edu]



(b) Picture 2 after Harris Corner Detector

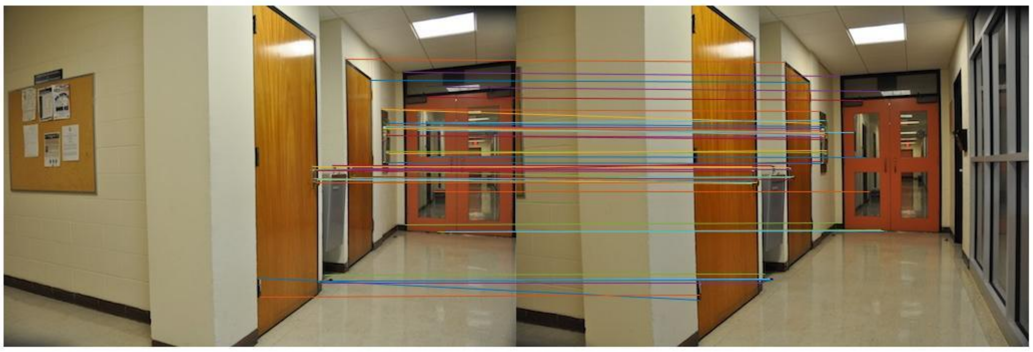


(c) Picture 3 after Harris Corner Detector

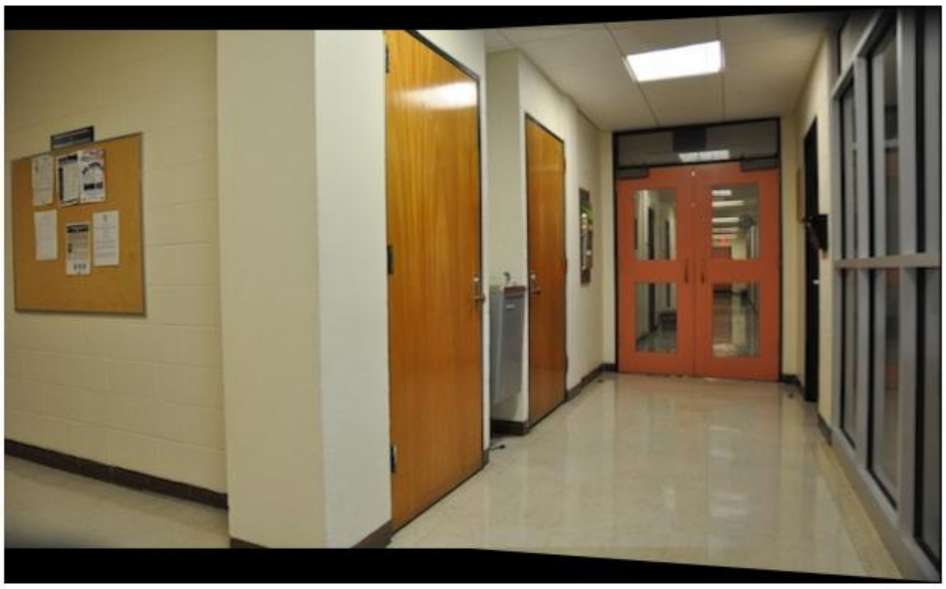


(d) Mapped images 1 & 2 with inlier and outlier distinction

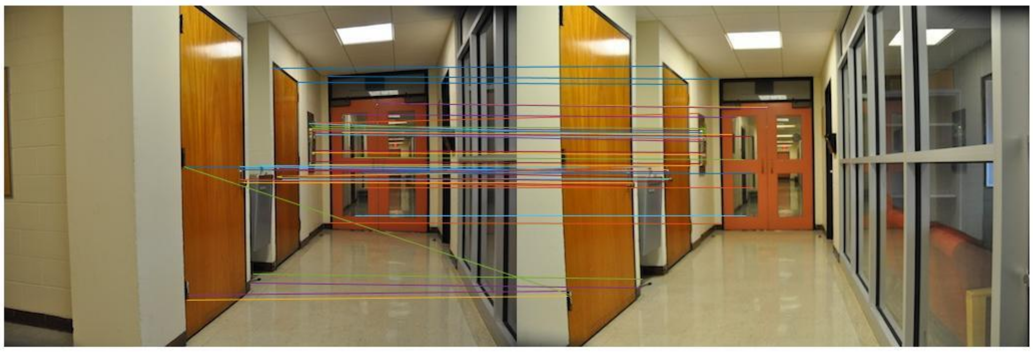
Sharma, Shivam [sharma.s@husky.neu.edu]
Shah, Romil [shah.romil@husky.neu.edu]



