IMAGE FORGERY DETECTION USING STRUCTURAL SIMILARITY INDEX (SSIM), ORIENTED FAST AND ROTATION BRIEF (ORB) AND SCALE – INVARIANT FEATURE TRANSFORM (SIFT)

THIASAN S/O CHANDRAN

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

IMAGE FORGERY DETECTION USING STRUCTURAL SIMILARITY INDEX (SSIM), ORIENTED FAST AND ROTATION BRIEF (ORB) AND SCALE – INVARIANT FEATURE TRANSFORM (SIFT)

THIASAN S/O CHANDRAN

This report is submitted in partial fulfillment of the requirements for the Bachelor of Computer Science (Computer Security) with Honours.

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2021

DECLARATION

I hereby declare that this project report entitled

IMAGE FORGERY DETECTION USING STRUCTURAL SIMILARITY INDEX (SSIM), ORIENTED FAST AND ROTATION BRIEF (ORB) AND SCALE – INVARIANT FEATURE TRANSFORM (SIFT)

is written by me and is my own effort and that no part has been plagiarized

without citations.

STUDENT : [illegible] Date : 15/9/2021

(THIASAN S/O CHANDRAN)

I hereby declare that I have read this project report and found

this project report is sufficient in term of the scope and quality for the award of

Bachelor of [Computer Science (Computer Security)] with Honours.

SUPERVISOR : fadzilah Date : 15/9/2021

(DR. NUR FADZILAH BINTI OTHMAN)

DEDICATION

I would like to dedicate this dissertation to everyone who has worked towards the improvement of technologies and those who have been involved in digital crime investigation. Beside that, I would like to dedicate to all my motivators especially my family members, lecturers and each of everyone I have met. It is because everyone that came across our life are meant to teach us something. It is very important also to gratitude and dedicate this dissertation to the Almighty God whom always keeping me healthy and happy.

" Success is when your signature turns into your autograph"

" Winners are not those who never fail but those who never quit "

– APJ Abdul Kalam

All the best

ACKNOWLEDGEMENTS

I would like to convey my first deep gratitude to my supervisor, Dr. Fadzilah binti Othman, who has been my support pillar and guide me throughout my thesis. Her patience and knowledge helped me a lot to develop this thesis successfully and without her this thesis would not be completed successfully.

Next, I would like to express my gratitude to my family members especially my parents Chandran Periasamy and Susila Chandran whom always motivate me and give me space and time to focus on my thesis.

I would also like to thank everyone whom being my supports and motivators especially my friends and course mates. Their willingness to share their knowledge and guidance make me more motivated to complete this thesis successfully. At the same time, I would thank everyone that came across my life because everyone that came across our life meant to teach us a lesson for the future.

Last but not least, I should praise and thank the Almighty God and the mother nature to allow me complete this thesis successfully without any mental or physical issues. Thank you for always giving me the strength and a happiness life so that I can successfully complete this thesis.

ABSTRACT

Digital images are widely used in everywhere in the world. It is very helpful to mankind in solving many problems especially in the communication field. Nowadays, almost every human being is owning a smart device which is able to capture photos and videos easily. Almost millions of photos are uploaded every day in media socials such as on Instagram and Facebook. Previously, photos have been used only for educational purposes and as moments memories. There is no problem until the images are used genuinely without any crime involvement. On the other hand, the images are playing important roles in the crime scenes. Images are being used as photographic evidences in crime cases. So, it is important to maintain the integrity of the images. Unfortunately, the advancement of technologies has introduced various photo editing software such Adobe Photoshop which is able to modify the images and alter the contents easily. People are getting more skills on manipulating the original images where it directly effecting the integrity of an image. So, it is difficult for digital forensic department to identify between the original images and manipulated images. In this project, a system with 3 image forgery detection algorithms have been used to identify the similarities and differences between two images. The algorithms are Structural Similarity Index (SSIM), Oriented FAST and Rotated BRIEF (ORB) and Scale-Invariant Feature Transform (SIFT). This system is developed using Python as the programming language to integrate with those algorithms to perform image forgery detection. As a result, the system is able to identify the similarities and differences between two images in term of similarities score values and running time. Finally, this project concluded that the best image forgery detection for rotated images is ORB and for copy move forgery images is SSIM. This thesis helps to find out and recommend the best algorithms among SSIM, ORB and SIFT in terms of similarities score value, running time and detection areas.

ABSTRAK

Penggunaan gambar digital semakin meningkat di seluruh dunia. Ia amat membantu umat manusia dalam menyelesaikan pelbagai masalah terutamanya dalam bidang komunikasi digital. Pada masa kini, hampir semua manusia memiliki peranti pintar di mana peranti tersebut dapat menangkap gambar dan merakam video dengan mudah. Hampir jutaan gambar dimuat naikkan ke media sosial seperti Instagram dan Facebook. Sebelum ini, gambar atau imej digunakan untuk tujuan pendidikan dan sebagai simpanan memori sahaja. Penggunaan gambar digital tersebut tidak menimbulkan sebarang masalah sehingga gambar tersebut digunakan tanpa melibatkan jenayah. Pada masa yang sama, bukti forensik dalam bentuk gambar amat penting dalam sesuatu kes jenayah. Oleh kerana itu, menjaga integriti sesebuah gambar adalah sangat penting. Malangnya, situasi menjadi bertentangan apabila kemajuan teknologi memperkenalkan pelbagai jenis perisian penyuntingan gambar seperti *Adobe Photoshop* di mana ia dapat mengubah dan menyunting isi kandungan sesebuah gambar dengan mudah. Ramai orang semakin mahir dalam mengubah kandungan gambar dan perkara tersebut menyebabkan integriti gambar terjejas. Oleh kerana itu, pihak forensik digital menghadapi kesukaran untuk mengenal pasti antara imej sebenar dan imej yang telah diubah suai. Dalam projek ini, satu sistem telah dibangunkan menggunakan 3 algoritma untuk mengesan pemalsuan gambar. Sistem ini dapat mengesan persamaan dan perbezaan antara dua gambar dari segi skor persamaan dan masa yang diambil untuk mengenal pasti perbezaan antara dua gambar. 3 Algoritma tersebut adalah *Structural Similarity Index (SSSIM)*, *Oriented FAST and Rotated BRIEF (ORB)* dan *Scale-Invariant Feature Transform (SIFT)*. Sistem ini dibangunkan menggunakan bahasa pengekod Python. Sebagai keputusan, sistem ini dapat mengenal pasti persamaan dan perbezaan antara dua gambar dari segi nilai skor persamaan dan masa yang diambil. Akhirnya, projek ini dapat membuat kesimpulan bahawa algoritma yang paling sesuai untuk mengesan gambar yang diputarkan *(rotated image)* adalah ORB dan algoritma yang paling sesuai untuk gambar pemalsuan *(copy move forgery)* adalah SSIM Tesis ini dapat membuktikan algoritma yang terbaik antara SSIM, ORB dan SIFT dari segi skor persamaan, masa yang diambil dan kawasan pengesanan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

SSIM	Structural Similarity Index
ORB	Oriented FAST and Rotated BRIEF
FAST	Features from Accelerated and Segment Test
BRIEF	Binary robust independent elementary feature
SIFT	Scale Invariant Feature Transform

CHAPTER 1 : INTRODUCTION

Digital images are widely used in everywhere in the world. It is very helpful to mankind in solving many problems especially in the communication field. Nowadays, almost every human being is owning a smart device which is able to capture photos and videos easily. Almost millions of photos are uploaded every day in media social such as on Instagram and Facebook. Previously, photos are been used only for educational purposes and as moments memories. There is no problem until the images are used genuinely without any crime involvement. On the other hand, images are playing important roles in crime scenes. Images are being used as photographic evidences in crime cases. So, it is important to maintain the integrity of the images.

As the technologies are getting improved and innovated, the demands for the digital images are getting increase. Unfortunately, the advancement of technology has introduced various photo editing software such as Adobe Photoshop which allow users to modify and alter the content of the images easily. The trend image editing also getting popular among the people in digital media world. The editing tools are able to modify, delete, add and edit the original images to something else. This kind of actions led to the digital image forgery. In some cases, image forgery leads to phishing attacks and some other cybercrimes.

The manipulated images are considering a serious crime since some of the images are counted as evidences in the court. Image forgery led to spoiled the authenticity and integrity of the images. The digital images that changed and altered might purposely done for remove or hide important evidence or information with the help of editing software like Adobe Photoshop. So, there is a difficulty for the forensic department to identify the original and altered images. Since the image forgery takes place in digital world, digital forensics department also introduced to investigate and find out the real and modified images. There are few types of image forgeries that has been identified until now. The types of image forgery are :

i. Image Retouching
ii. Image Splicing
iii. Image Morphing
iv. Image Enhancing
v. Copy Move
vi. Scaling
vii. Cropping
viii. Geometric Transformation
ix. Selective Color Change
x. Merging Another Image or some parts of similar or different image.

Since the number of image forgery cases are increasing, the world has been introduced with Image Forgery Detection Techniques. This detection techniques we need in many fields to protect and maintain the integrity and originality of the images. It can be used to preserve the copyright of the images. The main problem in image forgery is human cannot differentiate the original and tampered images with naked eyes. It is because that’s how perfect software and tools help to modify the images. To win against this image forgery, we need also some software and tools especially some techniques to identify the tampered images. There are two main categories of the image forgery detection techniques that are existing now which are :

i. Active Method
ii. Passive Method

Active Method forgery detection is referring to digital signatures and watermarking. These methods have been used widely in photography world to authenticate the originality of an image. There are several techniques have been used in the digital world to identify the tampered images. The techniques are :

i. Pixel-Based Image Forgery Detection
ii. Format-Based Image Forgery Detection
iii. Camera-Based Image Forgery Detection
iv. Physical environment-Based Image Forgery Detection
v. Geometry-Based Image Forgery Detection

1.1 Problem Statement

The tampered images are hard to be identified by normal naked human eyes. Some detection techniques have been introduced and implemented to identify the differences between original and tampered images. In some cases, the image forensic department need both original and tampered images to identify the difference between them. Moreover, in some image forgery cases the original images are hard to be found. Unfortunately, some detection techniques are not giving the accurate results. Sometimes, some information has been failed to be identified using some detection techniques. Low accuracy result will make harder to distinguish between original and tampered images.

Table 1.1 : Problem Statement

Problem Statement
i. Some image forgery detection techniques and algorithms are not efficient in producing the differences between original and tampered images.

1.2 Research Question

This study attempts to answer the following research questions:

Table 1.2 : Research Question

Research Question
i. How to identify between original image and manipulated image with a high accuracy result?
ii. Which detection algorithm is the best and efficient in terms of similarities score values and process runtime?

1.3 Research Objective

The objectives of this project are :

i. To explore and apply the Structural Similarity Index (SSIM), Scale Invariant Feature Transform (SIFT), Oriented FAST and Rotated BRIEF (ORB) algorithm to identify the similarity of two images.

ii. To identify and detect the differences between original and manipulated images.

iii. To recommend and propose the best algorithms between 3 of the algorithms based on the similarities score value and process runtime.

1.4 Project Scope

The scopes of this project are listed below :

i. This project will implement the OpenCV, Scikit – Image and Imutils along with Python programming language to develop the system.

ii. This project will develop Image Forgery Detection systems using Structural Similarity Index (SSIM) Approach, Scale Invariant Feature Transform (SIFT), Oriented FAST and Rotated BRIEF (ORB) algorithm.

iii. This detection program will turn the color images into a grayscale in order to identify and compute the differences between original and manipulated images.

iv. Drawing module will be added in this program. Once the differences were identified, the system will draw out the differences places with a rectangle shape or circle shape or lines according to their own searching methods.

v. The algorithms will be tested for two image categories which are rotated images and copy move forgery images.

1.5 Project Contribution

The project contribution is mainly to the forensic department which are able to find the differences between original and manipulated images. This program is able to identify and mark the differences between two images. The project contributions are as shown below :

i. A program that able to help forensic department to identify the differences between original and manipulated images.

ii. A program that able to find matching patches in stereo images and to track the pattern through an image sequence and determine exactly where the coordinate of the image differences (x , y).

iii. Propose and recommend best and efficient algorithms among SSIM, ORB and SIFT.

1.6 Report Organization

Chapter 1 : Introduction

This chapter covers and discuss about the project introduction and project background. It also includes project's problem statement, project question, project objective, project scope, project contribution, report organization and the conclusion of the project introduction.

Chapter 2 : Literature Review

This chapter covers and explains about the previous research or studies that have been done by any other researches across the world that similar or related to this project. In this chapter, it includes introduction about the literature review, related or previous works that have been done before. It also includes the critical review of the current problems and the justification for the problem. Furthermore, proposed solution also has been included in this chapter and the conclusion of the chapter 2.

Chapter 3 : Project Methodology

This chapter discuss about the flow of the project will be developed. Prototypes, diagrams and milestones of every stages in this project will be discussed and shown in this chapter.

Chapter 4 : Analysis and Design

This chapter includes the design of the project such as logical and physical design, the system architecture and all the designs that related to this project will presented and explained in this chapter.

Chapter 5 : Implementation

This chapter discuss about the steps and activities that involved during the implementation phase. Expected output is also included in this chapter. The analysis of the source codes are also included in this chapter.

Chapter 6 : Testing and Analysis

This chapter discuss about method and steps to test the program. The expected outputs are included in this chapter as well.

Chapter 7 : Project Conclusion

This chapter covers the summarization of the entire project.

1.7 Conclusion

As for the conclusion for chapter 1, this project is mainly focuses on the project background which describes about the problem statement, research questions and research objectives, project scopes and project contributions. This project is developed to help the image forensic department to identify the differences between original and manipulated images. The identified differences will help them generate their report on image forgery cases.

CHAPTER 2 : LITERATURE REVIEW

2.1 Introduction

This chapter mainly focuses on information, solutions and studies that already existing in the world that related to the scope of this project. Furthermore, the studies and proposed solutions by other researchers have been reviewed for project study purposes. It is to have a better understanding on the scope of this project to achieve the project objectives. All the previous research has been compared and reviewed as study materials.

The main element of this project is on developing an image forgery system using SSIM Approach, ORB and SIFT using a Python programming system. This includes a software development for the detection technique. Previously, many researchers have done their research on image forgery detection techniques. Some of the techniques are being used in worldwide now to detect the image differences. Most of the image detection technique are unable to identify accurately the manipulated location in the forged image. The main concept of this project is to identify the differences between original and manipulated images and mark the differences to make the identification process easy and fast.

