# Introduction to Computer Vision

## Assignment – 4

-Yashwanth Telekula

## Summary:

In this assignment, we try to find homography matrix of size 3X3 which can convert the co-ordinates of an image in one plane to different co-ordinates in a different plane.

In case1, homography matrix or h-matrix has 8 unknowns where the $9^{th}$ unknown is 1. To find the 8 unknows we need 8 or more equations. To get a minimum of 8 equations we need atleast 4 coordinate points in each image to compare.

**In order to run the code, run the Runcode.m file and manipulate the image reading lines in the Runcode.m file to test the project on different images. The 4 different H-matrix values are displayed in console and stored in h_values.mat file so that it can be used for later.**

## OBSERVATIONS:

⇨ If the number of points are less than 4 i.e. (N<4), then some values in H-matrix become absolute Zero to satisfy the 6 or less equations which decreases the accuracy.

⇨ If the number of points are greater than 4 i.e. (N>4), then the values obtained in H-matrix are less accurate because these H values need to satisfy all these 10 or more equations equally correct.

⇨ I think the performance decreases if the value of N>4. So, N=4 will be the best option and these 4 points on each side should be chosen with most accuracy to get the best-looking results.

⇨ If the more than 2 points are in a straight line, then this condition is similar to N<4 case and some of the values in the H-matrix will be absolute zero which is not an optimal solution that we want.

## Code Explanation:

➢ Runcode.m

➢ Precondition.m

➢ Normalization.m


## Runcode:

This is the main function that has to be run to start the process.  The function reads two images and asks the user to plot N points on one image at a time. The value of N can be changed at the start of the code. The default value of N is 4(assigned by me).  We use ginput(N) function to collect the plotted points.

### CASE 1: IF $H_{33}$ = 1

⇨ After point collection, 'A' matrix of size 2N X 8 is created just as mentioned in the slides.

⇨ Then we use the Pseudo Inverse function with A and b matrix to find the H-matrix.

⇨ Then we convert the 8X1 H-matrix into 3X3 H-matrix by keeping the last element ($H_{33}$) as 1.

### CASE 2: IF ||H|| = 1

⇨ After the point collection, we construct the 'A' Matrix of size 2N X 9 just as mentioned in the slides.

⇨ Then we use the eigen function [V, D] = eig (A' * A);

⇨ The first-row values of the V matrix are considered as the values of H-matrix.

⇨ The H-matrix of size 9X1 is reshaped to 3X3 matrix.


## Precondition:

This function is run at the end of the main function to generate the both H-matrix values after normalization of the co-ordinates.

⇨ The Center of mass(COM) co-ordinates is calculated by finding the mean of the x and y co-ordinates.

⇨ The Center of mass is shifted to Origin and other points are shifted according to this point.

⇨ Then the scale value is calculated by finding the sum of distances of each point to origin and dividing sqrt(2) with the sum of distance value.

⇨ Then each co-ordinate is multiplied with this scale value.

⇨ Then the same procedure is followed just as mentioned as above with the modified co-ordinates.

```
function [h_precondition,h_new_precondition  ] = precondition( x,y,u,v )
```

## Normalization:

This function takes the whole X and Y co-ordinates and converts them into normalized co-ordinates and returns them back.

⇨ The co-ordinates are normalized as we mentioned in the precondition definition.
⇨ This function is called twice independently for both the image co-ordinates to get the normalized co-ordinates of both images.

```
function [x,y] = Normalization(x,y);
```