# ENPM662

# Project Two


SURIYA SURESH

# TABLE OF CONTENTS

# LIST OF ILLUSTRATIONS

*Chapter 1*

# PROBLEM ONE

For this a video was given for which the homography had to determined and the rotation and the translation of the camera had to calculated for every frame of the video.

## 1.1 Pipeline Followed

The general overview of the steps that were undertaken are :

1. Image Processing
2. Hough Transform and Finding Corners
3. Finding Homography Matrix
4. Plotting Rotation and Translation

## 1.2 Image Processing

The video was read frame by frame under a while loop in the python program.Each frame was converted to grayscale, a gaussian blur applied to reduce the effects of the key and noise for edge detection.The processed frame is then passed onto the Canny Edge Detector which is the input for the Hough Transform.
The Hough Transform is done on the image which is now a binary image with only the outputs of the deges detected.

## 1.3 Hough Transform

The Hough Transform is a technique used to detect shapes in any binary image[1]. It works on converting Cartesian coordinates $(x, y)$ into Hough Space. Usually the equation of a line is represented as

$$y = mx + c$$

but for Hough Space the coordinates are taken as $(m, b)$.So this means a line in Cartesian becomes a point and a point becomes a line in Hough space.However if a line is vertical , $m$ can go till infinity, so we take the polar coordinate system with

the equation of a line being defined as

$$\rho = x \cos\theta + y \sin\theta$$

with the values $(\rho, \theta)$ being the Hough space for Hough Transform.For this specific problem, the values of $\rho$ vary from positive to negative diagonal length of the image and $\theta$ ranges from 0 to $\pi$ in steps of one radian.For every pair of $(\rho, \theta)$ for each $(x, y)$ represents a line passing through them in the $x - y$ plane.

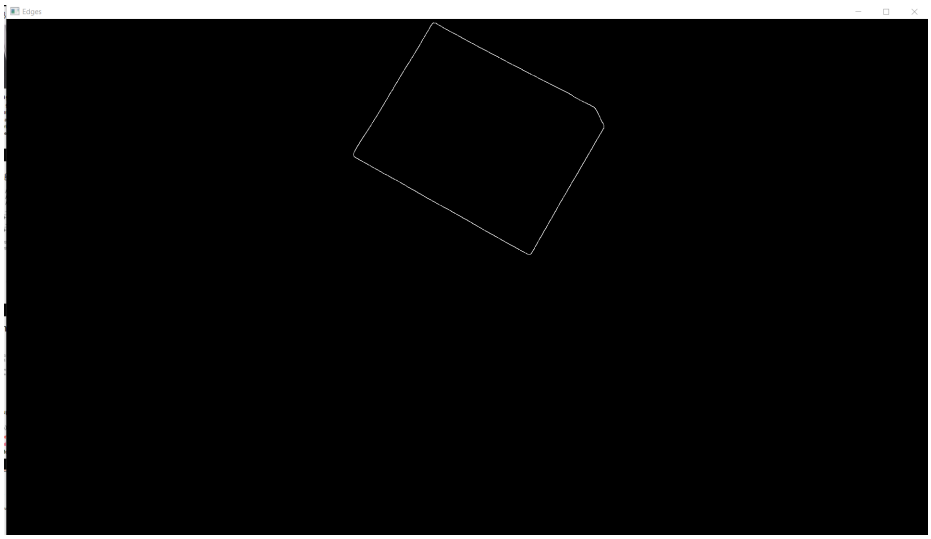

Figure 1.1: Canny Edge Detection Output

The Hough Transform operates on the edge points of $(x, y)$ which is given by the Canny Edge Detector in this pipeline.

A accumulator matrix of the size being the number of values of $\rho$ and columns being the number of values of $\theta$,which in this case is 180.For every value of $(\rho, \theta)$, the value is updated in the matrix by one.So every value of $(x, y)$ , for all values of $\theta$, the value of $\rho$ is calculated and the same value is updated in the accumulator.The updated value represents the number of intersections that each line has in the $(x, y)$ plane.

After updating the accumulator matrix, the max values are found, the max 6 values are taken, the intersecting lines are eliminated and only the four lines are taken and intersection points are found.The intersection between the lines represent the corners of the paper.

