GTU

**DEPARTMENT OF COMPUTER ENGINEERING**

**CSE 463 – Spring 2022**

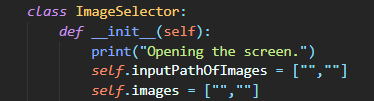
**HOMEWORK 3  
REPORT**

SÜLEYMAN GÖLBOL 1801042656

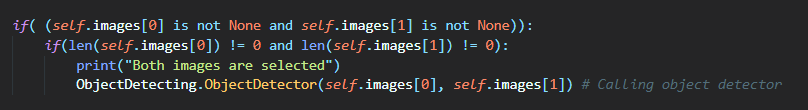
1. **PROBLEM SOLUTION APPROACH AND ALGORITHMS**

My problem that I have encountered in application was about image selecting.

I didn’t want to use global variables to store the value of input path. So I made inside of my ImageSelecting module as a class. In that class, I needed to hold values of some variables inside constructor . To initialize them, I holded them as string list.



But after that, problem was checking if both images are selected and it wasn’t easy. Because if no images are selected, images[i] values were None. But when one of them selected both of them changes. So put another if condition to check if(self.images[0] == “”). But this had an other problem. “FutureWarning: elementwise comparison failed; returning scalar, but in the future will perform elementwise comparison”. The warning was because of problem between python and numpy arrays. Python thinks Scalar or a Numpy tkinks ndarray. So, instead I checked the lengths.



**Because of example images are already are rectified, I didn’t need to rectify them again.**

For sift, I used “SIFT\_create()” method from OpenCV.

After this sift variable created, I converted images to grayscale for simplicity then used detectAndCompute(gsImage,None) to get the keypoints and descriptors.

Then I matched with cv.BFMatcher().match(descriptor1, descriptor2)which is brute force match.to my Homography module

Then I sorted with sorted method by the distance to get the best matches.



At the end I drawed the best 100 matches with drawMatches().

**3 ) TESTS AND RESULTS**

Running python code with python main.py

# 

# Running with Jupyter Notebook

# Homograph.py and PointSaving.py should be in the same folder with notebook file

# nbook\_main.ipynb

# 

# *Menu Screen to select 2 images*

# 

# *Example output from 2 different images*

# 

# A person wearing a garment Description automatically generated with low confidence

# *Another example*

# 