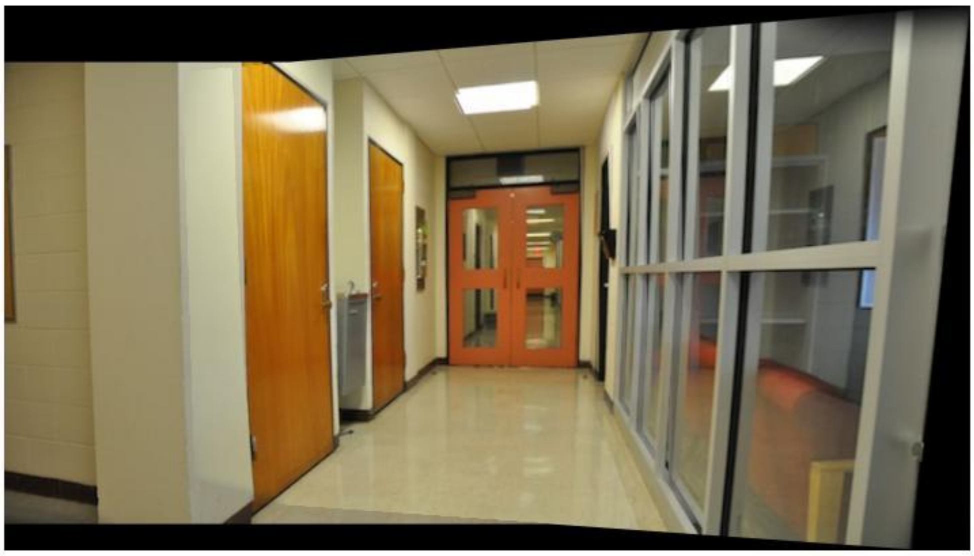
(e) Mosaiced Images 1 &2



(f) Mapped image 2 & 3 with inlier and outlier distinction

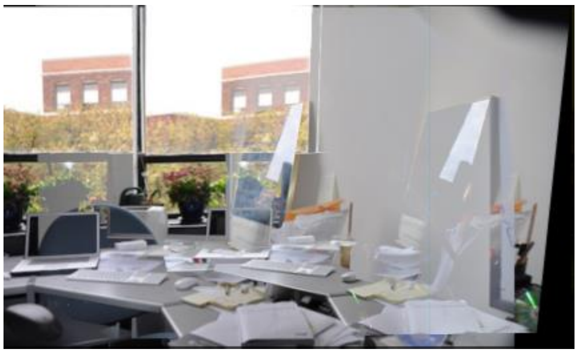


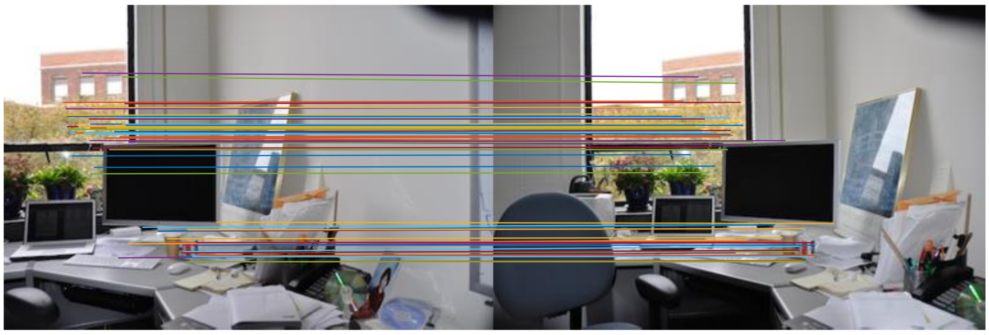
(g) Mosaiced images 1, 2 & 3



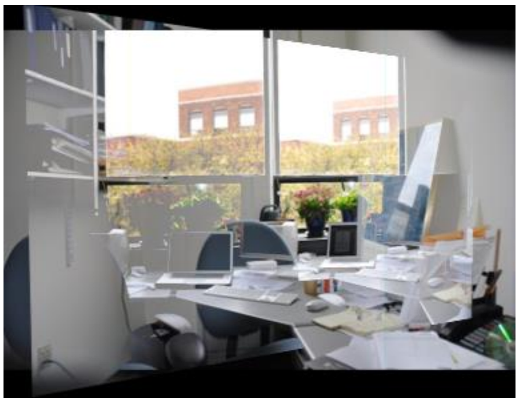
6.3. DanaOffice:

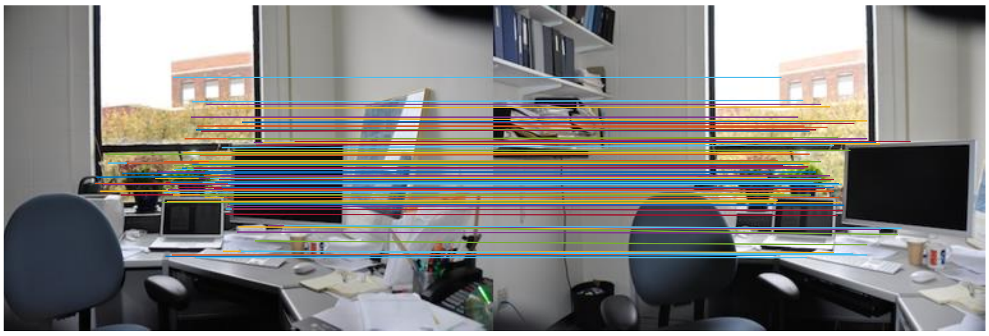
(a) Mosaicing picture 1& 2





(b) Mosaicing picture 2& 3





7. Conclusion

The images were analysed and applied upon with the mosaicing. The devised algorithm showed pretty accurate results. The areas which required warping were effectively tracked and applied warping operation upon. The mosaicing was performed to obtain a close to perfect panorama image with the given number of shots. The devised algorithm performed excellent for the task.

8. Future Scope

- Integration of this system for a continuous video mosaicing

9. Extra Credit Question

10. Appendix (MATLAB Code)

10.1. Main Code:

```
clear all
clc

imgList =
dir('C:\Users\rams1\Desktop\Spring2016\CV\Projects\Project_2\DanaHallWay2\*
.jpg');
imgNos = length(imgList);

dx = 0.25*[-1 0 1; -2 0 2; -1 0 1]; % The Mask
dy = dx';
sigma = 0.5;
gauss = fspecial('gaussian',[5,5],sigma);

scale = 1;

for i=1:imgNos
```