2.2 Image Forgery Detection Algorithms Concepts

2.2.1 Structural Similarity Index (SSIM)

Structural Similarity Index (SSIM) is method that developed to measure the similarities between two images. The luminance if the surface of an object being observed is the product of the illumination and the reflectance, but the structures of the objects in the scene are independent of the illumination. (Zhou Wang, 2004). The structural influence of the illumination will be separated to view the structural information in an image. The job of similarity measurement will be divided into three parts which are luminance, contrast and structure. First, the luminance of each signal is compared and assuming discrete signals, this is estimated as the mean intensity (Zhou Wang, 2004) :

$$SSIM(x, y) = [l(x, y)]^{\propto} \cdot [c(x, y)]^{\beta} \cdot\ [s(x, y)]^{\gamma}$$

Where:

l (x, y) – Luminance comparison function

c (x, y) – Contrast comparison function

s (x, y) – Structure comparison function

Below formula is a simplified expression after α = β = γ = 1 and C3 = C2/2 :

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

The author of the formula explained that, for image quality assessment, it is useful to apply the SSIM index locally rather than globally. First, image statistical features are usually highly spatially nonstationary. Second, image distortions, which may or may not depend on the local image statistics, may also be space- variant. Third, at typical reviewing distances, only a local area in the image can be perceived with high resolution by the human observer at one time instance) because of the foveation feature of the HVS). And finally, localized quality measurement can provide a spatially varying quality map of the image, which delivers more information about the quality degradation of the image and may be useful in some applications. (Zhou Wang, 2004). So, a new improved formula has been introduced which called Mean Structural Similarity Index.

$$MSSIM(X,Y) = \frac{1}{M}\sum_{j=1}^{M} SSIM(x_j\,, y_j)$$

The system will compare the two images based on 3 features which are Luminance, Contrast and Structure. The figure below shows how a Structural Similarity Index (SSIM) approach will work when comparing two images.

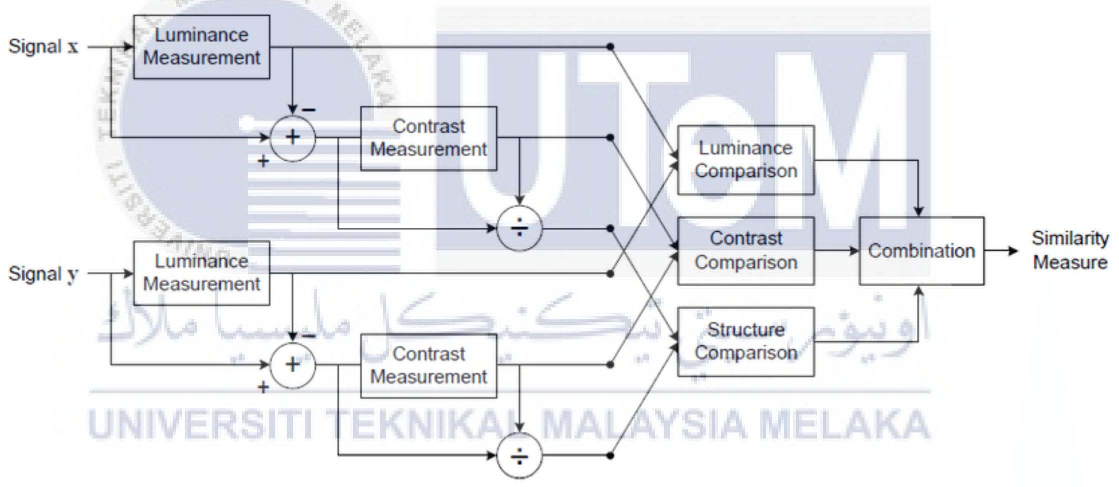


Figure 2.1 : Process for SSIM

(Zhou Wang, 2004)

Signal X and Signal Y are referring to the original (references) and manipulated (sample) images. This system will calculate the SSIM between the two images which will a value between -1 + 1. Value +1 is indicating the original and manipulated images are almost similar or both images are same and there is no differences between them while value -1 is indicating the original and manipulated images are very different. (Datta, 2020).

2.2.2 Scale-Invariant Feature Transform (SIFT)

Scale Invariant Feature Transform (SIFT) is an algorithm that widely used in image forgery detection. It is a key point-based method to detect the similarities between two images. Although it is a very efficient technique to detect the similarity but the process of SIFT is slow. It needs some time in terms of processing the images to extract and detect the forgery. This algorithm is coming from the scale-space theory. SIFT will examine the images extreme point, location, scales and rotations. One of the advantages of the SIFT algorithm is it is able to detect the changes of an image even though the image has been resized or rotated.

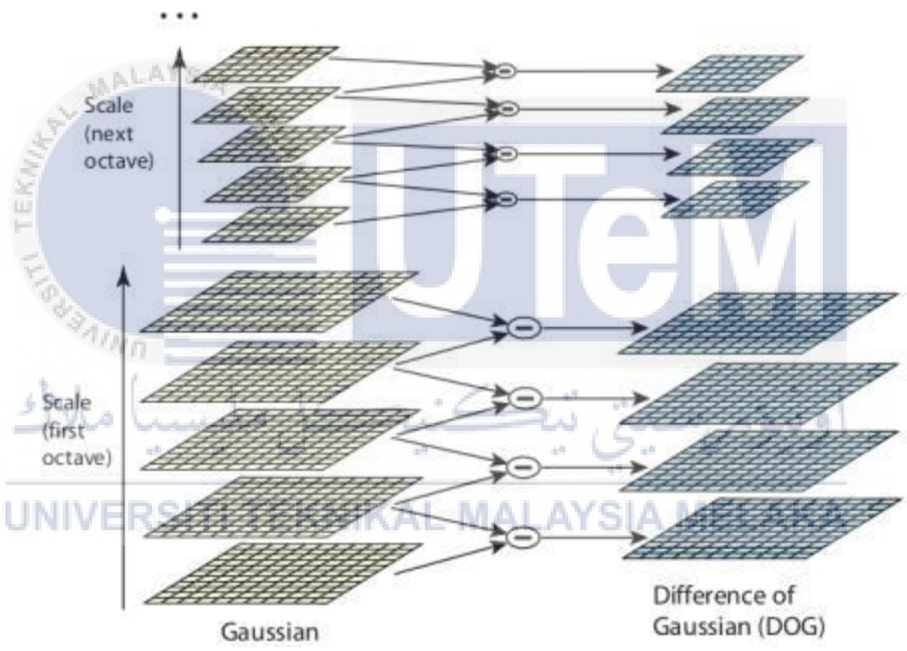


Figure 2.2 : Process of SIFT

(Lv, 2015)

2.2.3 Oriented FAST and Rotated BRIEF (ORB)

Oriented FAST and Rotated BRIEF (ORB) is one of the algorithms that commonly used nowadays to detect the image forgery. This algorithm was lately developed until it have the same superiority of the SIFT algorithm and it has been improved in the terms of the detection duration and the percentage of high accuracy detection on moving targets in real time. ORB algorithm basically is divided into three main steps which are :

i. Feature Point Extraction
ii. Feature Point Descriptors Generation
iii. Feature Point Matching

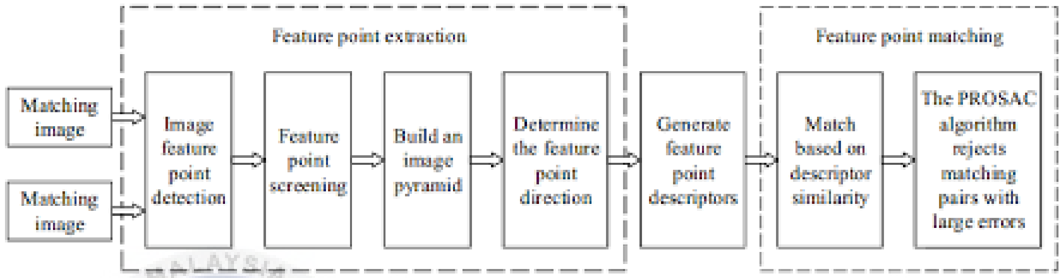

Feature point extraction
Feature point matching
Matching image
Matching image
Image feature point detection
Feature point screening
Build an image pyramid
Determine the feature point direction
Generate feature point descriptors
Match based on descriptor similarity
The PROSAC algorithm rejects matching pairs with large errors

Figure 2.3 : Process of ORB

(Chuan Luo, 2019)

2.3 Critical Review of existing detection techniques

The Table 2.1 shows some previous works that have been done by various people on different image forgery detection techniques. The works are been categorized according to detections' techniques name. These research papers have been reviewed to collect the studies about previous researches and works that implement algorithm in image forgery detection especially on SSIM, ORB and SIFT. So, it can help to develop a better understanding on project scopes and system developments.

Table 2.1 : Literature Review of Previous Works

Technique Name	**Technique Author**	**Paper Title**	**Features**	**Algorithm**			
				SSIM	SIFT	ORB	OTHERS
Copy – Move Forgery Detection	1. Xunyu and Siwei (P.Xunyu, 2011)	Region Duplication detection using Image Feature Matching	Scale -Invariant Feature Transform Keypoints (K-means clustering)		▨		
	2. Ye Zhu, Xuanjing Shen, Haipeng Chen	Copy-Move Forgery Detection Based on Scaled ORB	Scaled ORB			▨	

	3. Xiuxia Tian, Guishuai Zhou, Man Xu	Image Copy-Move Forgery Detection Algorithm Based on ORB and Novel Similarity Metric	Based on ORB and Novel Similarity Metric			▨	
	4. Sutthiwan (P. Sutthiwan, 2010)	Rake Transform and Edge Statistic for Image Forgery Detection	Multi – size Block DCT markov process				▨
	5. Luo (W.Luo, 2007)	A Survey of Passive Technology for Digital Image Forensics	Based on Intensity				▨
	6. Meenal Shandilya, Ruchira Naskar	Detection of Geometric Transformations in Copy Move Forgery of Digital Images	Scale Invariant Feature Transform (SIFT)		▨		
	7. Neetu Yadav, Rupal Kapdi	Copy Move Forgery Detection using SIFT and GMM	SIFT and GMM		▨		

	8. Atefeh Shahroudnejad, Mohammad Rahmati	Copy – Move Forgery Detection in Digital Images Using Affine – SIFT	Affine - SIFT		▨		
Detection using Illuminant Color	1. Reshma P.D and Arunvinodh C (Reshma P.D, 2015)	Image Forgery Detection Using SVM Classifier	Detection is based on machine learning and based on illuminant color				▨
Detection using Contrast Calculation	1. Ms. Jayshri Charpe (Ms Jayshri Charpe, 2015)	Revealing Image Forgery Through Image Manipulation Detection	Detection based on the contrast calculation of the contrast enhanced images.				▨
Detection method using Local Binary Pattern	1. Fahime Hakimi, Mahdi Hariri, farhad GharehBaghi (Fahime Hakimi, 2015)	Image Splicing Forgery Detection Using Local Binary Pattern and Discrete Wavelet Transform	Detection method is based on the Local Binary Pattern and Discrete Wavelet Transform				▨
Detection using Discrete Cosine Transform (DCT)	1. Matthias Carnein, PascaSchottle, Rainer Bohme	Forensics of High Quality JPEG Images with Color Subsampling	Detection method using DCT where a concept of block convergence and prolonged it to the color				▨

	(Matthias Carnein, 2015)		images. The detection is achieved by reviewing the behavior of convergence for the blocks.				▨
Detection using 2D – Discrete Wavelet Transform (DWT) and Singular Value Decomposition (SVD)	1. Varsha Karbhari Sanap, Vanita Manikrao Mane (Varsha Karbhari Sanap, 2015)	Region Duplication Forgery Detection in Digital Images Using 2D – DWT and SVD	Detection method using 2D – Discrete Wavelet Transform (DWT) and Singular Value Decomposition (SVD) which gives better accuracy compared to other techniques.				▨
Detection using Affine Scale Invariant Feature Transform (SIFT)	1. Atefeh Shahroudnejad, Mohammad Rahmati (Atefeh Shahroudnejad, 2016)	Copy – Move Forgery Detection in Digital Images Using Affine SIFT	Detection using process like blurring and addition Gaussian noise.		▨		
Detection of Robust Image Forgery using JPEG Artifacts technique	1. Fridrich and Lukas (J.Lukas, 2003)	Estimation of Primary Quantization Matrix in Double	Detection using Quantization Method Estimation (Neural Network Classifier)				▨

		Compressed JPEG Images					■
Image Splicing Detection Methods	1. Lint, Wang, Tang and Wang (Z.Lint, 2005)	Detecting Doctored Images using Camera Response Normality and Pattern Recognition	Detection using response of inverse camera functions by analyze the edges.				■
Image Differences using OpenCV and Python	1. Zhou Wang, Alan, Hamid and Eero (Zhou Wang, 2004)	Image Quality Assessment : From Error Visibility Structural Similarity	Detection using Structural Similarity Index (SSIM), Python, OpenCV, scikit – image and imutils	■			
	2. Adrian Rosebrock (Rosebrock, 2017)	Image Difference with OpenCV and Python		■			
			Based on ORB			■	

Improved Image Matching Method Based on ORB	1. Letian Li, Lin Wu, Yongcun Gao	Improved Image Matching Based on ORB				■	
Image Forgery Detection using SSIM	1. Alisha James, E.Bijolin Edwin, Anjana M C, Angel Abraham, Harsha Johnson	Image Forgery Detection on Cloud	Based on SSIM	■			

According to the Table 2.1, most of the algorithm that have been used in the detection techniques are SIFT apart from the algorithm that categorized as others. Nowadays, as the technologies getting improved and the image processing technology also getting advanced, most of the detection techniques started to implement ORB and SSIM because of their capabilities in producing effective results. These collected studies helps in understanding better about image processing and the 3 algorithms.

2.4 Proposed Solution

Image forgery detection techniques using Structural Similarity Index (SSIM) approach, Scale Invariant Feature Transform (SIFT), Oriented FAST and Rotated BRIEF (ORB) algorithm will be developed and implemented in this project to propose the best and efficient detection algorithms among three algorithms. A Python coding program will be developed along with OpenCV, Scikit – Image and Imutils. In this project, the program is able to identify the differences between original and manipulated images and make a decision whether the two images are identical or having any differences that are not visible to human's naked eyes. Furthermore, this program is able to identify and determine the exact location in terms of (x, y) of the image differences. Firstly, the program will require to include the two images that needed to be compared into one specific folder. This system will compare the two images based on the features of each algorithm. Then, it will run the process and change the colored images into grayscale. Once the program identified the differences between the two images, it will draw rectangle shaped boxes or circles or lines around location that have the differences or similarities between original and manipulated images. The SSIM approach will be great in determining the image forgery compared to other existing detection techniques especially in identifying the differences between two images. But the main purpose of this project is to find out the best image forgery detection algorithm among SSIM, SIFT and ORB. The percentage of similarities score, running time and areas detection in final result will determine which will be the best algorithm. So, there will be two categories' images will be used for testing. The categories are rotated images and copy move forgery images.

