Muhammabad Basit Iqbal Awan
20100153
Assignment 4
Using 3 late days

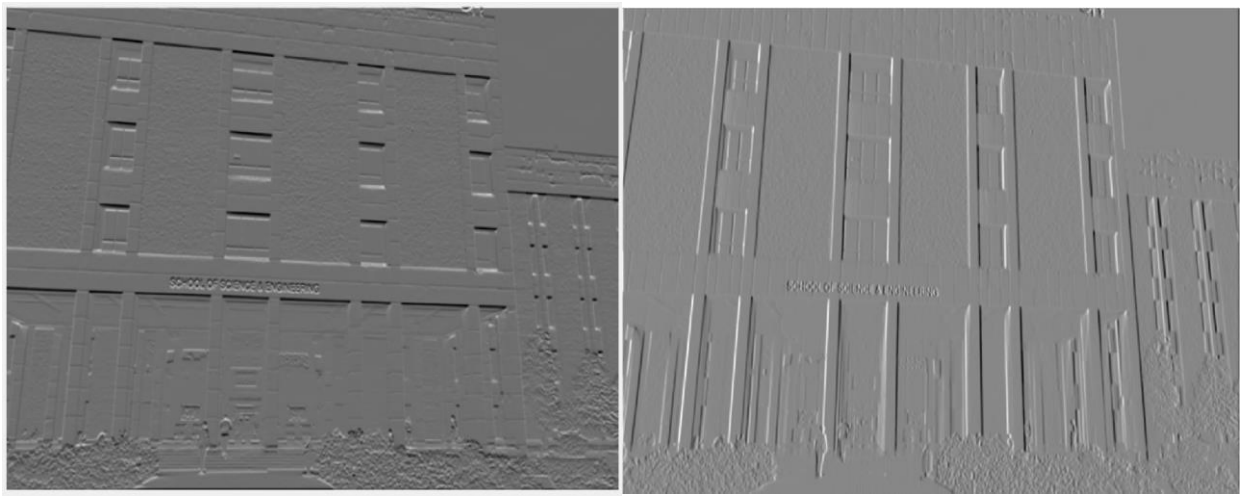
Finding gradients along y and x axis:

So I placed all the code for this part in 'Corner.m' file. The script takes an image value and converts it to grayscale to get better results. I then made a gaussian filter using the technique sir described in the tutorial and then took it's derivatives and applied those to the image.

```
function Corners = cornerFinder(imageName)

I= imageName;
A = rgb2gray(I);
%generating derivative of gaussian filter
sigma=1;
max_len= round(sqrt(-2*sigma^.2*log(0.1)));
[y,x]=meshgrid(-max_len:max_len,-max_len:max_len);
g= exp(-(x.^2 + y.^2)./2*sigma.^2);
dx= (-x./sigma^2).*g;
dy= (-y./sigma^2).*g;
%applying filter on A
Ix = conv2(A,dx,'same');
Iy = conv2(A,dy,'same');
```

Gradients along x axis and y axis respectably came out to be like this:



Finding KLT corners and applying NMS to find corners:

After getting the values for Ix and Iy, I started to take the covariance of a grid size of 17x17, and moved it around each pixel. For each pixel I then took the egienvalues of it's covarian and if they were greater than 5000 added the corresponding lowest eigenvalue in a Eigen Matrix. I used 5000 because it was giving me better results. The code is as shown below:

```
grid = 8;
eigenMat = zeros(size(I,1),size(I,2));
for i=grid+1:size(Ix,1)-grid
    for j=grid+1:size(Ix,2)-grid
        STX = i-grid;
        EDX = i+grid;
        STY = j-grid;
        EDY = j+grid;
        XMat = Ix(STX:EDX,STY:EDY);
        YMat = Iy(STX:EDX,STY:EDY);
        Covar = [sum(sum(XMat.^2)),sum(sum(XMat.*YMat));
                    sum(sum(XMat.*YMat)),sum(sum(YMat.^2))];
        Covar = Covar/(grid*2)^2;
        eigenVals = eig(Covar);
        if eigenVals(1)>5000
             eigenMat(i,j)= eigenVals(1);
        end


    end
end
```

After then I applied NMS, by taking the highest gradient storing its cordinates in variables 'Corners' and making all of the grid around it zero to avoid clustering of points. I used a bigger grid in this to avoid a cluster being formed around the letter head of SSE. So the grid is 5 times bigger than the previous one. I do this 100 times to ensure I get most points. The code is as follows:

```
Corners = [];
grid = grid*5;
for i = 1:100
    [~,ind]=max(eigenMat(:));
    [row,col]= ind2sub(size(eigenMat),ind);
    eigenMat(max(1,row-grid):row+grid,max(1,col-grid):col+grid) = 0;
    Corners = [Corners;row,col];
    end

end
```

I then give the corner values back to the function to return as an argument.