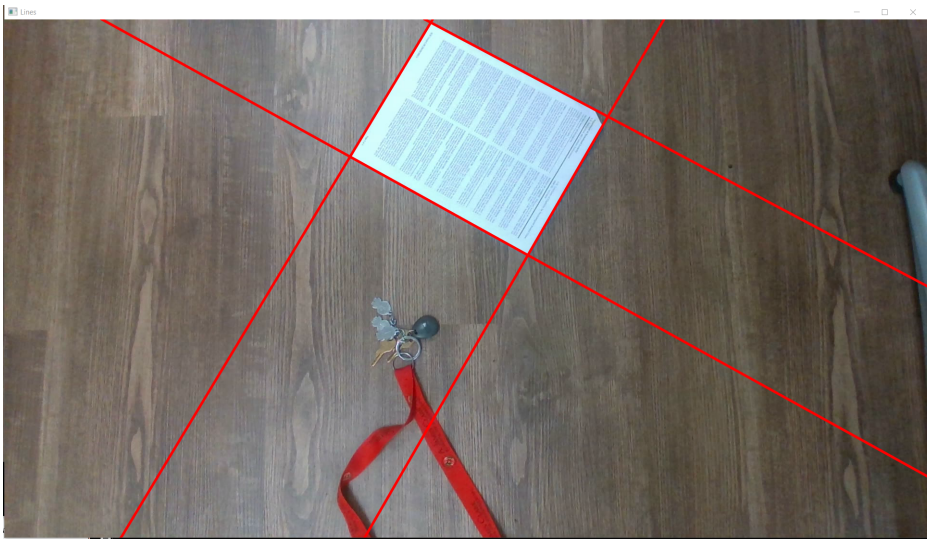

Figure 1.2: Hough Lines of Image

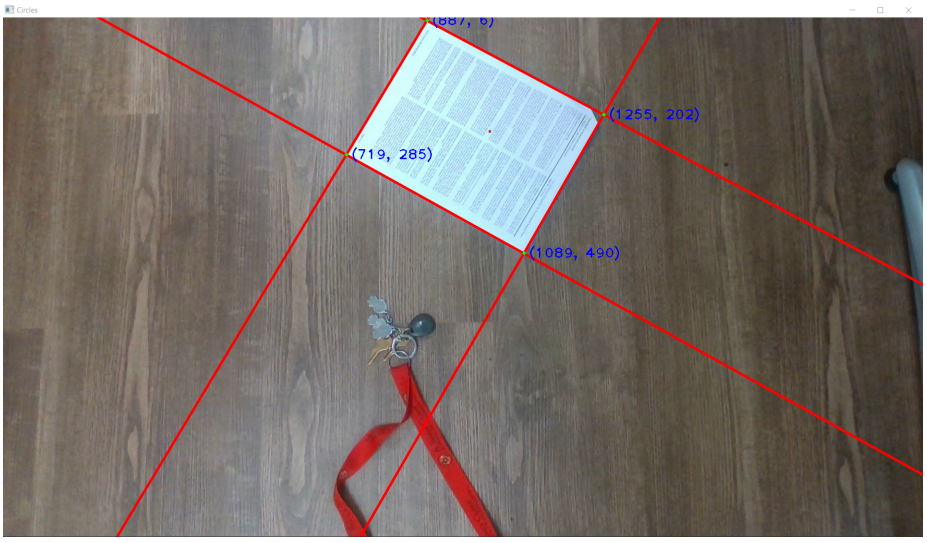


Figure 1.3: Detected Corners of the Image

## 1.4 Finding Homography Matrix

The homography matrix is found taking the following equations

$$x' = Hx \tag{1.1}$$

We take two sets of points $(x', y')$ and $(x, y)$ which are points on the camera sensor and in the world frame. The Z axis is assumed to be 0. The H matrix is a 3x3 matrix and the points are 3x1 vectors.The equation can be written as shown in fig 1.4: The equation for determining the homography matrix has been changed to: This is the

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Figure 1.4: Homography Equation

$$\underset{\mathbf{2n \times 9}}{\mathbf{A}} \quad \underset{\mathbf{9}}{\mathbf{h}} \quad \underset{\mathbf{2n}}{\mathbf{0}}$$

Figure 1.5: Equations to find Homography Matrix

equation that is used in the program and after the homography matrix is found.The rotation and translation vectors are found.The equations in Fig 1.5 can be solved using numpy.SVD and the last column of the output of SVD is the homography matrix, to ensure that the last element of the homography matrix is one, the entire matrix is divided by that value. The last element is not considered in fig 1.5 in the X matrix of the equation $AX = B$ and hence is taken as a unit value.

