Assignment 4 – SIFT Method

Prepared by

**Dylan Gartin**

**Course name** -Computer Vision (SP 2022)

**Course number** - CAP4410.01

**Date of submission** – 4/5/2022

\* Note we will be formatting the reports now with style, Due to major issues that occur even still with trying to implement the template. If you want something added to this template let us know.

# Table Of Contents

[Table Of Contents 2](#_Toc100502141)

[Introduction 2](#_Toc100502142)

[Figure 1: Images used with the Shift algorithm (Right Base, Left Orientation example) 3](#_Toc100502143)

[Shift Methodology – Detection / key points localization 3](#_Toc100502144)

[Figure 2: Sift Algorithm Key Point Detection Code 4](#_Toc100502145)

[Figure 3: Sift Algorithm Key Point Detection Results 5](#_Toc100502146)

[Orientation assignment with feature matching 5](#_Toc100502147)

[Figure 4: Sift Algorithm Orientation Assignment Code 6](#_Toc100502148)

[Figure 5: Sift Algorithm Orientation Assignment Results 7](#_Toc100502149)

[Challenges 8](#_Toc100502150)

[Conclusion 8](#_Toc100502151)

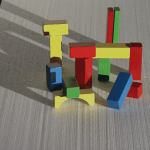
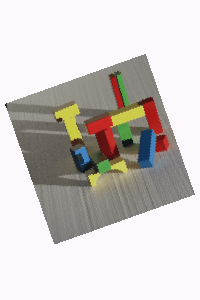
[References - 9](#_Toc100502152)

# Introduction

The Goal of this assignment is to use the SIFT methodology in order to detect features and points of interest in the application in order to get key points that are part of an object to detect it in various angles. This is going to be used for feature matching in order to detect the various features of objects. This allows it to still detect the same object even if it is a different scale, rotation and orientation. This is done with three main features of the SIFT algorithm;

* Scale-space extrema detection
* Key point localization
* Orientation assignment

It should be noted due to only having a single image provided we have to also make our own image for the assignment as well where the image is altered. The two images that we will be using for this is shown below;