Getting points from function and choosing appropriate points for better matching:

I got the corners back in ‘imageStitch.m’ file and then took the 30 right most corners of the left image and the 30 left most corners of the right to get better matching and then displayed them as shown below:

The code I used to do this is as follows :

```
function final = imageStitch(image1,image2)
Corners1 = cornerFinder(image1);
Corners2 = cornerFinder(image2);
temp= sortrows(Corners1,2,'descend');
Corners1 = temp(1:30,:);
temp= sortrows(Corners2,2);
start =1;
for start=1:100
    if temp(start)~=1
        break;
    end
end
Corners2 = temp(min(start,71):min(start+29,100),:);
figure()
I1=image1;
imshow(I1);
hold on;
plot(Corners1(:,2), Corners1(:,1), 'r.', 'LineWidth', 2, 'MarkerSize', 15);
figure()

I2=image2;
imshow(I2);
hold on;
plot(Corners2(:,2), Corners2(:,1), 'b.', 'LineWidth', 2, 'MarkerSize', 15);
```

Match Finding using Hungarian algorithm:

I then converted the image into grayscale to get better matches and then made a cost matrix for Hungarian algorithm. The code I used to do this is as follows.

```
I2 = rgb2gray(I2);
costMat = zeros(30);
grid=8;
for i = 1:30
    for j = 1:30
        x = Corners1(i,1);
        y = Corners1(i,2);
        if(x==1||y==1)
            costMat(i,j)=99999999999999999;
            continue;
        end
        STX = max(x-grid,1);
        EDX = x+grid;
        STY = max(y-grid,1);
        EDY = y+grid;
        f_patch = double(I1(STX:EDX,STY:EDY));
        x = Corners2(j,1);
        y = Corners2(j,2);
        STX = max(x-grid,1);
        EDX = x+grid;
        STY = max(y-grid,1);
        EDY = y+grid;
        s_patch = double(I2(STX:EDX,STY:EDY));
        if(x==1||y==1)
            costMat(i,j)=0;
            continue;
        end
        costMat(i,j) = norm(f_patch-s_patch,2);
    end
end
```

I got the matches and swapped the corners according to their matches.

Applying ransac to find homogrpahy and stitching images together:

I then applied ransac to the matches and got about homography H and 10 inliers as shown below:

```
[H,inliers] = ransacfithomography(Corners1', Corners2', 0.001);
H=inv(H);
I1 = image1;
I2 = image2;

inliers =

     7     8    10    11    12    18    20    21    22    24
```

I then found the dimension of the canvas by applying the inverse of the homography to the 4 corners of image2 and combining them with the corners of image1 and getting max values for both. I had to convert the matrix of the canvas into uint8 so it represented an image. The code is as shown below:

```
Left1 = H*[1;1;1];
Left1 = Left1./Left1(3);
Left2 = H*[454;1;1];
Left2 = Left2./Left2(3);
right1 = H*[454;807;1];
right1 = right1./right1(3);
right2 = H*[1;807;1];
right2 = right2./right2(3);

corners = [1,1;1,size(I1,2);size(I1,1),1;size(I1,1),size(I1,2);
    Left1(1),Left1(2);Left2(1),Left2(2);right1(1),right1(2);right2(1),right2(2)];
dim = max(corners);
final = zeros(round(dim(1)),round(dim(2)),3);
final= uint8(final);
```

After which I added 70% of image 1 into it's appropriate position and then started to find homographs from all the missing region. If it was present in image 2 I placed the value of image 2 in there as shown below :

```
final(1:size(I1,1),1:round(size(I1,2)*0.7),1:3)= I1(:,1:round(size(I1,2)*0.7),:);
H = inv(H);
for i= 1:size(final,1)
    for j=1:size(final,2)
        if final(i,j)==0
            new = H*[i;j;1];
            new = new./new(3);
            new= round(new);
            if (new(1)<1 || new(2)<1 || new(1)>size(I2,1) || new(2)>size(I2,2))
                continue;
            end
            final(i,j,:) = I2(new(1),new(2),:);
        end
    end
end
```

The end result I got was this image , which honestly looks pretty nice:

Part c (Partially attempted

I returned the imageData to Assignment4.m where I have displayed it. Had to resize the images from there as I want to make a panorama of 5 images.

I couldn’t find the five images for the panaroma but I was thinking of recursively stitching the images together to form a whole panaroma.

I tried to stitch images onto another image on the left but ransac was giving a problem which I couldn’t solve as I couldn’t change the maximum number of trials.