### 1.5 Plotting Rotation and Translation

After the Homography matrix is found, we know that

$$x' = PX \tag{1.2}$$

which then is considered as

$$x' = K[r_1 r_2 t] \tag{1.3}$$

$$= K[r_1 r_2 t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1.4}$$

After eliminating $Z = 0$,we get:

$$= K[r_1 r_2 t] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{1.5}$$

We bring H into the equation:

$$= H \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1.6}$$

$H$ can be written as:

$$H = [r_1 r_2 t] \tag{1.7}$$

$$K^{-1} H = \lambda [r_1 r_2 t] \tag{1.8}$$

$r1\&r2$ are unit vectors and $r3$ is the cross product of $r1 \text{and} r2$. Finally the equation can be written as :

$$P = K[r_1 r_2 (r12) t] \tag{1.9}$$

This is the equation used to solve to find the rotation and translation vectors.After the L.H.S is found, the norm of first and second columns are taken to find $\lambda$ , which is the average of the values from the two columns and the whole matrix is divided by it. The first column and second columns of the LHS are now the vectors of $r1 \text{and} r2$.The last column is the translation vector. The rotation and translation vectors are plotted.

The rotation vectors have a magnitude of one.The rotation does not change much as seen from the plots.The movement along the Z axis is the most while movement along X and Y axes can be seen.The angular rotation is relatively less as the lines are lmost straight ,except for noise and estimation errors of calculation.

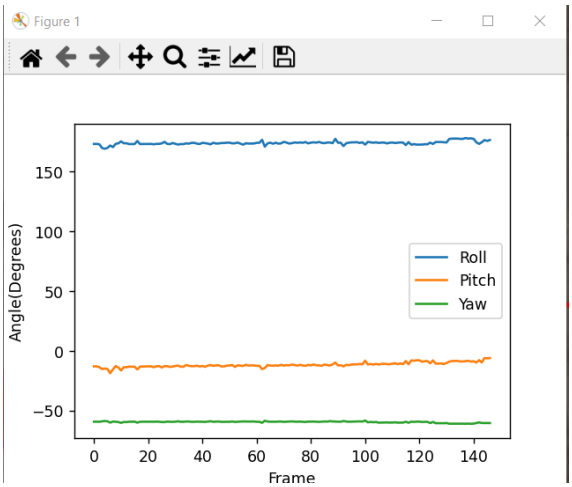


Figure 1.6: Plots of Rotation of the Camera

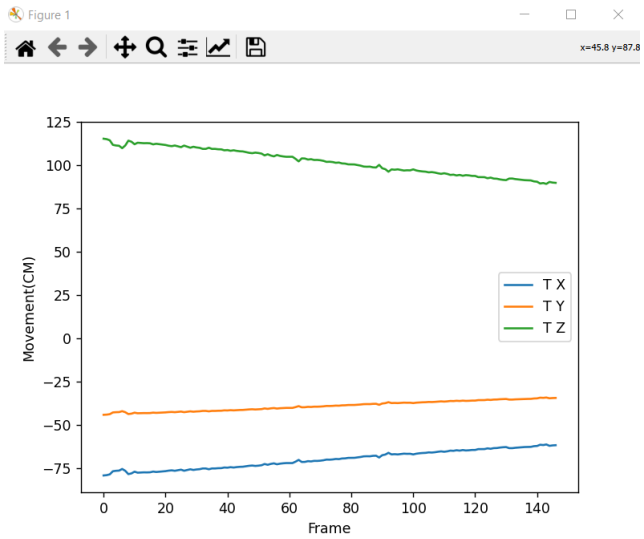


Figure 1.7: Plots of Translation of the Camera

## 1.6 Problems Faced and Solutions Done

This section describes the problems faced while solving problem one.