## Figure 1: Images used with the Shift algorithm (Right Base, Left Orientation example)

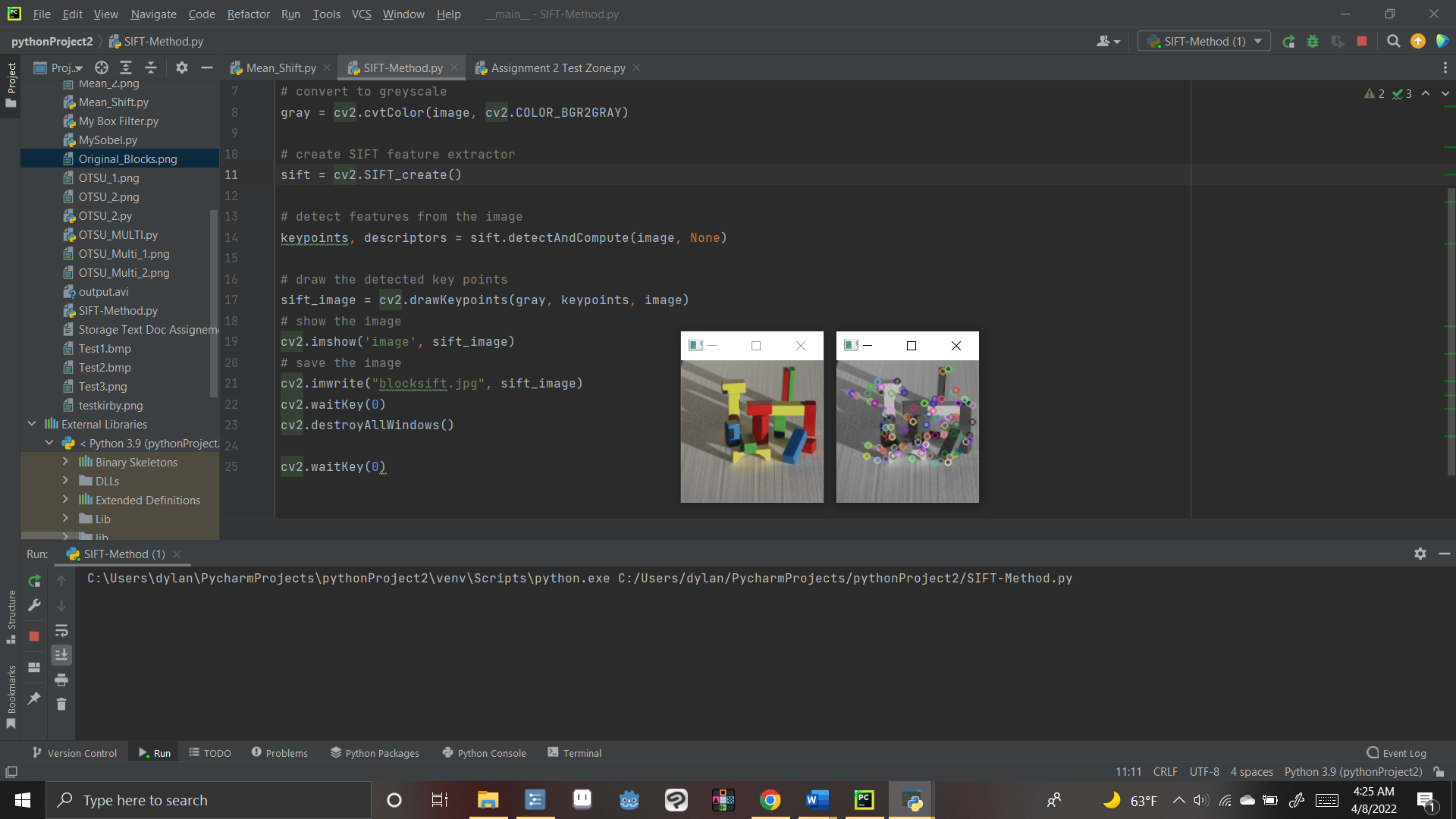
With the brief introduction completed let’s begin study of the SIFT Methodology.

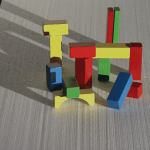
# Shift Methodology – Detection / key points localization

While there are a main set of three steps to the sift method, when it comes to the code-based items it really is more broken into two sets of steps, finding the key points, and then matching of the key points on different images. First off when it comes to detecting the key points, we can use open.cv built in functionalities with the sift method in order to find these rather easily. This is done by first running a detection for various forms of contrast on the image, after which then it discards the lower contrast one and tries to remove key points that are just the edges, this then results in the final set of key points shown. Below is the code for this segment of the code.

# Sift method part 1 Detection and keypoint localization   
import cv2 # import opencv  
  
# input the image  
image = cv2.imread('Original\_Blocks.png')  
  
cv2.imshow('Intial\_img', image) # baseline Image  
# convert the image to greyscale  
gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
  
# make the SIFT feature extractor  
sift = cv2.SIFT\_create()  
  
# detect features from the image  
keypoints, descriptors = sift.detectAndCompute(image, None)  
  
# draw on a new image the detected points  
sift\_image = cv2.drawKeypoints(gray, keypoints, image)  
# show the image produced  
cv2.imshow('image', sift\_image)  
# save the image produced  
cv2.imwrite("blocksift.jpg", sift\_image)  
cv2.waitKey(0) # wait so can see the image  
cv2.destroyAllWindows()

## Figure 2: Sift Algorithm Key Point Detection Code



## Figure 3: Sift Algorithm Key Point Detection Results

As you can see the Sift algorithm builds a bunch of key points on the various block structures. Overall, it does a very good job of detection of various points where the blocks has corners and shadows. In fact, it might even be working too well as it detects a lot of different areas for the key points, which could make the data a bit noisy and/or redundant. However, for our purposes of orientation assessment this might come to our benefit as it will allow us to have more potential examples to use.