Sharma, Shivam [sharma.s@husky.neu.edu]
Shah, Romil [shah.romil@husky.neu.edu]

```
cd('C:\Users\rams1\Desktop\Spring2016\CV\Projects\Project_2\DanaHallWay2\')
    filename = imgList(i).name;
    OrigI{i} = imresize(imread(filename),scale);
    I{i} = rgb2gray(OrigI{i});
    img = I{i};
    Ix2 = conv2(img(:,:), dx, 'same').^2; Sx2{i} = conv2(Ix2,gauss,
'same');
    Iy2 = conv2(img(:,:), dy, 'same').^2; Sy2{i} = conv2(Iy2,gauss,
'same');
    Ixy = conv2(img(:,:), dx, 'same').*conv2(img(:,:), dy, 'same'); Sxy{i}
= conv2(Ixy,gauss, 'same');
    [H{i},R{i},C{i}] = harrisResponse(Sx2{i},Sy2{i},Sxy{i});
    cd('C:\Users\rams1\Desktop\Spring2016\CV\Projects\Project_2\');
%     figure(i)
    Cout{i} = nonMaxSup(C{i},Sxy{i});
%     imshow(OrigI{i});
%     hold on;
%     plot(Cout{i}(:,1),Cout{i}(:,2),'r*');
%     hold off;
end

[XY,OFFXY,stitch1] = NCC(I{1}, I{2}, Cout{1}, Cout{2}, OrigI{1}, OrigI{2});

[originliers, inliers, origoutliers, outliers] = INLIERS(OFFXY,[XY(:,1)-
512,XY(:,2)],400);
inliers(:,1) = inliers(:,1) + 512;
outliers(:,1) = outliers(:,1) + 512;
figure;
imshow(stitch1);
hold on;
plot([originliers(:,1) inliers(:,1)]',[originliers(:,2) inliers(:,2)]','-
Og',...
    [origoutliers(:,1) outliers(:,1)]',[origoutliers(:,2)
outliers(:,2)]','-Or');
hold off;

figure;
imshow(stitch1);
hold on;
for i=1:length(inliers)
    hold on;
    plot([originliers(i,1) inliers(i,1)]',[originliers(i,2)
inliers(i,2)]');
end
hold off;

inliers(:,1) = inliers(:,1) - 512;
outliers(:,1) = outliers(:,1) - 512;

generateOutput(OrigI{1},OrigI{2},inliers,originliers,OFFXY,XY);
```

10.2. Harris Corner Detector:

```
function [H,R,C] = harrisResponse(Ixx, Iyy, Ixy)

Ixx = double(Ixx);
Iyy = double(Iyy);
Ixy = double(Ixy);

[sx, sy] = size(Ixx);
R = zeros(sy, sx);
for i = 1:sx,
    for j = 1:sy,
        H = [Ixx(i,j), Ixy(i,j);
            Ixy(i,j), Iyy(i,j)];
        R(j,i) = det(H) - 0.04*(trace(H).^2);
    end
end
[Cidx,Cidy] = find(R > 10000);
C = [Cidx,Cidy];
end
```

10.3 Non Maximum Suppression:

```
function Cout = nonMaxSup(Cin,img)

Cy = Cin(:,1);
Cx = Cin(:,2);

L = length(Cin);
[m,n] = size(img);
count = 0;

for i=1:L
    if Cx(i)>1 & Cy(i)>1 & Cx(i)<m & Cy(i)<n
        if img(Cx(i),Cy(i))>img(Cx(i)-1,Cy(i)) &
img(Cx(i),Cy(i))>img(Cx(i)+1,Cy(i)) & img(Cx(i),Cy(i))>img(Cx(i),Cy(i)-1) &
img(Cx(i),Cy(i))>img(Cx(i),Cy(i)+1) &...
                img(Cx(i),Cy(i))>img(Cx(i)-1,Cy(i)-1) &
img(Cx(i),Cy(i))>img(Cx(i)+1,Cy(i)+1) & img(Cx(i),Cy(i))>img(Cx(i)+1,Cy(i)-
1) & img(Cx(i),Cy(i))>img(Cx(i)-1,Cy(i)+1)
            count = count + 1;
            Cout(count,:) = Cin(i,:);
        end
    end
end
```