- Removing Parallel Lines or Lines that are close to each other-When the Hough Transform output was drawn on the image, there were multiple lines on each side of the paper that were almost parallel.This affects finding the corners.This was fixed by finding the distance between lines and removing them if they were too close.

- Code taking too much time to run- The use of many for loops resulted in the

code running for a long time for each frame.Vectorizing the code and using numpy packages reduced the time taken by a large margin.

- Finding origin and plotting points: Due to the origin of the image being on the top left corner ,while calculating the origin of the paper in the camera sensor and the real world, there were mistakes in the calculation. Careful checking of values fixed the issue.

*Chapter 2*

# PROBLEM 2

Problem 2 involves the stitching of 4 images into a single panoramic image.We begin by reading the images and follow the pipeline as illustrated below.

## 2.1 Pipeline Followed

The following steps were followed to solve problem two.

- 1.Loading Image
- 2.Feature Extraction and Feature Matching
- 3.Homography
- 4.Wrap Perspective
- 5.Stitching Images

## 2.2 Image Loading

The images were loaded and resized to make them smaller.They were converted to gray-scale for easier feature detection.

## 2.3 Feature Extraction and Feature Matching

For extracting features from images, the SIFT (Scale Invariant Feature Transform)[2] algorithm was used to detect features in images.It is known to be comparatively faster while being accurate compared to other feature detection algorithms generally.A SIFT object is created and the function detect and compute is used to extract key-points and descriptors from the image. This has to be done for every image to be processed.

After the features are extracted, the Brute Force Matching was used with match function from openCV[3].The output is a matches matrix which is a Matcher Object of the type DMatch object which holds values of distance(distance between descriptors,lower vlue is better),train index(Index of the descriptor in train descriptors), query index (Index of the descriptor in query descriptors) and image distance(index of the train image)[3].Sample code from the Docs was referred for the python script.

After the features are matched, the matches are filtered based on distance for better results.

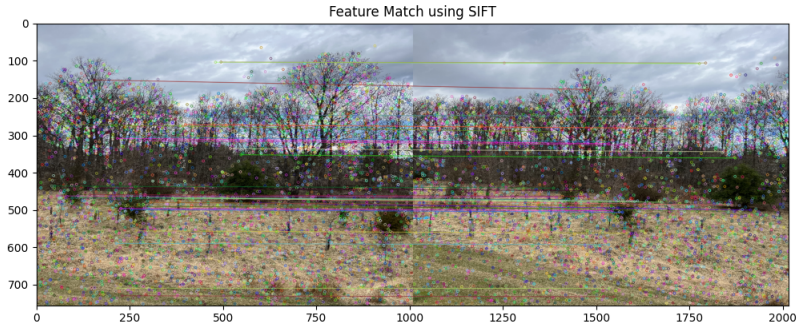


Figure 2.1: Images Features Using SIFT

## 2.4 Homography

After feature extraction and feature matching using builtin OpenCV functions, the homography matrix was found for the feature matched points on two images.RANSAC fitting was used to determine the best possible Homography matrix that is able to rotate the images so that while stitching images, the overlapping features connect and form a whole image.

The error function used for RANSAC is the distance between the points of the second image and the points obtained after the product of the Homography matrix and points of the first image.The threshold is taken in terms of pixel distance and is in the magnitude of $10^1$. The functions for determining homography and RANSAC were taken from the previous section and older assignments.

## 2.5 Wrap Perspective

After the Homography matrix is obtained after RANSAC fitting , the image is warped to change perspective for stitching images together. The second image , along with the homography matrix found is passed to the cv2.WrapPerspective function with the size of the new image being the combined widths of the image to be stitched and the same height.

### RANSAC algorithm

The general alogirthim for RANSAC is as follows:

- 1. A subset of the points is chosen, usually the size which is least needed to find the values of a model.