2.5 Conclusion

In conclusion, this chapter has shown some previous research works and developed detection techniques and algorithms that existing today for identifying the image forgery. The literature review is an important chapter since it will help this project in terms features that need to be improvised. It will help to get a better understanding about the detection technique that will be developed in this project.

CHAPTER 3 : PROJECT METHODOLOGY

3.1 Introduction

In this chapter, project methodology will be discussed. Project methodology is techniques and analysis that has been used in this project to develop the project. It includes from data collection, analyzing the previous studies, collect the datasets for the project purpose and so on. It will function as the main flow and plan that will bring this project in a correct path in order to achieve the objectives of the project. The Waterfall model flowchart will be applied and used in this project to demonstrate the flow of the project.

3.2 Methodology

The waterfall methodology flow chart has been implemented in this project. The process of framework has been designed with a flowchart as shown in Figure 3.1:

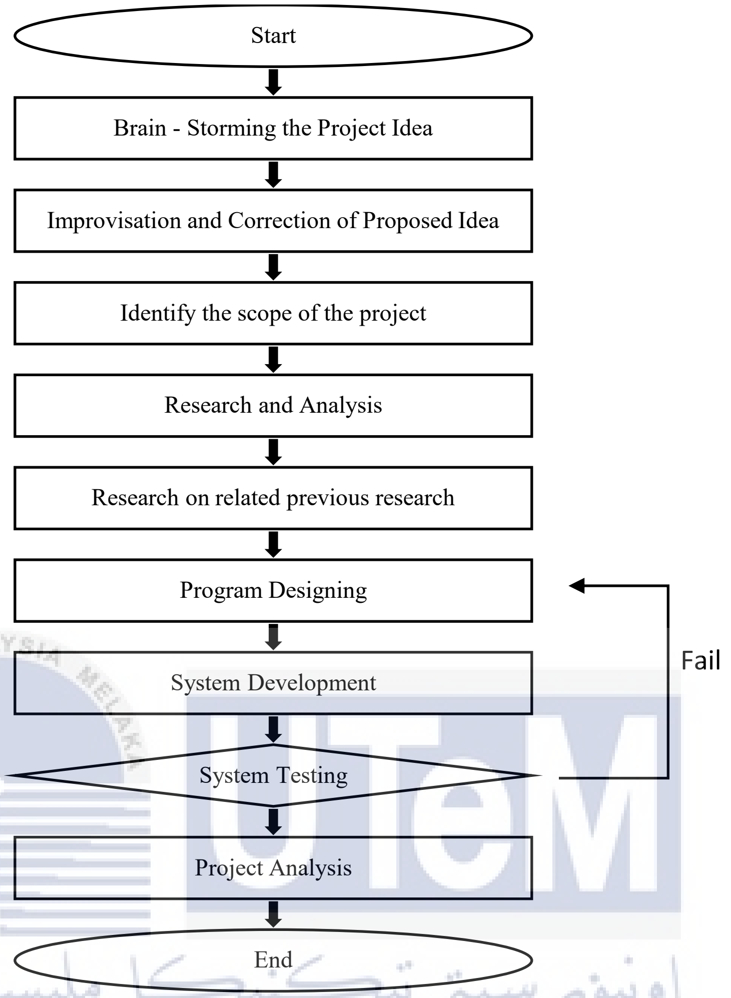

Start
Brain - Storming the Project Idea
Improvisation and Correction of Proposed Idea
Identify the scope of the project
Research and Analysis
Research on related previous research
Program Designing
System Development
System Testing
Fail
Project Analysis
End

Figure 3.1 : Flow Chart of Research Methodology

Table 3.1 : Activities of each stages

Stages	**Activities**
Brain Storming of the Project Idea	Gather and list down all final year project title ideas that suitable to my field of studies and propose the ideas to supervisor

Improvisation and Correction of Proposed Idea	Enhance the prepared proposal and improve the ideas according to the title. Identification of research objectives and problem identifications.
Project Scope Identification	Identify the scopes of this project and implementations of the project.
Research and Analysis	Collect and do research on the previous works that related to this project title. Gather some information from the previous studies to have better understanding on the scopes and implementations on the systems.
Research on previous related works	
Program Designing	Initial and rough design of the expected system's outcome and decision on what coding languages and modules and packages that must be included for system development.
System Development	Implement the designed ideas into practically running systems.
System Testing	Test the systems that have been designed and developed from previous stages.

Project Analysis	Do analysis from the system testing to find out weakness and errors of the systems and do further improvisation for the systems.

3.3 Milestone

A project milestone is important to complete a project successfully. It is well organized and planned to smoothen the project process. It helps to bring the project progress without any problems and delay. Milestone will make sure the project finish by the time given. Furthermore, a Gantt chart is included to illustrates the progress of the project. The milestone is presented in Table x and a Gantt chart is displayed in Table x.

Table 3.2 : Project Milestone

Phase	**Week**	**Progress**
	1	**Meeting 1** - Proposal PSM : Discussion - Proposal Assessment and Verification
	2	- Proposal Correction and Improvement - List of Supervisor / Title
	3	**Meeting 2** - Proposal Presentation and Submission via PSM Online System

Planning Phase		- Chapter 1 (System Development Begins)
	4	- Chapter 1 - Chapter 2
	5	- Chapter 2
Analysis	**6**	**Meeting 3** - Chapter 2 - Chapter 3
	7	- Chapter 3 - Chapter 4
	8	- Chapter 4 - Project Demo
	9	**Meeting 4** - Chapter 4 - Project Demo
	10	- Project Demo

Design	**11**	- Project Demo - PSM 1 Report
	12	**Meeting 5** - Project Demo - PSM 1 Report
	14	**Final Presentation**
	15	**Revision Week**
	16	**Final Examination Week**

Table 3.3 : Project Gantt Chart

Project Activity	**Week**														
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
First Meeting and Discussion with Supervisor	■														
Proposal Improvisation		■													
Proposal Submission			■												

Development of System				■											
System Designing					■										
Implementation of Project						■	■	■	■	■					
System Testing and Analysis											■	■			
System Maintenance												■	■		
Final Report Preparation													■	■	
Final Report Submission and Presentation															■

3.4 Conclusion

In conclusion, this chapter wraps about the methodology that has been applied in this project. This research methodology helps the project developing process smoothly and the project can be done within the time given. A project milestone and Gantt chart are presented in this chapter to demonstrate the development progress of this project along with the time given.

CHAPTER 4 : ANALYSIS AND DESIGN

4.1 Introduction

This chapter is about the analysis and design of the approach and system development process are been identified as the preparation for the next phase. This chapter further explains about the coding language and packages that needed to be developed and install for the project. At the same time, this chapter explains about the process and development of coding for the project.

4.2 System Architecture

This coding system is developed and designed in a way that will have input of the images, process and the result. The input of the system will require users to give two images (original image and manipulated image). Once the system gets the images, the process will start by examining the images for checking the similarity and differences between the two images. This system uses Python language coding and the packages of SSIM, ORB and SIFT inside the Python. So, it makes the calling function easier since it is already in the Python modules. The system will use 3 algorithms to compute the similarities score between two images and give the similarity score result and time taken to process at the end of the system. As for the output, the system will show the similarity score result with a percentage number. If the result number is more to 1 means the two images are almost similar and if the value getting closer to 0 means the two images are not similar.

4.3 Modules

4.3.1 OpenCV (Open-Source Computer Vision Library)

OpenCV (Open Source Computer Vision Library) is an open source software which consists machine learning software library and computer vision. It is used to solve computer vision problems. It supports most popular types of programming languages such as Python, C++, Java and so on. In this project, the detection technique programming is implemented with OpenCV and Python.

4.3.2 Scikit-Image

Scikit – image is algorithm that used in image processing purpose. It is a package that included in Python which dedicated to image processing. It is an open-source tools. Since this project is mainly to develop about detecting image forgery which works with digital image, this Python package is important and compulsory to be included. (Ansari, 2020)

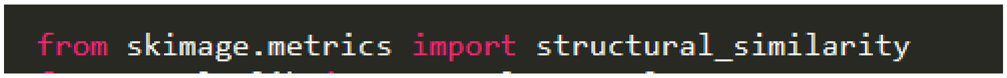
4.3.3 Imutils

Imutils is a Python package. It is a series of functions to perform the basic image processing such as rotation, translation, resizing, skeletonization, displaying Matplotlib images, sorting contours, and detecting edges. (python, 2021).

4.4 Packages or Algorithms (Coding Commands)

Packages or module inside Python is really helpful for this project implementation. It is because we no need to rewrite the coding commands for the algorithms during the system development. We just need to call and include the packages or module in system development. The modules are as mentioned below:

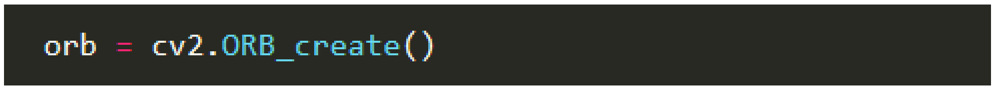
Figure 4.1 showing that package scikit-image metrics is imported from structural similarity function to make sure the SSIM algorithm run smoothly. Function of this package module is it is used for image processing in Python.



```
from skimage.metrics import structural_similarity
```

Figure 4.1 : SSIM Package Inclusion

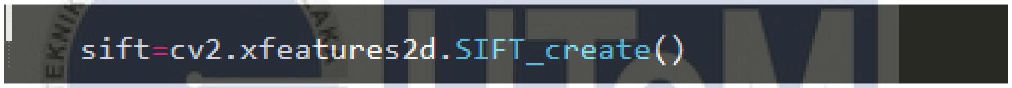
Figure 4.2 showing that package module ORB algorithm is imported to the system.



```
orb = cv2.ORB_create()
```

Figure 4.2 : ORB Package Inclusion

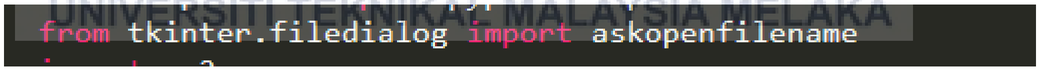
Figure 4.3 shows that SIFT package module is imported to the system.



```
sift=cv2.xfeatures2d.SIFT_create()
```

Figure 4.3 : SIFT Package Inclusion

Figure 4.4 shows that module package tkinter file dialog is imported from askopenfilename function which this module will help the system to prompt the windows console and requires users to choose the images.



```
from tkinter.filedialog import askopenfilename
```

Figure 4.4 : Package for User Input

The main process of the system is presented as illustration image below. Firstly, the system will prompt users to choose two images to be processed. Once the user selects the images, the system will start to examine and analysis the images. The system will analyze for the similarities and differences that occur between two images that selected by the user.

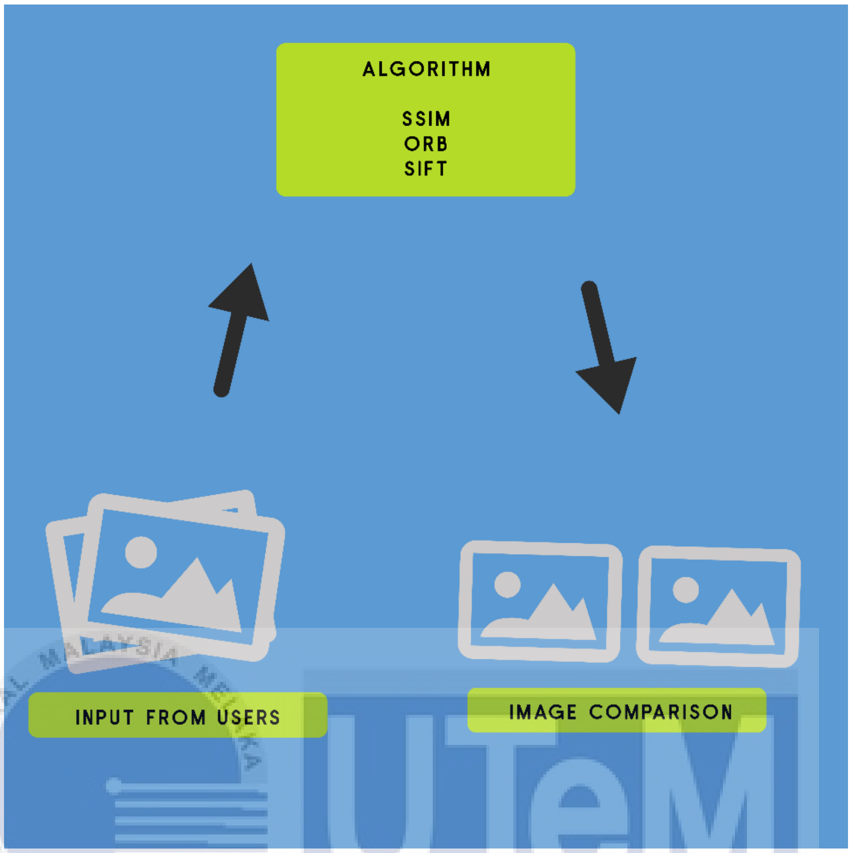


Figure 4.5 : System Ideology Flow

This coding is combined 3 algorithms that mentioned above and define them into own separate functions. So, every one of the algorithms will be their own defined functions. At last, the system will compare the same images using 3 algorithms and compare the scores and running time of each other.

4.5 Conclusion

As for conclusion, this chapter covers the outlines and how the system will work in order to identify differences between two images. The coding analysis of the systems are also been presented in this chapter. The implementation and testing of comparing two images using the algorithms will be performed on next chapter.

CHAPTER 5 : IMPLEMENTATION

5.1 Introduction

This chapter mainly focus on the implementation of the project. Full source codes of the system will be included and explanation for the source codes are already been explained in previous chapter. The implementation in detecting forgery using two different categories of images will be performed in this chapter using the designed or developed code on previous chapter.

5.2 Environment Setup

There are 3 stages that involved in this image forgery detection system.

1. Data Input
   During this stage, user will be prompted with windows console to select two images for the detection.

2. Algorithm Processing
   During this stage, two images will be process through each algorithm with their own respective methods.