10.4 Normalised Cross Correlation NCC :

```
function [XY,OFFXY,newImg] = NCC(img1, img2, Cout1, Cout2, colimg1,
colimg2)

[n,m] = size(img1);
L1 = length(Cout1);
L2 = length(Cout2);
xpeak = 1;
ypeak = 1;
cnt = 0;
for i=1:L2
    if Cout2(i,1)>45 & Cout2(i,1)<m-45 & Cout2(i,2)>45 & Cout2(i,2)<n-45
        patch2 = img2(Cout2(i,2)-45:Cout2(i,2)+45,Cout2(i,1)-
45:Cout2(i,1)+45);
        c = normxcorr2(patch2,img1);
        [ypeak, xpeak] = find(c==max(c(:)));
        yoffSet = ypeak-size(patch2,1);
        xoffSet = xpeak-size(patch2,2);
        for j=1:L1
            if Cout1(j,1)>45 & Cout1(j,1)<m-45 & Cout1(j,2)>45 &
Cout1(j,2)<n-45
                patch1 = img1(Cout1(j,2)-45:Cout1(j,2)+45,Cout1(j,1)-
45:Cout1(j,1)+45);
                if corr2(patch1,patch2)>0.7
                    cnt = cnt + 1;
                    XY(cnt,1) = Cout2(i,1)+m;
                    XY(cnt,2) = Cout2(i,2);
                    OFFXY(cnt,1) = Cout1(j,1);
                    OFFXY(cnt,2) = Cout1(j,2);
                end
            end
        end
    end
end

newImg = [colimg1, colimg2];
figure;
imshow(newImg);
hold on;
plot([OFFXY(:,1) XY(:,1)]',[OFFXY(:,2) XY(:,2)]','-Og');
hold off;
```

10.5 Detecting Inliers:

```
function  [originliers, inliers, origoutliers, outliers] =
INLIERS(xyoff,xy,iter)

n = length(xyoff);
inliers = [];
originliers = [];
hmat = xyoff\xy;
XYC = (hmat*xyoff')';
```

```
% mndist = sqrt((XYC(:,1)-xyoff(:,1)).^2 + (XYC(:,2)-xyoff(:,2)).^2);
%
for i=1:iter
    sno = randperm(n,4);
    loc1 = xyoff(sno,:)';
    loc2 = xy(sno,:)';
    if length(unique(loc2','rows'))==4
        H = loc2/loc1;
        xycap = round(H*loc1);
        dist = sqrt((loc2(1,:)-xycap(1,:)).^2 + (loc2(2,:)-xycap(2,:)).^2);
        thresh = 2;
        [~,idy] = find(dist<thresh);
        inliers = [inliers, loc2(:,idy)];
        originliers = [originliers, loc1(:,idy)];
    end
end

mat = unique([inliers', originliers'],'rows');
inliers = mat(:,1:2);
originliers = mat(:,3:4);

[a,b,~] = unique(inliers,'rows');
inliers = a;
originliers = originliers(b,:);

% for i=1:length(inliers)
%     for j=i:length(inliers)
%         if ((inliers(i,1)==inliers(j,1)) & (inliers(i,2)==inliers(j,2)))
| ((originliers(i,1)==originliers(j,1)) &
(originliers(i,2)==originliers(j,2)))
%             inliers(j,:) = [];
%             originliers(i,:) = [];
%         end
%     end
% end

temp = ismember(xy,inliers);
pos = find(temp(:,1)==0 & temp(:,2)==0);
outliers = xy(pos,:);
origoutliers = xyoff(pos,:);

mat = unique([outliers, origoutliers],'rows');
outliers = mat(:,1:2);
origoutliers = mat(:,3:4);

end
```

10 .6 Generating Output:

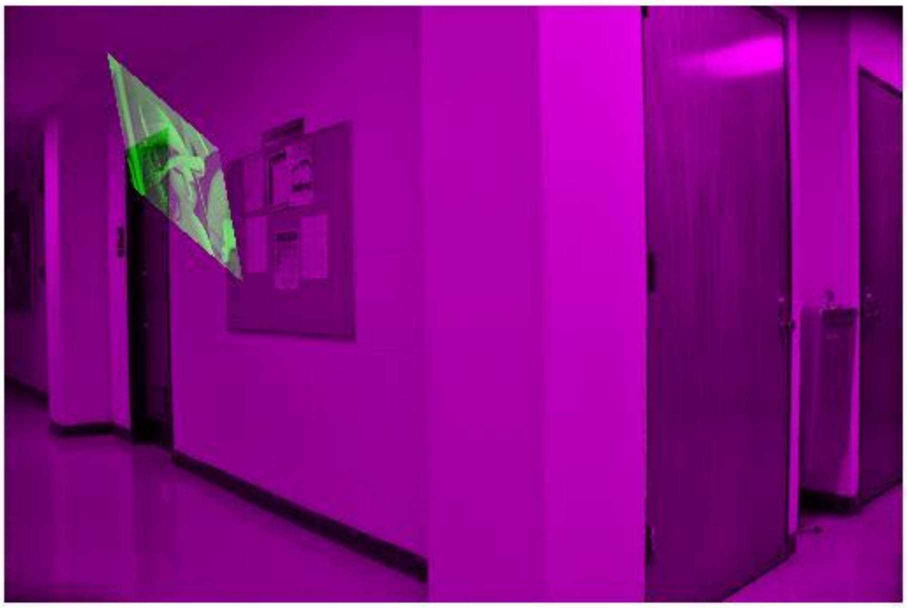
```
function generateOutput(I1,I2,inliers,originliers,OFFXY,XY)

quad11 = [];
quad21 = [];
quad12 = [];
quad22 = [];
```

Sharma, Shivam [sharma.s@husky.neu.edu]
Shah, Romil [shah.romil@husky.neu.edu]

```
[m,~] = size(I1);
hmat = originliers\inliers;
% imshow(I1); [X1,Y1]=ginput(4);
% imshow(I2); [X2,Y2]=ginput(4);
% rnd = ([rnd(1) rnd(2) rnd(4) rnd(3)])
%
% X1 = originliers(rnd,1);
% X2 = inliers(rnd,1);
% Y1 = originliers(rnd,2);
% Y2 = inliers(rnd,2);
% X1 =
[originliers(idx1,1);originliers(idx2,1);originliers(idx3,1);originliers(id
x4,1)];
% Y1 =
[originliers(idx1,2);originliers(idx2,2);originliers(idx3,2);originliers(id
x4,2)];
% X2 = [inliers(idx1,1);inliers(idx2,1);inliers(idx3,1);inliers(idx4,1)];
% Y2 = [inliers(idx1,2);inliers(idx2,2);inliers(idx3,2);inliers(idx4,2)];

for i=1:length(inliers)
    if inliers(i,2)<m/2
        quad11 = [quad11;originliers(i,:)];
        quad21 = [quad21;inliers(i,:)];
    else
        quad12 = [quad12;originliers(i,:)];
        quad22 = [quad22;inliers(i,:)];
    end
end

% quad11 = quad11';
% quad21 = quad21';
% quad12 = quad12';
% quad22 = quad22';

[~,id11] = min(quad11(:,1));
[~,id12] = max(quad11(:,1));
[~,id21] = min(quad12(:,1));
[~,id22] = max(quad12(:,1));

rnd1 = [id11,id12];
rnd2 = [id22,id21];
X1 = [quad11(rnd1,1);quad12(rnd2,1)];
X2 = [quad21(rnd1,1);quad22(rnd2,1)];
Y1 = [quad11(rnd1,2);quad12(rnd2,2)];
Y2 = [quad21(rnd1,2);quad22(rnd2,2)];

imgpts1 = [X1 Y1];
imgpts2 = [X2 Y2];
imgptscap2 = imgpts1*hmat;
dist = (sqrt((imgpts2(:,1)-imgptscap2(:,1)).^2 + (imgpts2(:,2)-
imgptscap2(:,2)).^2))

if mean(dist)<10
    figure;
    plot([X1],[Y1],'-Og',[X2],[Y2],'-Or')
    T = maketform('projective',[X2 Y2],[X1 Y1]);
    h = T.tdata.T
    [~,xdataim2t,ydataim2t]=imtransform(I2,T);

    xdataout=[min(1,xdataim2t(1)) max(size(I1,2),xdataim2t(2))];
    ydataout=[min(1,ydataim2t(1)) max(size(I1,1),ydataim2t(2))];

    im2t=imtransform(I2,T,'XData',xdataout,'YData',ydataout);

im1t=imtransform(I1,maketform('affine',eye(3)),'XData',xdataout,'YData',yda
taout);

    ims=im1t/2+im2t/2;
    imd=uint8(abs(double(im1t)-double(im2t)));
    ims=max(im1t,im2t);

    [m,n]=size(ims);
    a1 = n/4;
    a2 = 3*n/4;
    filtimg = imfilter(ims(:,a1:1:a2),fspecial('gaussian',[1,3]),0.5);
    ims(:,a1:1:a2) = filtimg;
    figure;
    imshow(ims);


else
    [originliers, inliers, ~, ~] = INLIERS(OFFXY,[XY(:,1)-
512,XY(:,2)],200);
    generateOutput(I1,I2,inliers,originliers,OFFXY,XY);
end
```