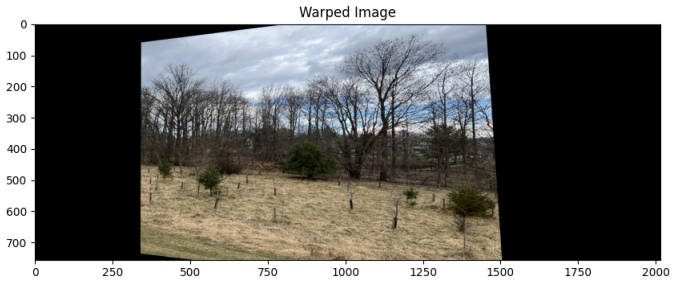


Figure 2.2: Wrapped Image

- 2. A model is fit to that subset.
- 3. Using a certain parameter, the points that are close to the model with respect to that parameter are found and the rest is beyond a certain threshold are rejected as outliers.
- 4. This process is done N times and the best model with the least outliers is chosen as the model for the set of points.

Here the model was the homography matrix which was fitted for a sample of 4 points.Afterwards , the error for the found homography matrix is found and rejected or used after the alogortuhim is finished running.The number of iterations chosen here was 100. The number of iterations can be found by:

$$N = log(1p)/log(1(1e)s) \tag{2.1}$$

Where:
$N$ is the total number of iterations
$p$ is the probability of getting a good sample
$s$ is the number of points in a subset
$e$ is the probability a point is an outlier.
For fitting, the values taken were s=3, p=0.99, and e=0.5. RANSAC handles noise and outliers better than least squares and total least squares.However, the same matrix cannot be used for similar images as the features may change.Running this for too many iterations can also result in improper stitching of images.There is a fine line between under-fitting, fitting it aptly, and over-fitting the Homography matrix.

## 2.6 Stitching Image

After the wrapped image is obtained , the image is stitched to form a larger image.Here, stitching was done part by part for 4 images and finally the whole image was stitched to gather to form a whole image.This process involves stitching together the width of two images as the canvas and making the height the same for all stitched images.The whole width of the stitched images is not equal to the 4 into the width of a single image.

## 2.7 Output of Stitched Images

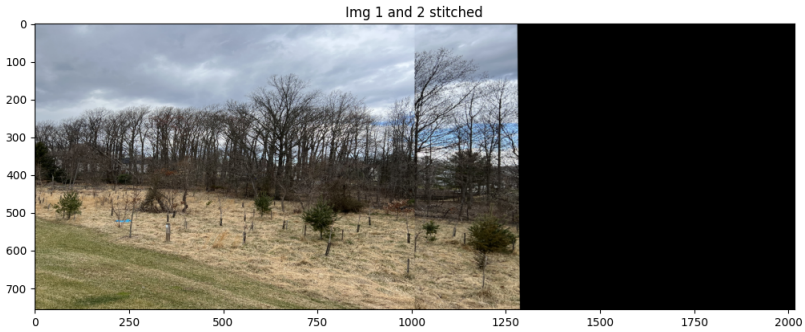


Figure 2.3: Image 1 and 2 stitched

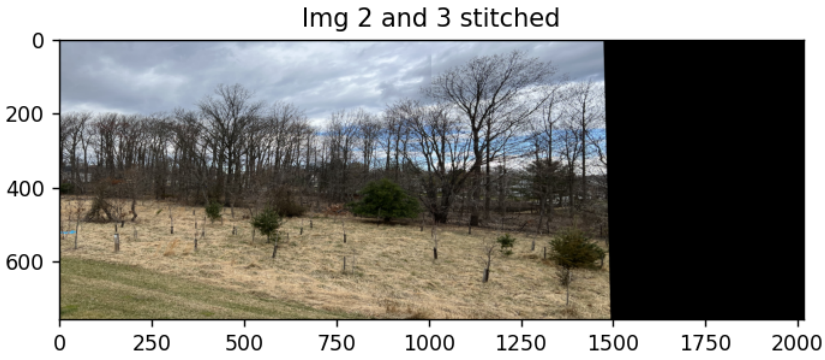


Figure 2.4: Image 2 and 3 stitched

The entire image did not stitch properly.