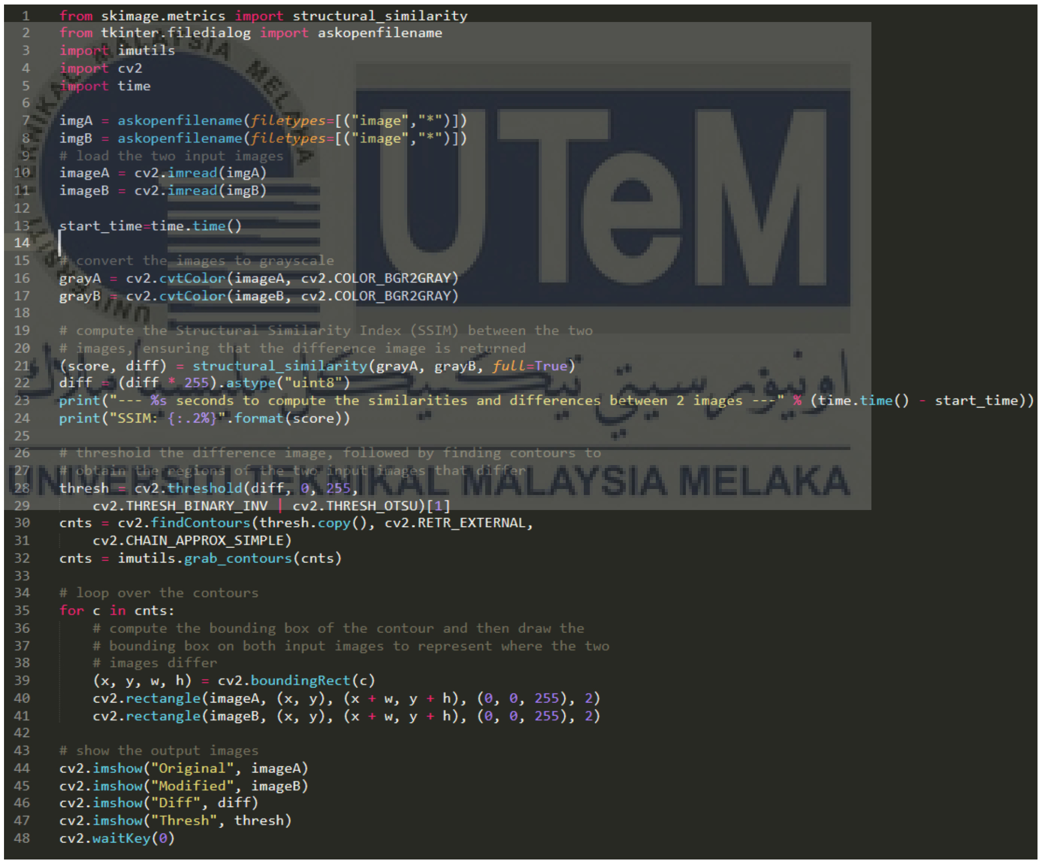
3. Result
   During this stage, the inputted images will be shown on screen with their differences and similarities between the two images. At the same

time, a score value of similarities and running time will be calculated and shown on the windows console.

5.3 Coding Implementation

All system development in this project is being developed by Python, OpenCV and some other libraries as for each image forgery detection algorithms which will be explained below. Each coding command will be shown in the figures and followed by the explanation of the coding.

i. Structural Similarity Index (SSIM)

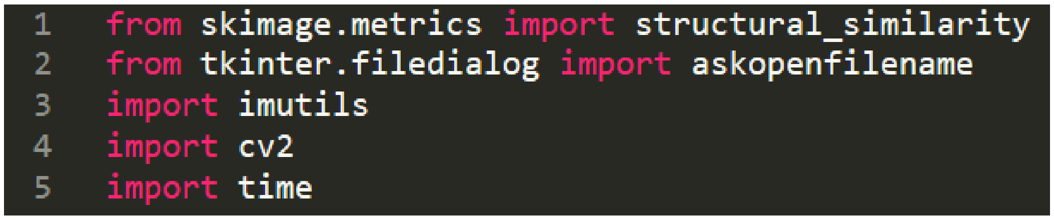


```python
from skimage.metrics import structural_similarity
from tkinter.filedialog import askopenfilename
import imutils
import cv2
import time

imgA = askopenfilename(filetypes=[("image","*")])
imgB = askopenfilename(filetypes=[("image","*")])
# load the two input images
imageA = cv2.imread(imgA)
imageB = cv2.imread(imgB)

start_time=time.time()

# convert the images to grayscale
grayA = cv2.cvtColor(imageA, cv2.COLOR_BGR2GRAY)
grayB = cv2.cvtColor(imageB, cv2.COLOR_BGR2GRAY)

# compute the Structural Similarity Index (SSIM) between the two
# images, ensuring that the difference image is returned
(score, diff) = structural_similarity(grayA, grayB, full=True)
diff = (diff * 255).astype("uint8")
print("--- %s seconds to compute the similarities and differences between 2 images ---" % (time.time() - start_time))
print("SSIM: {:.2%}".format(score))

# threshold the difference image, followed by finding contours to
# obtain the regions of the two input images that differ
thresh = cv2.threshold(diff, 0, 255,
    cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)

# loop over the contours
for c in cnts:
    # compute the bounding box of the contour and then draw the
    # bounding box on both input images to represent where the two
    # images differ
    (x, y, w, h) = cv2.boundingRect(c)
    cv2.rectangle(imageA, (x, y), (x + w, y + h), (0, 0, 255), 2)
    cv2.rectangle(imageB, (x, y), (x + w, y + h), (0, 0, 255), 2)

# show the output images
cv2.imshow("Original", imageA)
cv2.imshow("Modified", imageB)
cv2.imshow("Diff", diff)
cv2.imshow("Thresh", thresh)
cv2.waitKey(0)
```

Figure 5.1 : SSIM Full Coding

The system coding is divided into several parts. Starting from the package and module inclusion until to print out the differences and similarity scores.

First part of the coding. (Header)



```
1   from skimage.metrics import structural_similarity
2   from tkinter.filedialog import askopenfilename
3   import imutils
4   import cv2
5   import time
```

Figure 5.2 : Libraries Needed for SSIM (Line 1 – 5)

Figure 5.2 showing that the commands that system needs to include such as algorithm functions, libraries and some modules in order to make the system work perfectly.

Explanation of Figure 5.2

Line 1 ***from skimage.measure import compare_ssim***

Scikit – image (skimage) is a library for image processing that been used in Python. Since, we are dealing with images, so we need this library to handle the image processing. So, in order to use the Scikit – Image library, we need to install the library if it is not been installed and if it is already installed then we need to upgrade the scikit – image library to the latest updates.

compare_ssim is module for compute the mean structural similarity index between two images. It is more to a function to perform the image processing.

Line 2 ***from tkinter.filedialog import askopenfilename***

This is a module where it deals with file system. Since our system will prompt and ask users to choose two images for comparison from the windows file system. So, this module will provide the dialog that allows for the file (image) selections. This module is also allow the users to select more than one images and return the selected files (images) to the system for processing. Once the users selected the images, *open()* method will allow and open them.

Line 3 ***import imutils***

This is a complication of functions that help in image processing in terms of rotation, scaling, resizing, translation, detecting edges and much more with the help of OpenCV and Python. This module or package is being used in computer vision and image processing.

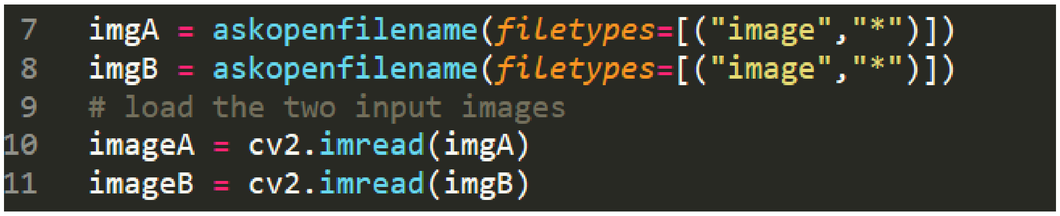
Line 4 ***import cv2***

This is a library of Python where it is use for works such as computer vision and image processing. The full name of *cv2* is Open-CV. This is mostly used in Python for computer vision works purposes. For example, to read or display the images in the system. The system will use *cv2.imread()* to read the image that been specified.

Line 5 ***import time***

This is a module is consisting of functions that are related to time. This module is been used to calculate the running time of the system. So, it is able to calculate how long will the program will be taken to run the whole system.

Middle part of the coding (Body)



```
7   imgA = askopenfilename(filetypes=[("image","*")])
8   imgB = askopenfilename(filetypes=[("image","*")])
9   # load the two input images
10  imageA = cv2.imread(imgA)
11  imageB = cv2.imread(imgB)
```

Figure 5.3 : Prompt User to choose images (Line 7 – 11)

Explanation of Figure 5.3

Line 7 and Line 8

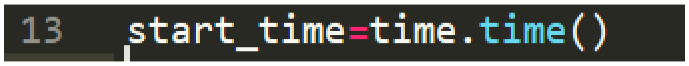
***imgA=askopenfilename(filetypes=[("image", "*")])* and**
imgB=askopenfilename(filetypes=[("image", "*")])

These lines instruct the system to open file systems for user to select the images. As mentioned above, *askopenfilename()* will prompt the system to open windows and allow users to select the files that they want.

To be specified only filetypes of images are allowed to be included, we include *filetypes=([("images","*")])* command to only allow images filetypes such as JPG, JPEG, PNG and much more only to be included. This will prevent and stop users from select other filetypes which are not from images filetypes. For this system, there is no image filetypes being specified. So, users can include any image filetypes as the input. The selected images will be saved as *imgA* and *imgB* as the system needs a name to store the image and to use as a reference.

Line 10 and Line 11
imageA = cv2.imread(imgA)
imageB = cv2.imread(imgB)

After that, the system will read the inputted images that been selected by users. The read images will be saved into a name which is first image (imgA) saved as imageA and second image (imgB) saved as imageB.

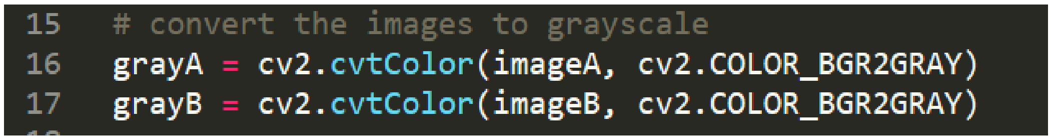


```
13   start_time=time.time()
```

Figure 5.4 : Starting Time (Line 13)

Explanation of Figure 5.4
Line 13 ***Start_time=time.time()***

This coding line is where the time module will start the timing to calculate the running process.



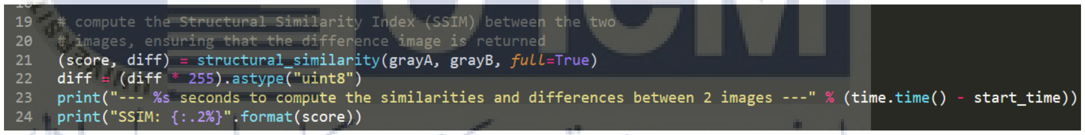
```
15   # convert the images to grayscale
16   grayA = cv2.cvtColor(imageA, cv2.COLOR_BGR2GRAY)
17   grayB = cv2.cvtColor(imageB, cv2.COLOR_BGR2GRAY)
```

Figure 5.5 : Conversion to Grayscale images (Line 15 – 17)

Explanation of Figure 5.5

Line 16 and Line 17
grayA = cv2.cvtColor(imageA, cv2.COLOR_BGR2GRAY)
grayB = cv2.cvtColor(imageB, cv2.COLOR_BGR2GRAY)

This coding line is being used to convert the colored images (BGR) into grayscale. It is because a grayscale image can simplify the algorithm and reduce the process of computational requirements. The grayscale converted images are stored in name *grayA* and *grayB* respectively.



```
19   # compute the Structural Similarity Index (SSIM) between the two
20   # images, ensuring that the difference image is returned
21   (score, diff) = structural_similarity(grayA, grayB, full=True)
22   diff = (diff * 255).astype("uint8")
23   print("--- %s seconds to compute the similarities and differences between 2 images ---" % (time.time() - start_time))
24   print("SSIM: {:.2%}".format(score))
```

Figure 5.6 : Score Calculation (Line 19 – 24)

Explanation of Figure 5.6

Line 21 ***(score, diff) = compare_ssim(grayA, grayB, full=True)***

Now it is time to compute the structural similarity index between the two input images. So, the system will calculate the scores of similarities of two images. The *score* is representing the structural similarity index between the two inputted images. As mentioned in previous chapter, the score range will be on between -1 to 1 where -1 is indicating that the two images are completely different and 1 indicating the two images are completely similar. This command line will do the process to compute the SSIM for the both grayscale images and return a score.

As for *diff* command, it is differences between two images which identify the actual differences between them on which place does it differ each other in (x,y) terms. It will help the system to visualize the differences later.

Line 22 ***diff = (diff * 255).astype("uint8)***

The differences are represented as a datatype float point where the range is between 0 and 1. So, the system need to convert the array to 8 – bit unsigned integers in range of between 0 to 255 to process on visualizing the differences between two images.

Line 23
print("--- %s seconds to compute the similarities and differences between 2 images ---" % (time.time() - start_time))

This line is to print the time taken by the system to complete the image forgery detection system. % *(time.time() – start_time))* is the formula to calculate the time taken. So, it will take the current time minus the started time.

Line 24 ***print("SSIM: {}".format(score))***

This line is indicating that the system should print the score value in Line 21 in the format of scores,

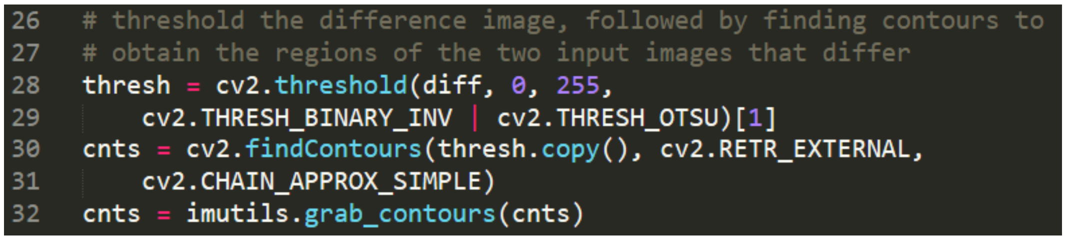

```
26  # threshold the difference image, followed by finding contours to
27  # obtain the regions of the two input images that differ
28  thresh = cv2.threshold(diff, 0, 255,
29      cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
30  cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
31      cv2.CHAIN_APPROX_SIMPLE)
32  cnts = imutils.grab_contours(cnts)
```

Figure 5.7 : Threshold Calculation (Line 26 – 32)

Explanation of Figure 5.7

Line 28 and Line 29

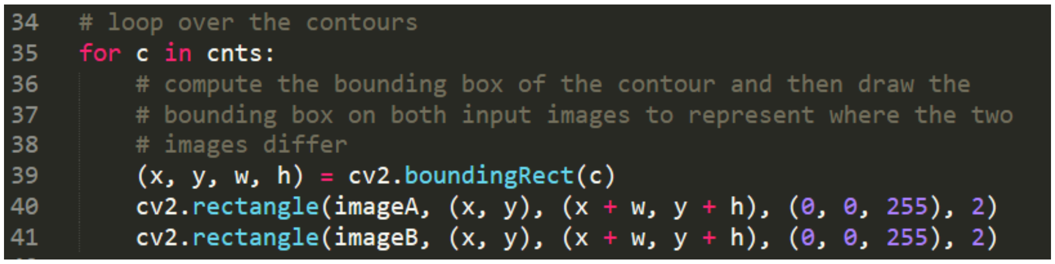
Thresh = cv2.threshold(diff, 0, 255,
cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]

This coding line is to threshold the values in *diff* . The system will apply and implement the both threshold process at the same time using the symbol " | " as can been seen in the snip of the command above. The thresholding is where the system will change the pixels of an image to be analyze easier by convert the image to binary image.