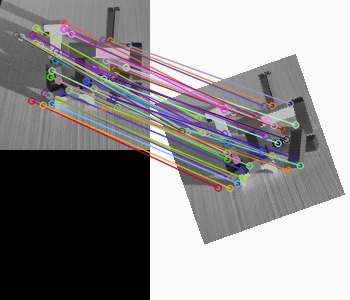
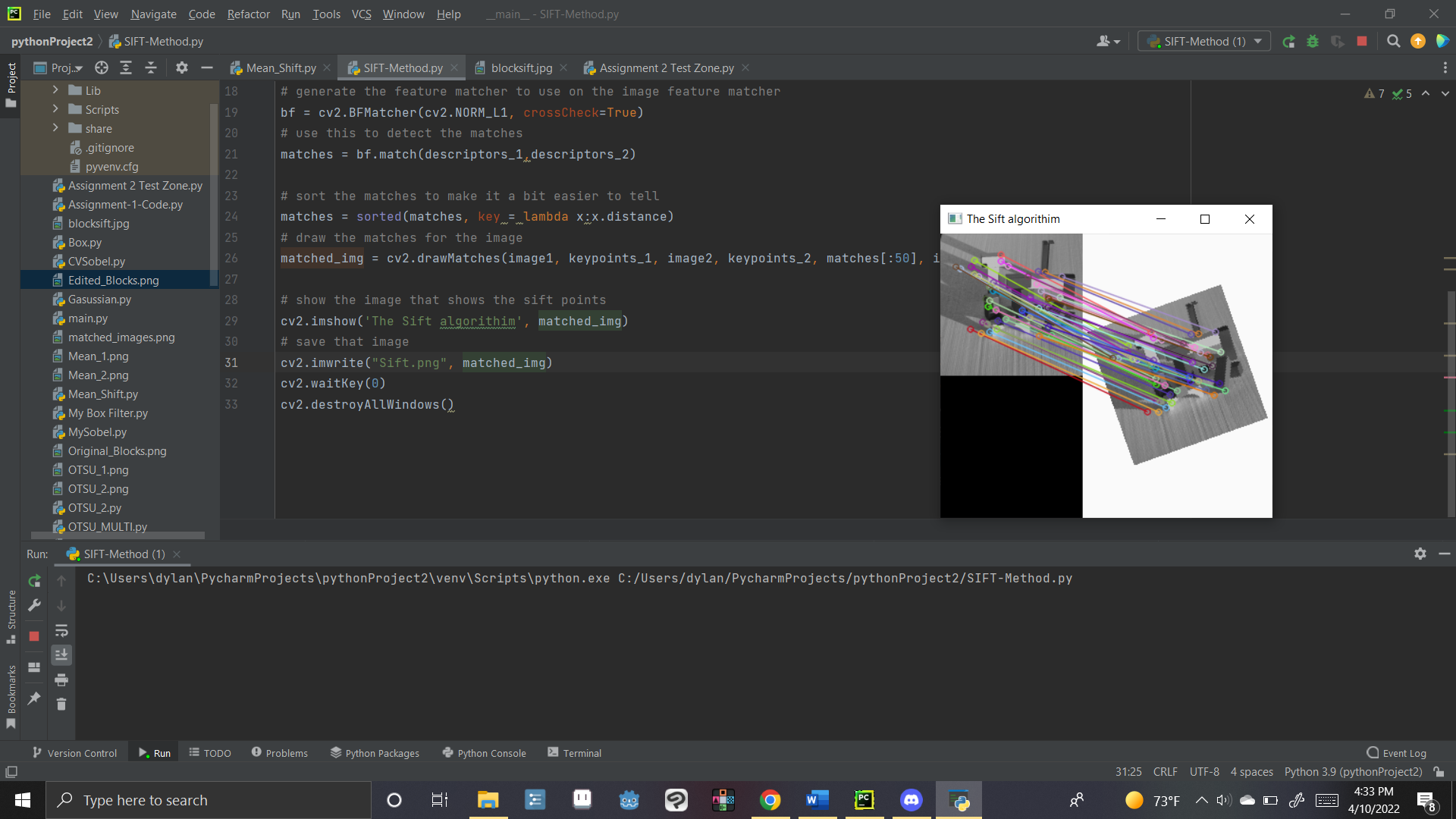
# Orientation assignment with feature matching