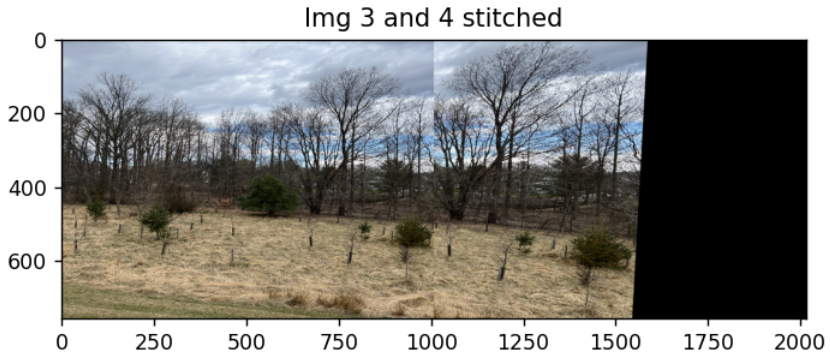


Figure 2.5: Image 3 and 4 stitched

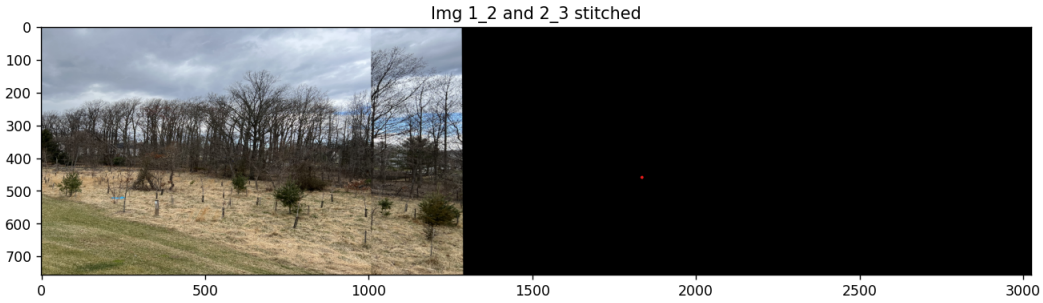


Figure 2.6: Image 1-2 and 3-4 stitched

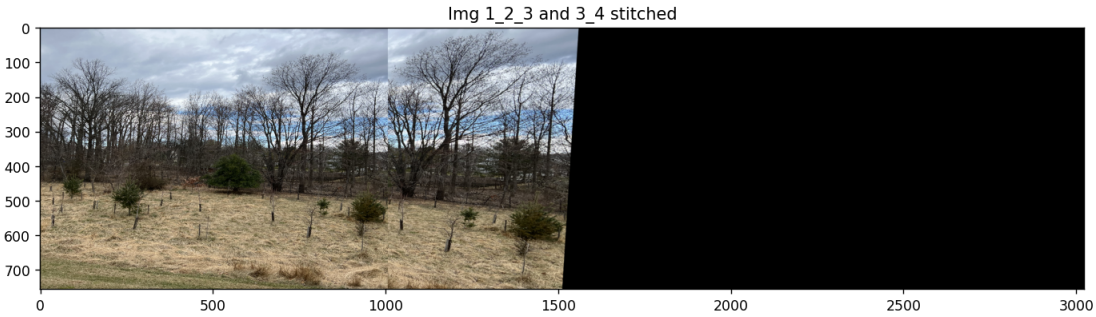


Figure 2.7: Image wholly stitched

## 2.8 Problems Faced and Solutions Implemented

This section describes the problems faced and solutions :

- Improper Homography Parameters-Due to the random nature of the RANSAC algorithim and the lower number of sample points, the hmography matrix calculated could cause the image to warp and result in improper stitching. Increasing the sample points and increasing the iterations of the RANSAC sampling reduced the error in the matrix.
- Stitching the images- Stitching the 4 images one by one resulted in poor joining of images.The images had to stitched in a specific order so that the end image was of better fit.
- Large size of images- The raw images were very large and were reduced in size to allow for easier view of the output and reduce the time needed for calculations.
- Janky stitching of Images- Images were not stitched perfectly with each other.This is a result of RANSAC and hompgraphy matrix errors.It has been overcome to a extend by increasing number of iterations and number of sample points for randomness but still can be improved.

# REFERENCES

[1] Opencv Docs. *Hough Line*. https://docs.opencv.org/3.4/d3/de6/tutorial_js_houghlines.html. [Online]. 2022.

[2] OPEN-CV docs. *SIFTt*. https://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html. [Online]. 2022.

[3] Opencv Docs. *BF*. https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html. [Online]. 2022.