Line 30 to Line 32

cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)

This coding line is to identify and find the contours (outline of the shape) of the thresh from the command on Line 30 and Line 31. Line 32 is basically being used to adjust the result and return signature from previous line according to the versions of OpenCV that the system used so that the system do not show an error. So, the system will able to store the contours in a list without any error.



```
34  # loop over the contours
35  for c in cnts:
36      # compute the bounding box of the contour and then draw the
37      # bounding box on both input images to represent where the two
38      # images differ
39      (x, y, w, h) = cv2.boundingRect(c)
40      cv2.rectangle(imageA, (x, y), (x + w, y + h), (0, 0, 255), 2)
41      cv2.rectangle(imageB, (x, y), (x + w, y + h), (0, 0, 255), 2)
```

Figure 5.8 : Drawing Rectangle of Differences (Line 34 – 41)

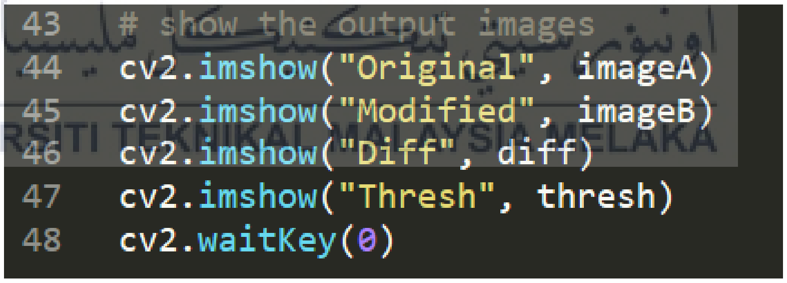
Explanation of Figure 5.8

Line 35 to Line 41
For c in cnts:

```
        (x, y, w, h) = cv2.boundingRect(c)
        cv2.rectangle(imageA, (x, y), (x + w, y + h), (0, 0, 255), 2)
        cv2.rectangle(imageB, (x, y), (x + w, y + h), (0, 0, 255), 2)
```

This coding lines are being used to draw a box around the image differences. Here the system will do looping over the contours from the previous command line. So, the system will draw the bounding rectangle box around the contour using the *cv2.boundingRect* function. So, the box will around the relevant coordinates in terms of (x, y) and x and y as the width or height of the rectangle box as w and h.

Then, the coding on Line 40 and Line 41 will instruct the system to draw a rectangle box (red color) using command *cv2.rectangle* . So, once the drawing has been done. The user can clearly see the differences between two images where the differences will be bounded with rectangles.



```
43   # show the output images
44   cv2.imshow("Original", imageA)
45   cv2.imshow("Modified", imageB)
46   cv2.imshow("Diff", diff)
47   cv2.imshow("Thresh", thresh)
48   cv2.waitKey(0)
```

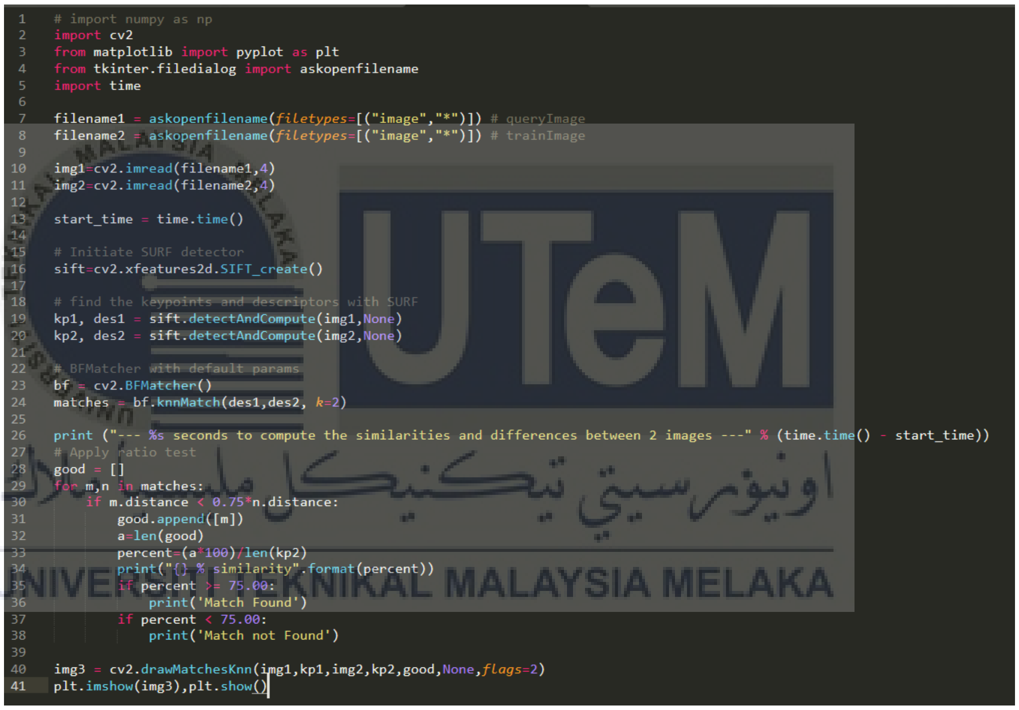
Figure 5.9 : Result Showing (Line 43 – 38)

Explanation of Figure 5.9

Line 44 to Line 48
cv2.imshow("Original", imageA)
cv2.imshow("Modified", imageB)
cv2.imshow("Diff", diff)
cv2.imshow("Thresh", thresh)
cv2.waitKey(0)

This coding lines are basically being used to show the result of the process. Finally, the system will show the 4 windows where for original image, modified image, differences value and threshold values in term of grayscale image. Lastly, command on Line 48 is instruct the system to wait until a key is pressed in the command prompt where it will exit the system.

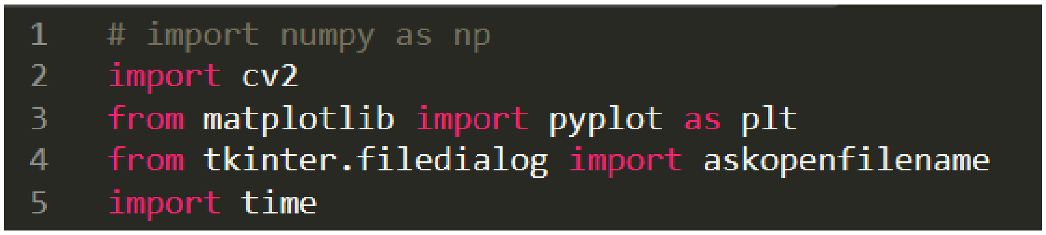
ii. Scale – Invariant Feature Transform (SIFT)



```
# import numpy as np
import cv2
from matplotlib import pyplot as plt
from tkinter.filedialog import askopenfilename
import time

filename1 = askopenfilename(filetypes=[("image","*")]) # queryImage
filename2 = askopenfilename(filetypes=[("image","*")]) # trainImage

img1=cv2.imread(filename1,4)
img2=cv2.imread(filename2,4)

start_time = time.time()

# Initiate SURF detector
sift=cv2.xfeatures2d.SIFT_create()

# find the keypoints and descriptors with SURF
kp1, des1 = sift.detectAndCompute(img1,None)
kp2, des2 = sift.detectAndCompute(img2,None)

# BFMatcher with default params
bf = cv2.BFMatcher()
matches = bf.knnMatch(des1,des2, k=2)

print ("--- %s seconds to compute the similarities and differences between 2 images ---" % (time.time() - start_time))
# Apply ratio test
good = []
for m,n in matches:
    if m.distance < 0.75*n.distance:
        good.append([m])
        a=len(good)
        percent=(a*100)/len(kp2)
        print("{} % similarity".format(percent))
        if percent >= 75.00:
            print('Match Found')
        if percent < 75.00:
            print('Match not Found')

img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,good,None,flags=2)
plt.imshow(img3),plt.show()
```

Figure 5.10 : SIFT Full Coding

First part of the coding. (Header)



```
1    # import numpy as np
2    import cv2
3    from matplotlib import pyplot as plt
4    from tkinter.filedialog import askopenfilename
5    import time
```

Figure 5.11 : Libraries Inclusion (Line 1 – 5)

The system needs to include the algorithm functions, libraries and some modules to work perfectly.

Explanation of Figure 5.11

Line 2 ***import cv2***

This is a library of Python where it is use for works such as computer vision and image processing. The full name of *cv2* is Open-CV. This is mostly used in Python for computer vision works purposes. For example, to read or display the images in the system. The system will use *cv2.imread()* to read the image that been specified.

Line 3 ***from matplotlib import pyplot as plt***

This is a module of functions where it is use to visualize the figures such as open the figures in the plot area, create plot area and much more. So, it will represent the figures in the form of x and y axis and labels so that the users can see the differences between two images clearly.

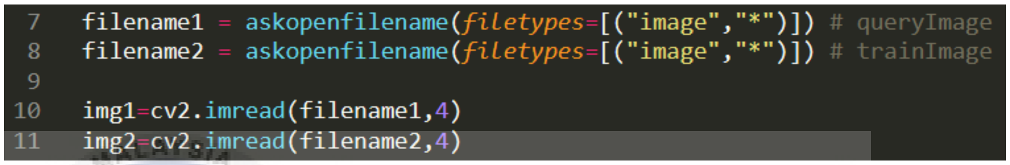
Line 4 ***from tkinter.filedialog import askopenfilename***

This is a module where it deals with file system. Since our system will prompt and ask users to choose two images for comparison from the windows file system. So, this module will provide the dialog that allows for the file (image) selections. This module is also allow the users to select more than one images and return the selected files (images) to the system for processing. Once the users selected the images, *open()* method will allow and open them.

Line 5 ***import time***

This is a module is consisting of functions that are related to time. This module is been used to calculate the running time of the system. So, it is able to calculate how long will the program will be taken to run the whole system.

Middle part of the coding (Body)



```
filename1 = askopenfilename(filetypes=[("image","*")]) # queryImage
filename2 = askopenfilename(filetypes=[("image","*")]) # trainImage

img1=cv2.imread(filename1,4)
img2=cv2.imread(filename2,4)
```

Figure 5.12 : User Input Images (Line 7 – 11)

Explanation of Figure 5.12

Line 7 and Line 8
***filename1=askopenfilename(filetypes=[("image", "*")])* and**
filename2=askopenfilename(filetypes=[("image", "*")])

These lines instruct the system to open file systems for user to select the images. As mentioned above, *askopenfilename()* will prompt the system to open windows and allow users to select the files that they want.

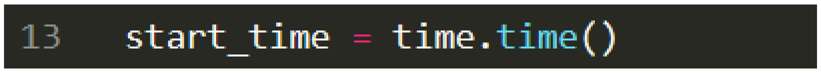
To be specified only filetypes of images are allowed to be included, we include *filetypes=([("images","*")])* command to only allow images filetypes such as JPG, JPEG, PNG and much more only to be included. This will prevent and stop users from select other filetypes which are not from images filetypes. For this system, there is no image filetypes being specified. So, users can include any image filetypes as the input. The selected images will be saved as *filename1* and *filename2* as the system needs a name to store the image and to use as a reference.

Line 10 and Line 11

img1 = cv2.imread(filename1, 4)

img2 = cv2.imread(filename2, 4)

After that, the system will read the inputted images that been selected by users. The read images will be saved into a name which is first image (filename1) saved as img1 and second image (filename2) saved as img2.



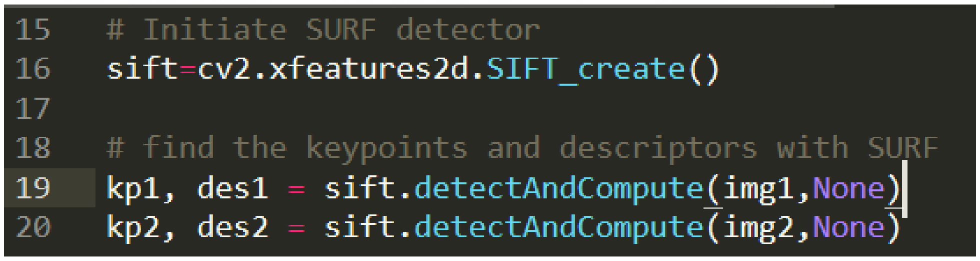
```
13   start_time = time.time()
```

Figure 5.13 : Starting Time (Line 13)

Explanation of Figure 5.13

Line 13 ***Start_time=time.time()***

This coding line is where the time module will start the timing to calculate the running process.



```
15   # Initiate SURF detector
16   sift=cv2.xfeatures2d.SIFT_create()
17
18   # find the keypoints and descriptors with SURF
19   kp1, des1 = sift.detectAndCompute(img1,None)
20   kp2, des2 = sift.detectAndCompute(img2,None)
```

Figure 5.14 : SIFT Detection Process (Line 15 – 20)

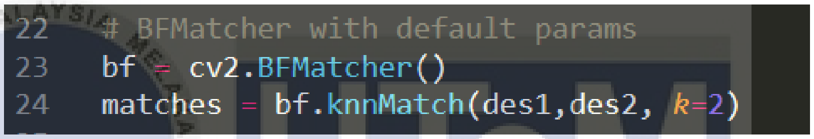
Explanation of Figure 5.14

Line 16 ***sift=cv2.xfeature2d.SIFT_create()***

This coding line is being used to create the detector and initiate the SIFT algorithm and the features of it.

