horizontal line

**Computer Vision**

Image Mosaics

**30th May 2021**

| **Name** | **ID** |
| --- | --- |
| **Jamal Eldin Ahmed Khalaf** | **15** |
| **Yomna Gamal El-Din Mahmoud** | **63** |

# Problem Statement

implement an image stitcher that uses image warping and homographies to automatically create an image mosaic. We will focus on the case where we have two input images that should form the mosaic, where we wrap one image into the plane of the second image and display the combined views. This problem will give some practice manipulating homogeneous coordinates, computing homography matrices, and performing image warps. For simplicity, we will specify corresponding pairs of points manually using mouse clicks.

# Getting Correspondences

Using plotly.express library to plot the image then all we have to do is to hover on the pixel and the coordinates with the pixel value will appear.

We obtained 4 pixels in each image.  
For image1: [[380,290],[310,490],[110,500],[50,175]]  
For image2: [[815,315],[765,530],[570,540],[565,240]]



# Compute the Homography Parameters

## Compute H Matrix

Function get\_homography(p1, p2) takes the points from the first image and the points in the second image and calculate the H3x3 using lstsq from numpy.linalg

We have n points so we can set up a solution using a system of linear equations Ax = b where the 8 unknowns of H are stacked into an 8-vector x , the 2n-vector b contains image points from one view, and the 2n × 8 matrix A is filled appropriately

So the b vector = [xi, yi]T where xi & yi are points in p2  
For the A matrix it will contain 2 equations for each point

1. [xi, yi, 1, 0, 0, 0, -x`i\*x, -x`i\*y]
2. [0, 0, 0, xi, yi, 1, -y`i\*x, -y`i\*y]

Where x`i & y`i are the points in p2 and xi & yi are points in p1  
So now we can solve this equation system by using lstsq function in numpy.linalg this will produce H as a vector of length 8 we append 1 to it as we were solving for H3,3 = 1 then we reshape the H vector to matrix 3x3

## Verify H Matrix

We can use the built-in function from the open-cv library called findHomography() . It takes 2 vectors containing the points in image1 and image2. Our H matrix was similar to the H matrix that came from the built-in function.

# Warping Between Image Planes

It contains 2 steps: first get the wrapped image shape then wrap the image.

## Get Wrapped Image Shape

Function called get\_warped\_image\_shape(img, H) takes the image we want to wrap and the H matrix we calculated.

First we get the height & width of the image then get the boundary box of the image [ [0, 0], [h-1, 0], [0, w-1], [h-1, w-1] ] then we calculate the projection of these points in the second image by multiplying them by H matrix P` = H\*P after that we get the minimum and maximum for x & y the wrapped image size will have a height of Xmax - Xmin - 1 and width Ymax - Ymin - 1 and the height will be shifted by -Xmin and the width will be shifted by -Ymin then we have these 4 values new height and width and the shift in height & width.

## Make Warping

Function make\_warping(img, H) takes the image we want to warp and the H matrix calculated before.

First we get the warped image shape then make a numpy array of this shape then take every coordinate x & y in the image and apply the transformation using H matrix P` = H \* P

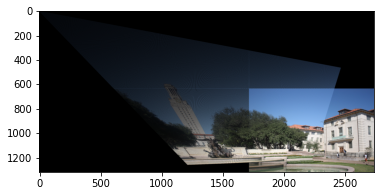
Finally we sum the shift on the transformed coordinates and map them to pixel values of x,y in the image.

# Create The Output Mosaic

Function create\_mosaic(wraped\_image, second\_view, h\_shift, w\_shift) takes the warped image with height shift and width shift and the second view of the image.

We calculate the height and width of the mosaic and then put the pixel values of the second view and warped image in the mosaic image.

# Output Sample



## Note

The output can be improved if the selected points were chosen with more accuracy.