11. Appendix Extra Credit (MATLAB Code)

Sharma, Shivam [sharma.s@husky.neu.edu]
Shah, Romil [shah.romil@husky.neu.edu]



```
clear all
clc

imgList =
dir('C:\Users\rams1\Desktop\Spring2016\CV\Projects\Pr
oject_2\DanaHallWay1\*.jpg');
imgNos = length(imgList);

dx = 0.25*[-1 0 1; -2 0 2; -1 0 1]; % The Mask
dy = dx';
sigma = 0.5;
gauss = fspecial('gaussian',[5,5],sigma);

scale = 1;

cd('C:\Users\rams1\Desktop\Spring2016\CV\Projects\Pro
ject_2\DanaHallWay1\')
filename = imgList(1).name;
OrigI = imresize(imread(filename),scale);
I = rgb2gray(OrigI);
img = I;
```

```
Ix2 = conv2(img(:,:), dx, 'same').^2; Sx2 =
conv2(Ix2,gauss, 'same');
Iy2 = conv2(img(:,:), dy, 'same').^2; Sy2 =
conv2(Iy2,gauss, 'same');
Ixy = conv2(img(:,:), dx, 'same').*conv2(img(:,:),
dy, 'same'); Sxy = conv2(Ixy,gauss, 'same');
[H,R,C] = harrisResponse(Sx2,Sy2,Sxy);
cd('C:\Users\rams1\Desktop\Spring2016\CV\Projects\Pro
ject_2\');
Cout= nonMaxSup(C,Sxy);
I2 = double(rgb2gray(OrigI));
I1 = double(rgb2gray(imread('lena.png')));
[um,un,~]=size(I2);

imshow(OrigI); [X2, Y2] = ginput(4);
X1 = [1;un;un;1];
Y1 = [1;1;um;um];

h = [X1 Y1]\[X2 Y2];
h = [h [0;0]; 0 0 1];
[xi, yi] = meshgrid(1:512, 1:340);
h = [1 0 X2(1);0 1 Y2(1); 0 0 1] * (h) * [1 0 -X2(2);
0 1 Y2(2); 0 0 1];
h = inv(h); %TAKE INVERSE FOR USE WITH INTERP2
xx =
(h(1,1)*xi+h(1,2)*yi+h(1,3))./(h(3,1)*xi+h(3,2)*yi+h(
3,3));
yy =
(h(2,1)*xi+h(2,2)*yi+h(2,3))./(h(3,1)*xi+h(3,2)*yi+h(
3,3));
foo = uint8(interp2(I1,xx,yy)/255);
figure(1); imshow(imfuse(foo,I2))
```