Line 19 and Line 20
kp1, des1 = sift.detectAndCompute(img1,None)
kp2, des2 = sift.detectAndCompute(img2,None)

This coding lines instruct the system to detect the similarities and differences between two images using SIFT methods and find for the keypoints and descriptors using SIFT**.**



```
22   # BFMatcher with default params
23   bf = cv2.BFMatcher()
24   matches = bf.knnMatch(des1,des2, k=2)
```

Figure 5.15 : Results Matches (Line 22 – 24)

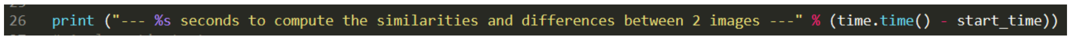
Explanation of Figure 5.15

Line 23 ***bf = cv2.BFMatcher()***

This coding lines are used to load the BFMatcher where it is one of the methods that being used to find the matches that occur between the descriptors of the two images.

Line 24 ***matches = bf.knnMatch(des1,des2, k=2)***

This line is to find the matches that occur between two images and the system will store the matches that found inside the array called *matches.* So, inside the array, there will be containing many matches as well as the false matches.



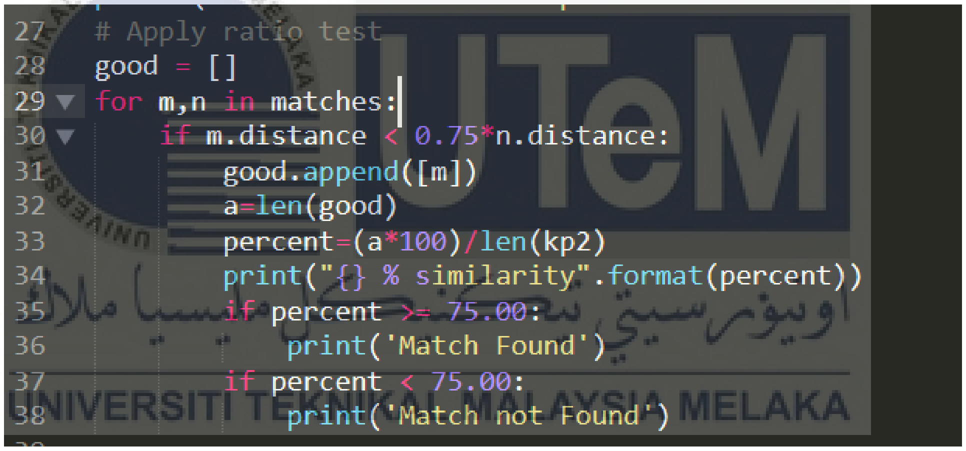
```
26  print ("--- %s seconds to compute the similarities and differences between 2 images ---" % (time.time() - start_time))
```

Figure 5.16 : Print Running time (Line 26)

Explanation of Figure 5.16

Line 26
print ("--- %s seconds to compute the similarities and differences between 2 images ---" % (time.time() - start_time))

This line is to print the time taken by the system to complete the image forgery detection system. % *(time.time() – start_time))* is the formula to calculate the time taken. So, it will take the current time minus the started time.



```
27  # Apply ratio test
28  good = []
29  for m,n in matches:
30      if m.distance < 0.75*n.distance:
31          good.append([m])
32          a=len(good)
33          percent=(a*100)/len(kp2)
34          print("{} % similarity".format(percent))
35          if percent >= 75.00:
36              print('Match Found')
37          if percent < 75.00:
38              print('Match not Found')
```

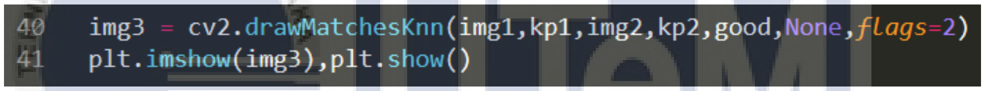
Figure 5.17 : Choose Good Matches (Line 27 – 38)

Explanation of Figure 5.17

Line 28 to Line 38
```
good = []
for m.n in matches:
    if m.distance < 0.75*n.distance:
        good.append([m])
        a=len(good)
        percent=(a*100)/len(kp2)
        print("{} % similarity". Format(percent))
        if percent >= 75.00:
                print('Match Found')
```

if percent < 75.00:
    print('Match not Found')

This coding line is where the system will apply for ratio test. This is a test where it is performed to make the system choose only the good matches that occur between two images. It is because it is the distance between the matches makes the quality of the matches. So, the lower the number, the more is the feature similarity. The ratio test can make decision to make sure the system to take only the lower distance matches and it will lead to a high-quality match. Low ratio value will allow the system to get high quality matches but as the result only few matches only will be shown. So, it is a good idea to give an average ratio where the ratio will be **0.75 .**



```
img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,good,None,flags=2)
plt.imshow(img3),plt.show()
```

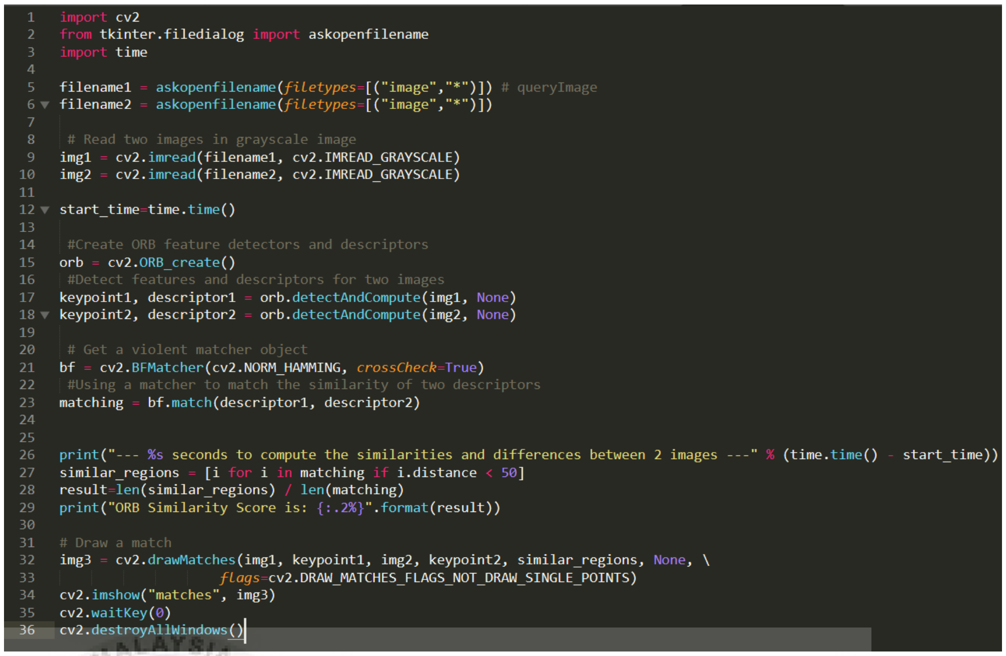
Figure 5.18 : Result Showing (Line 40 – 41)

Explanation of Figure 5.18

Line 40 and Line 41
img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,good,None,flags=2)
plt.imshow(img3),plt.show()

This coding lines are included to show the result of matching between two images. The results will be stored in a variable name img3 and the system will show the similarities between two images in a plot diagram.

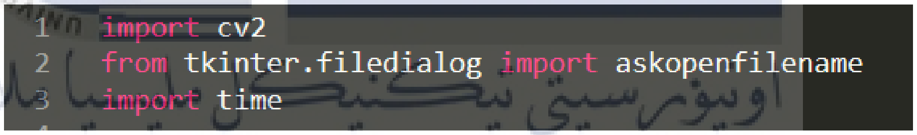
iii. Oriented FAST and Rotated BRIEF (ORB)



```
1   import cv2
2   from tkinter.filedialog import askopenfilename
3   import time
4
5   filename1 = askopenfilename(filetypes=[("image","*")]) # queryImage
6   filename2 = askopenfilename(filetypes=[("image","*")])
7
8    # Read two images in grayscale image
9   img1 = cv2.imread(filename1, cv2.IMREAD_GRAYSCALE)
10  img2 = cv2.imread(filename2, cv2.IMREAD_GRAYSCALE)
11
12  start_time=time.time()
13
14   #Create ORB feature detectors and descriptors
15  orb = cv2.ORB_create()
16   #Detect features and descriptors for two images
17  keypoint1, descriptor1 = orb.detectAndCompute(img1, None)
18  keypoint2, descriptor2 = orb.detectAndCompute(img2, None)
19
20   # Get a violent matcher object
21  bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
22   #Using a matcher to match the similarity of two descriptors
23  matching = bf.match(descriptor1, descriptor2)
24
25
26  print("--- %s seconds to compute the similarities and differences between 2 images ---" % (time.time() - start_time))
27  similar_regions = [i for i in matching if i.distance < 50]
28  result=len(similar_regions) / len(matching)
29  print("ORB Similarity Score is: {:.2%}".format(result))
30
31  # Draw a match
32  img3 = cv2.drawMatches(img1, keypoint1, img2, keypoint2, similar_regions, None, \
33                      flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
34  cv2.imshow("matches", img3)
35  cv2.waitKey(0)
36  cv2.destroyAllWindows()
```

Figure 5.19 : ORB Full Coding

First part of the coding (Header)



```
1   import cv2
2   from tkinter.filedialog import askopenfilename
3   import time
```

Figure 5.20 : Libraries Inclusion (Line 1- 3)

Explanation of Figure 5.20

Line 1 ***import cv2***

This is a library of Python where it is use for works such as computer vision and image processing. The full name of *cv2* is Open-CV. This is mostly used in Python for computer vision works purposes. For example, to read or display the images in the system. The system will use *cv2.imread()* to read the image that been specified.

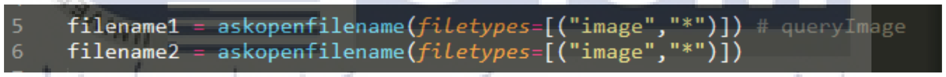
Line 2 ***from tkinter.filedialog import askopenfilename***

This is a module where it deals with file system. Since our system will prompt and ask users to choose two images for comparison from the windows file system. So, this module will provide the dialog that allows for the file (image) selections. This module is also allow the users to select more than one images and return the selected files (images) to the system for processing. Once the users selected the images, *open()* method will allow and open them.

Line 3 ***import time***

This is a module is consisting of functions that are related to time. This module is been used to calculate the running time of the system. So, it is able to calculate how long will the program will be taken to run the whole system.

Second part of the coding (Body)



```
5   filename1 = askopenfilename(filetypes=[("image","*")]) # queryImage
6   filename2 = askopenfilename(filetypes=[("image","*")])
```

Figure 5.21 : User Input Images (Line 5 – 6)

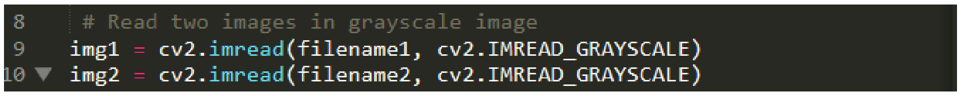
Explanation of Figure 5.21

Line 5 and Line 6

filename1=askopenfilename(filetypes=[("image", "*")]) and filename2=askopenfilename(filetypes=[("image", "*")])

These lines instruct the system to open file systems for user to select the images. As mentioned above, *askopenfilename()* will prompt the system to open windows and allow users to select the files that they want.

To be specified only filetypes of images are allowed to be included, we include *filetypes=([("images","*")])* command to only allow images filetypes such as JPG, JPEG, PNG and much more only to be included. This will prevent and stop users from select other filetypes which are not from images filetypes. For this system, there is no image filetypes being specified. So, users can include any image filetypes as the input. The selected images will be saved as *filename1* and *filename2* as the system needs a name to store the image and to use as a reference.



```
8     # Read two images in grayscale image
9     img1 = cv2.imread(filename1, cv2.IMREAD_GRAYSCALE)
10 ▼  img2 = cv2.imread(filename2, cv2.IMREAD_GRAYSCALE)
```

Figure 5.22 : Changing to Grayscale Images (Line 8 – 10)

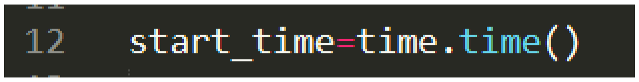
Explanation of Figure 5.22

Line 9 and Line 10

img1 = cv2.imread(filename1, cv2.IMREAD_GRAYSCALE)

img2 = cv2.imread(filename2, cv2.IMREAD_GRAYSCALE)

After that, the system will read the inputted images that been selected by users. The read images will be saved into a name which is first image (filename1) saved as img1 and second image (filename2) saved as img2. The images will converted into grayscale before saving into the img1 and img2 respectively. It is because the algorithm needs grayscale images to proceed.



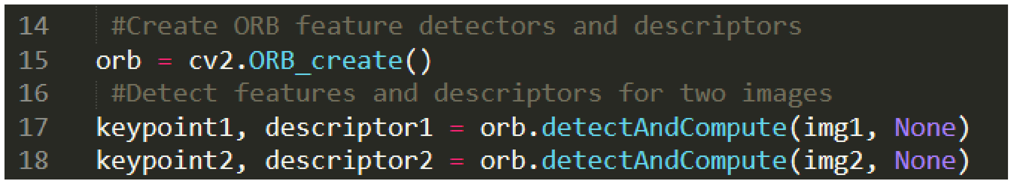
```
12    start_time=time.time()
```

Figure 5.23 : Starting Time (Line 12)

Explanation of Figure 5.23

Line 12 ***Start_time=time.time()***

This coding line is where the time module will start the timing to calculate the running process.



```
14    #Create ORB feature detectors and descriptors
15   orb = cv2.ORB_create()
16    #Detect features and descriptors for two images
17   keypoint1, descriptor1 = orb.detectAndCompute(img1, None)
18   keypoint2, descriptor2 = orb.detectAndCompute(img2, None)
```

Figure 5.24 : ORB Findings Process (Line 14 – 18)

Explanation of Figure 5.24

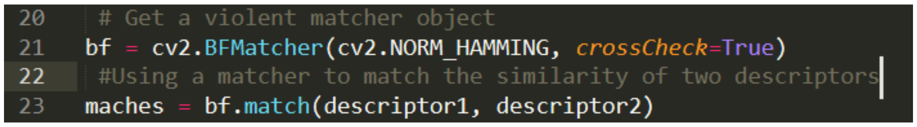
Line 15 ***orb = cv2.ORB_create()***

This coding line is being used to create the detector and initiate the ORB algorithm and the features of it.

Line 17 and Line 18

keypoint1, descriptor1 = orb.detectAndCompute(img1,None)
keypoint2, descriptor2 = orb.detectAndCompute(img2,None)

This coding lines instruct the system to detect the similarities and differences between two images using **ORB** methods and find for the keypoints and descriptors using **ORB.**



```
20    # Get a violent matcher object
21   bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
22    #Using a matcher to match the similarity of two descriptors
23   maches = bf.match(descriptor1, descriptor2)
```

Figure 5.25 : Results Matching (Line 20 – 23)

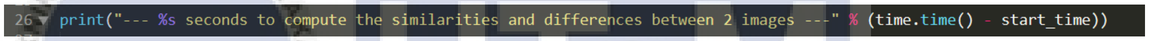
Explanation of Figure 5.25

Line 21 ***bf = cv2.BfMatcher(cv2.NORM_HAMMING, crossCheck=True)***

This coding lines are used to load the BFMatcher where it is one of the methods that being used to find the matches that occur between the descriptors of the two images using distance calculation.