With the key points now found we can now try to look into orientation assignment and to connect points to one another. For this we simply need to use another one of OpenCV’s built in functions, there bf matcher. This scans the image and tries to connect similar points of interest to that of your input image, thus allowing you to do feature detection and matching. Below is the code for such:

# other half of the sift method code we have  
import cv2  
  
# input the images  
image1 = cv2.imread('Original\_Blocks.png')  
image2 = cv2.imread('Edited\_Blocks.png')  
# convert images to grayscale versions of the image  
image1 = cv2.cvtColor(image1, cv2.COLOR\_BGR2GRAY)  
image2 = cv2.cvtColor(image2, cv2.COLOR\_BGR2GRAY)  
  
# create the basline sift algoritim  
sift = cv2.SIFT\_create()  
  
# detect SIFT features in both images  
keypoints\_1, descriptors\_1 = sift.detectAndCompute(image1,None)  
keypoints\_2, descriptors\_2 = sift.detectAndCompute(image2,None)  
  
# generate the feature matcher to use on the image feature matcher  
bf = cv2.BFMatcher(cv2.NORM\_L1, crossCheck=True)  
# use this to detect the matches  
matches = bf.match(descriptors\_1,descriptors\_2)  
  
# sort the matches to make it a bit easier to tell  
matches = sorted(matches, key = lambda x:x.distance)  
# draw the matches for the image  
matched\_img = cv2.drawMatches(image1, keypoints\_1, image2, keypoints\_2, matches[:50], image2, flags=2)  
  
# show the image that shows the sift points  
cv2.imshow('The Sift algorithim', matched\_img)  
# save that image  
cv2.imwrite("Sift.png", matched\_img)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

## Figure 4: Sift Algorithm Orientation Assignment Code

And the results of such are the following:



## Figure 5: Sift Algorithm Orientation Assignment Results

As you can see by connecting the lines (all be it is rather cluttered due to the mass number of them and the small size of the image.) our code is able to find and connect most of the similar features that the previous one had. By using this we would be able to detect various different types of objects in all manner of different scenarios and as such would be very useful in the identification of objects.

# Challenges

For this assignment there was no real major issues encountered, besides outside forces that we had to deal with at the same time as this assignment.

# Conclusion

In conclusion, the SIFT algorithm is rather useful for object detection, and with modern technology is also rather easy to implement as well. While it might be a bit overly zealous in the number of features it is able to detect, with proper filtering it could be tuned to suit any multitude of possibilities.

# References -

The following Sources were used within this document to help with making the code and try to investigate our mean shift error

“Introduction to SIFT (Scale-Invariant Feature Transform).” *OpenCV*, https://docs.opencv.org/4.x/da/df5/tutorial\_py\_sift\_intro.html.

“SIFT Interest Point Detector Using Python - Opencv.” *GeeksforGeeks*, 11 Dec. 2020, https://www.geeksforgeeks.org/sift-interest-point-detector-using-python-opencv/.

Waheed, Ahmed. “SIFT Feature Extraction Using Opencv in Python.” *Python Code*, 18 Nov. 2020, https://www.thepythoncode.com/article/sift-feature-extraction-using-opencv-in-python#orientation\_assignment.