Line 23 ***maches = bf.match(descriptor1,descriptor2)***

This line is to find the matches that occur between two images and the system will store the maches that found inside the array called *maches*. So, inside the array, there will be containing many matches as well as the false matches.



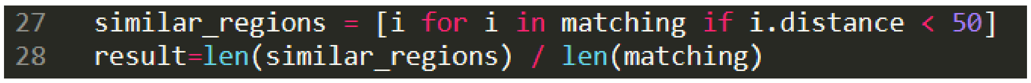
```
26  print("--- %s seconds to compute the similarities and differences between 2 images ---" % (time.time() - start_time))
```

Figure 5.26 : Time Printout (Line 26)

Explanation of Figure 5.26

Line 26
print ("--- %s seconds to compute the similarities and differences between 2 images ---" % (time.time() - start_time))

This line is to print the time taken by the system to complete the image forgery detection system. % *(time.time() – start_time))* is the formula to calculate the time taken. So, it will take the current time minus the started time.



```
27  similar_regions = [i for i in matching if i.distance < 50]
28  result=len(similar_regions) / len(matching)
```

Figure 5.27 : Similar Region scores (Line 27 – 28)

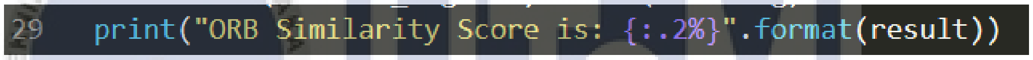
Explanation of Figure 5.27

Line 27 to Line 28

similar_region = [i for i in matching if i.distance < 50]

result=len(similar_regions) / lens(matching)

This line is telling the system to find out the similar regions in matching descriptors between two images. The results will be stored and passed to variable name *similar_region* if the similarities matching descriptor's distances are below than 50. The found similarities descriptors will be calculated with the formula shown in Figure 5.27 (Line 28 the found results will be saved under variable name *result.*



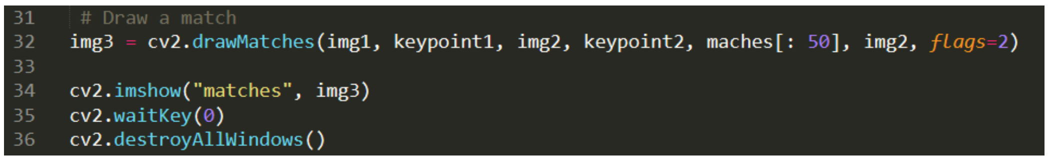
```
29    print("ORB Similarity Score is: {:.2%}".format(result))
```

Figure 5.28 : Result Print Out (Line 29)

Explanation of Figure 5.28

Line 29 ***print("ORB Similarity Score is: {:.2}".format(result))***

Line 29 is use for print the calculated result on Line 28 with 2 decimal places.



```
31    # Draw a match
32    img3 = cv2.drawMatches(img1, keypoint1, img2, keypoint2, maches[: 50], img2, flags=2)
33
34    cv2.imshow("matches", img3)
35    cv2.waitKey(0)
36    cv2.destroyAllWindows()
```

Figure 5.29 : Result Showing (Line 31 – 36)

Explanation of Figure 5.28

Line 32

img3 = cv2.drawMatches(img1, keypoint1, img2, keypoint2, maches[: 50], img2, flags=2)

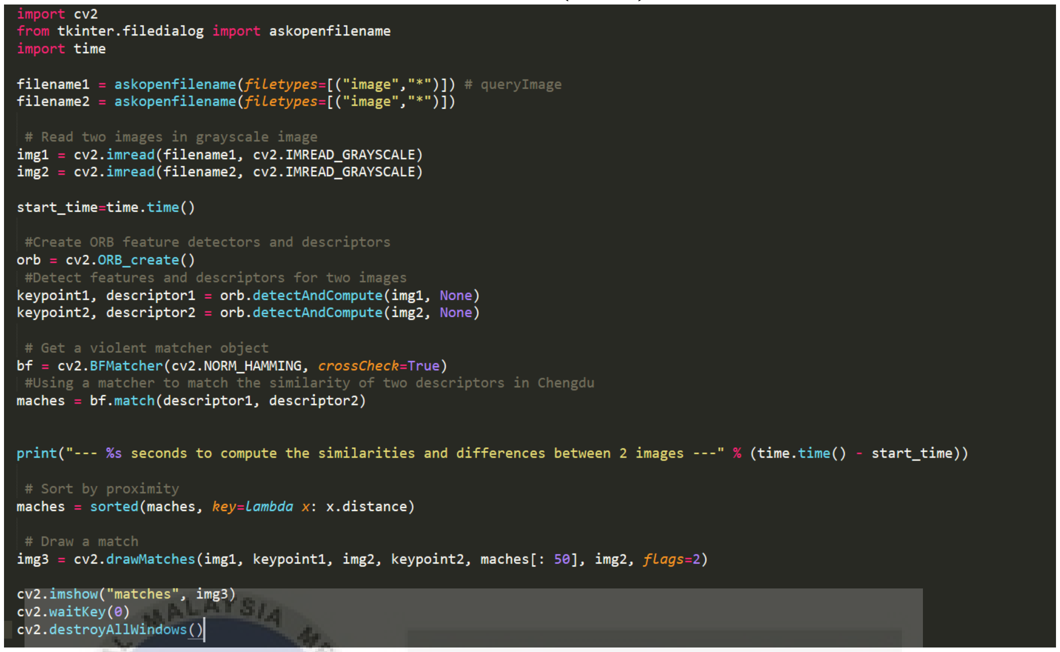
This coding line is basically the system to draw around the keypoints that it found similarities between two images. The found result will be stored in variable name *img3* .

Line 34 to Line 36

cv2.imshow("matches", img3)
cv2.waitKey(0)
cv2.destroyAllWindows()

This command lines are telling the system to show the result which is *img3*. Lastly, command on **Line 29** is instruct the system to wait until a key is pressed in the command prompt where it will exit the system as per in **Line 30.**

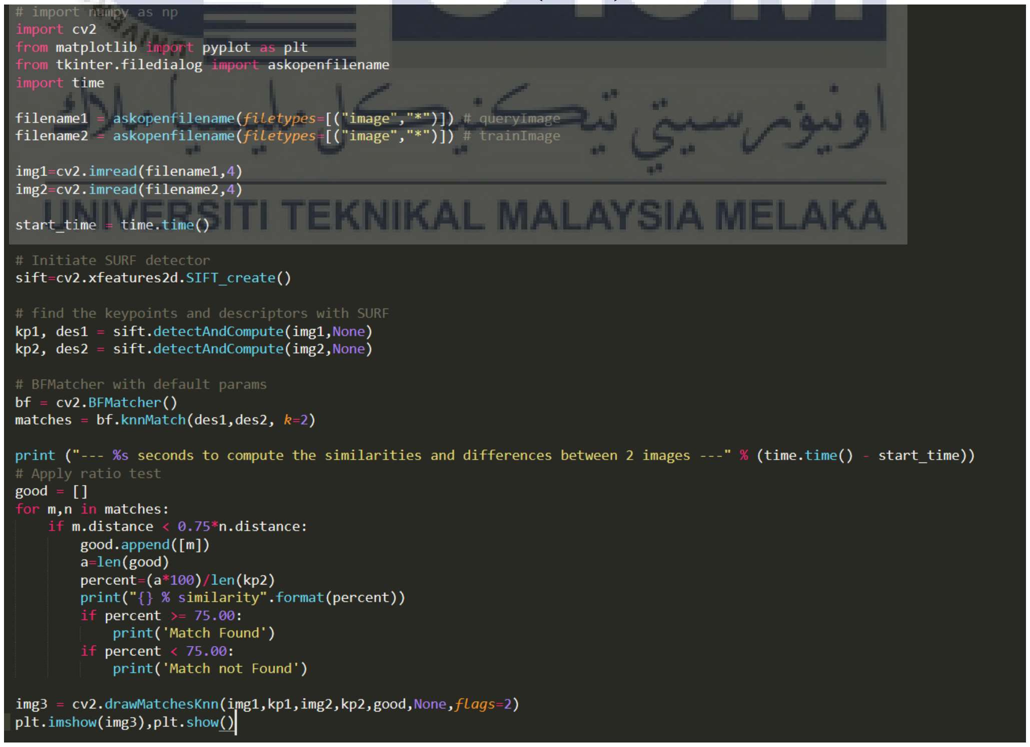
5.3.1 Oriented FAST and Rotated BRIEF (ORB)



```
import cv2
from tkinter.filedialog import askopenfilename
import time

filename1 = askopenfilename(filetypes=[("image","*")]) # queryImage
filename2 = askopenfilename(filetypes=[("image","*")])

 # Read two images in grayscale image
img1 = cv2.imread(filename1, cv2.IMREAD_GRAYSCALE)
img2 = cv2.imread(filename2, cv2.IMREAD_GRAYSCALE)

start_time=time.time()

 #Create ORB feature detectors and descriptors
orb = cv2.ORB_create()
 #Detect features and descriptors for two images
keypoint1, descriptor1 = orb.detectAndCompute(img1, None)
keypoint2, descriptor2 = orb.detectAndCompute(img2, None)

 # Get a violent matcher object
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
 #Using a matcher to match the similarity of two descriptors in Chengdu
maches = bf.match(descriptor1, descriptor2)


print("--- %s seconds to compute the similarities and differences between 2 images ---" % (time.time() - start_time))

 # Sort by proximity
maches = sorted(maches, key=lambda x: x.distance)

 # Draw a match
img3 = cv2.drawMatches(img1, keypoint1, img2, keypoint2, maches[: 50], img2, flags=2)

cv2.imshow("matches", img3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figure 5.30 : Oriented FAST and Rotated BRIEF Full Coding

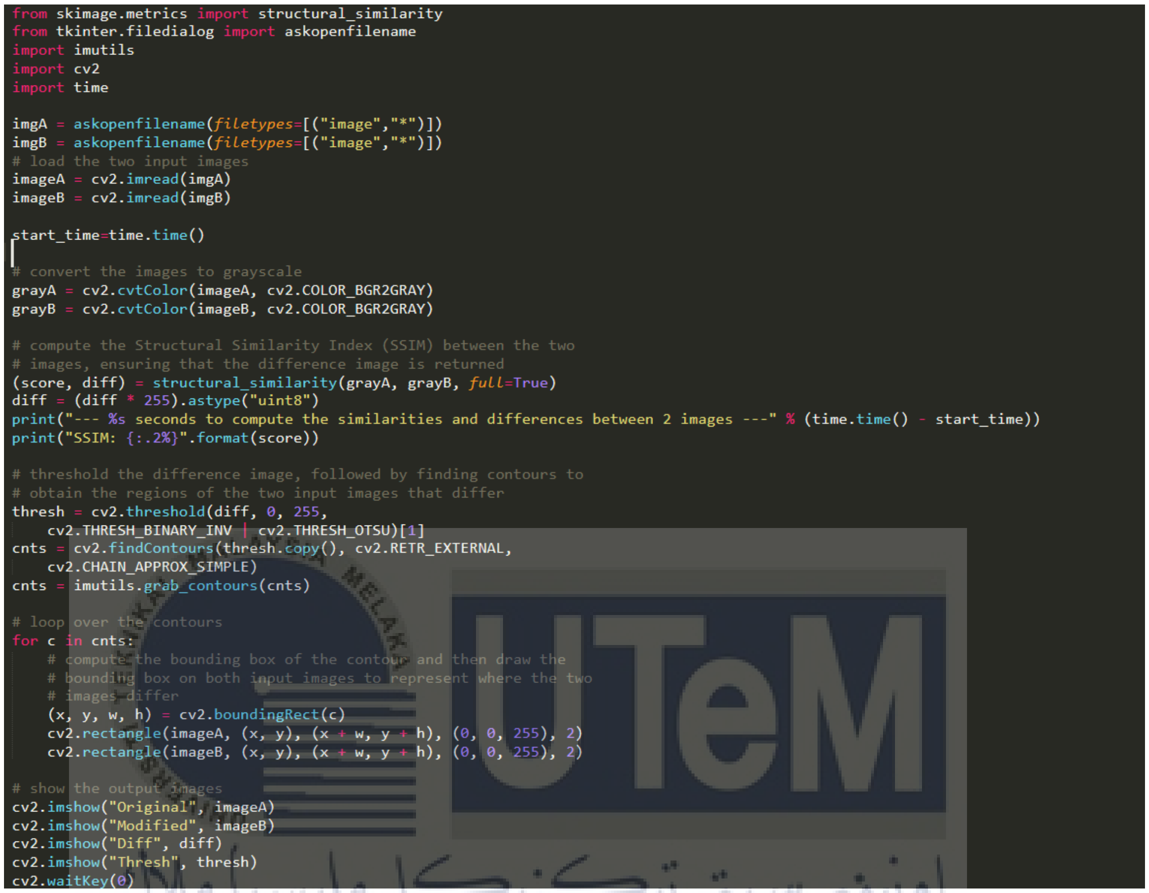
5.3.2 Scale-Invariant Feature Transform (SIFT)



```
# import numpy as np
import cv2
from matplotlib import pyplot as plt
from tkinter.filedialog import askopenfilename
import time

filename1 = askopenfilename(filetypes=[("image","*")]) # queryimage
filename2 = askopenfilename(filetypes=[("image","*")]) # trainImage

img1=cv2.imread(filename1,4)
img2=cv2.imread(filename2,4)

start_time = time.time()

# Initiate SURF detector
sift=cv2.xfeatures2d.SIFT_create()

# find the keypoints and descriptors with SURF
kp1, des1 = sift.detectAndCompute(img1,None)
kp2, des2 = sift.detectAndCompute(img2,None)

# BFMatcher with default params
bf = cv2.BFMatcher()
matches = bf.knnMatch(des1,des2, k=2)

print ("--- %s seconds to compute the similarities and differences between 2 images ---" % (time.time() - start_time))
# Apply ratio test
good = []
for m,n in matches:
    if m.distance < 0.75*n.distance:
        good.append([m])
        a=len(good)
        percent=(a*100)/len(kp2)
        print("{} % similarity".format(percent))
        if percent >= 75.00:
            print('Match Found')
        if percent < 75.00:
            print('Match not Found')

img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,good,None,flags=2)
plt.imshow(img3),plt.show()
```

Figure 5.31 : Scale-Invariant Feature Transform Full Coding

5.3.3 Structural Similarity Index (SSIM)



```
from skimage.metrics import structural_similarity
from tkinter.filedialog import askopenfilename
import imutils
import cv2
import time

imgA = askopenfilename(filetypes=[("image","*")])
imgB = askopenfilename(filetypes=[("image","*")])
# load the two input images
imageA = cv2.imread(imgA)
imageB = cv2.imread(imgB)

start_time=time.time()

# convert the images to grayscale
grayA = cv2.cvtColor(imageA, cv2.COLOR_BGR2GRAY)
grayB = cv2.cvtColor(imageB, cv2.COLOR_BGR2GRAY)

# compute the Structural Similarity Index (SSIM) between the two
# images, ensuring that the difference image is returned
(score, diff) = structural_similarity(grayA, grayB, full=True)
diff = (diff * 255).astype("uint8")
print("--- %s seconds to compute the similarities and differences between 2 images ---" % (time.time() - start_time))
print("SSIM: {:.2%}".format(score))

# threshold the difference image, followed by finding contours to
# obtain the regions of the two input images that differ
thresh = cv2.threshold(diff, 0, 255,
    cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)

# loop over the contours
for c in cnts:
    # compute the bounding box of the contour and then draw the
    # bounding box on both input images to represent where the two
    # images differ
    (x, y, w, h) = cv2.boundingRect(c)
    cv2.rectangle(imageA, (x, y), (x + w, y + h), (0, 0, 255), 2)
    cv2.rectangle(imageB, (x, y), (x + w, y + h), (0, 0, 255), 2)

# show the output images
cv2.imshow("Original", imageA)
cv2.imshow("Modified", imageB)
cv2.imshow("Diff", diff)
cv2.imshow("Thresh", thresh)
cv2.waitKey(0)
```

Figure 5.32 : Structural Similarity Index Full Coding

5.4 Conclusion

The implementation coding lines are been outlined and shown in this chapter. It covers all the coding of each algorithm and the combined algorithms. Testing from the implementation will be carried out on the next chapter.

CHAPTER 6 : TESTING

6.1 Introduction

This chapter is about the findings of the testing results of the image forgery system. These systems are tested with various categories of images such as rotated images and copy move forgery images. The accuracy and efficiency of all algorithms can be seen through the results table. The time taken for each algorithm to run the system also been calculated and will be compared among each other. Although the calculated time might be affected due to the host hardware and software, but the calculated average time can be compared roughly.

The testing is carried out using 2 categories of images. The testing image categories are rotated images and copy-move forgery images. This test is carried out to find how robust and accurate the system and the algorithms can detect the differences between two images and how efficient the systems and the algorithms can detect the various categories of images. For each test, similarities score value will be calculated and algorithm running time will be calculated to compare between each algorithm.

6.2 Rotated Images

These two images are similar but the orientation is different. The image has been rotated 90 degrees. This category of images is used to test the efficiency the image forgery detection algorithms. So, below are the images being used to test the algorithms.

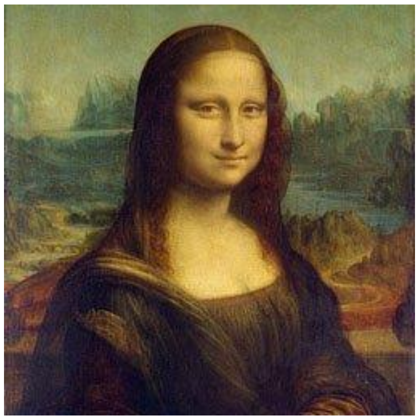


Figure 6.1 : Original Image

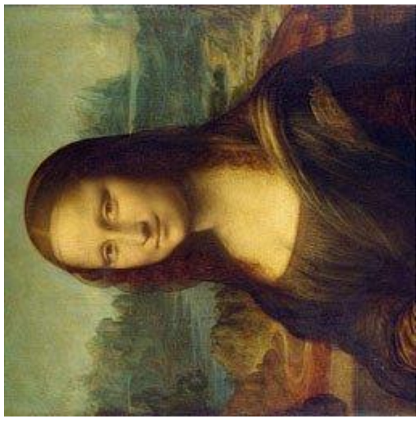


Figure 6.2 : Rotated Image

6.2.1 Results based on Similarities Scores

Table 6.1 shows the similarities score results of image forgery detection for rotated image using the 3 algorithms. The value is presented in percentage format. So, the higher the score value, the higher the similarities between two images.

Table 6.1 : Results for rotated image testing in terms of score value

Results for Rotated Image (Score Value)			
Algorithms	**SSIM**	**SIFT**	**ORB**
Similarities Algorithm Score (%)	27.30	5.76	42.74

From the Table 6.1 results, it clearly shows that ORB has the highest similarities score value compare to SSIM and SIFT which is 42.74%. Second highest score value is SSIM which is 27.30% and followed by SIFT with 5.76% score.

6.2.2 Results based on Time

Table 6.2 shows the running time results of image forgery detection for rotated image using the 3 algorithms. The value is presented in seconds format. So, the lower the time, the faster the algorithm works.

Table 6.2 : Result for rotated image testing in terms of time

Results for Rotated Image (Running time)			
Algorithms / **Tests (Seconds)**	**SSIM**	**SIFT**	**ORB**
1st Test	0.2440624237060547	0.1810441017150879	0.7329912185668945
2nd Test	0.22804522514343262	0.18003129959106445	0.7211823463439941
3rd Test	0.21405720710754395	0.1720414161682129	0.7034354209899902
Average	**0.22872161865234376**	**0.17770560582478842**	**0.7192029953002929**

From the Table 6.2 results, it clearly shows that ORB has the longest average running time compare to SSIM and SIFT which is 0.72 seconds. Second longest average running time is SSIM which is 0.23 seconds and followed by SIFT with 0.18 seconds.

6.2.3 Testing results images (System Console)

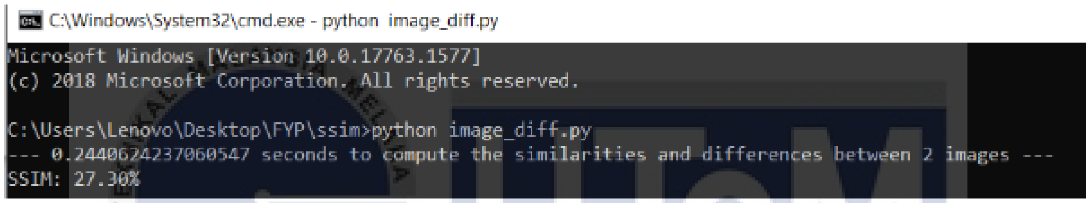
3 algorithms' testing snapshot has been shown in following section. Snapshot of the system console and area of similarities detection have been included. This is to clearly shows the score value and running time taken by each algorithm. 3 tests have been done

for each algorithm to find out an average time taken. So, there will be 3 system console snippet and 1 figure of similarities or differences detected areas.

6.2.3.1 SSIM

Figure 6.3 to Figure 6.6 are the snippets of the system console when the rotated image is tested with the SSIM.

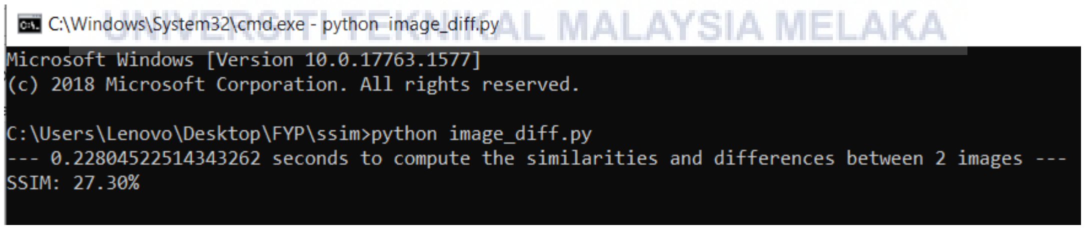
Figure 6.3 is first SSIM testing of rotated image. The score value is 27.30% and the time is 0.2440624237060547 seconds.



```
C:\Windows\System32\cmd.exe - python  image_diff.py
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\ssim>python image_diff.py
--- 0.2440624237060547 seconds to compute the similarities and differences between 2 images ---
SSIM: 27.30%
```

Figure 6.3 : Snippet of first SSIM testing time and score

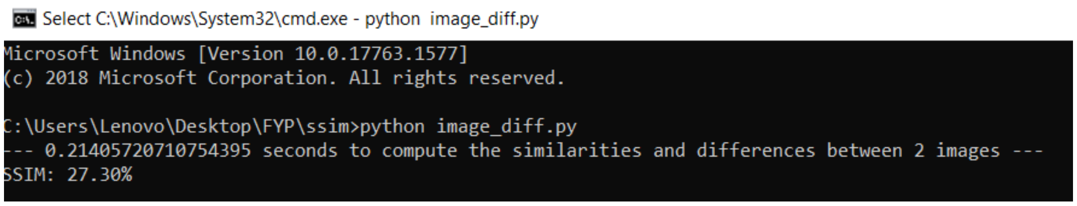
Figure 6.4 is second SSIM testing of rotated image. The score value is 27.30% and the time is 0.22804522514343262 seconds.



```
C:\Windows\System32\cmd.exe - python  image_diff.py
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\ssim>python image_diff.py
--- 0.22804522514343262 seconds to compute the similarities and differences between 2 images ---
SSIM: 27.30%
```

Figure 6.4 : Snippet of second SSIM testing time and score

Figure 6.5 is third SSIM testing of rotated image. The score value is 27.30% and the time is 0.21405720710754395 seconds.



Select C:\Windows\System32\cmd.exe - python image_diff.py

```
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\ssim>python image_diff.py
--- 0.21405720710754395 seconds to compute the similarities and differences between 2 images ---
SSIM: 27.30%
```

Figure 6.5 : Snippet of third SSIM testing time and score

Figure 6.6 is SSIM Similarities area detection of the rotated image. The red boxes are the detected region of similarities and differences area between two images. The grayscale system console are the part of the algorithm process wehre SSIM will convert the coloured images to grayscale image to identify the similarities and differences. It is more easier to identify the regions in grayscale images.

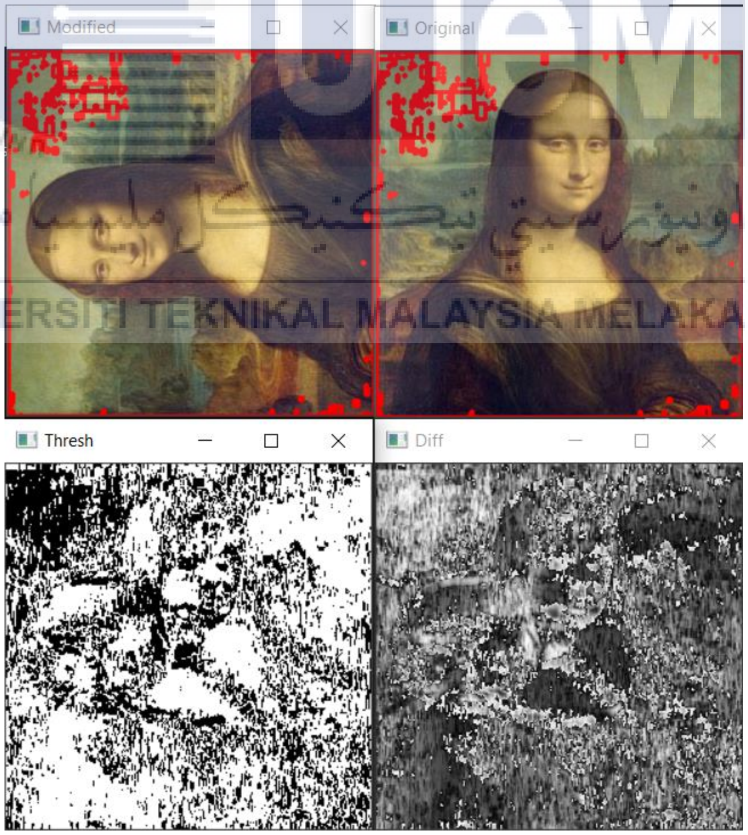
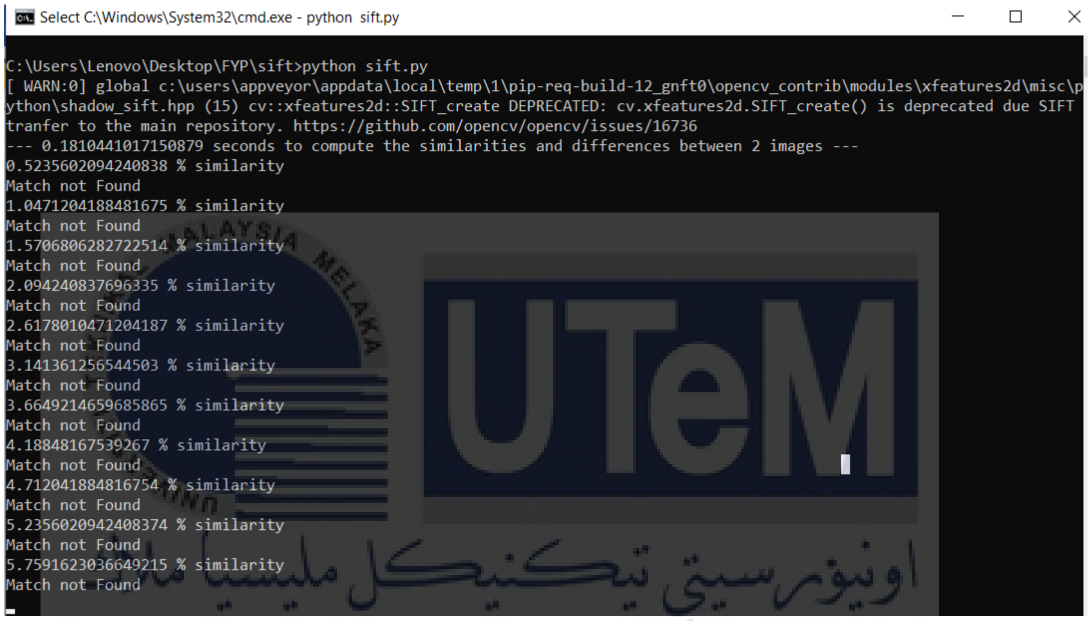


Figure 6.6 : Snippet of Similarities region from SSIM test

6.2.3.2 SIFT

Figure 6.7 to Figure 6.10 are the snippets of the system console when the rotated images are tested with the SIFT.

Figure 6.7 is first SIFT testing of rotated image. The score value is 5.76% and the time is 0.1810441017150879 seconds

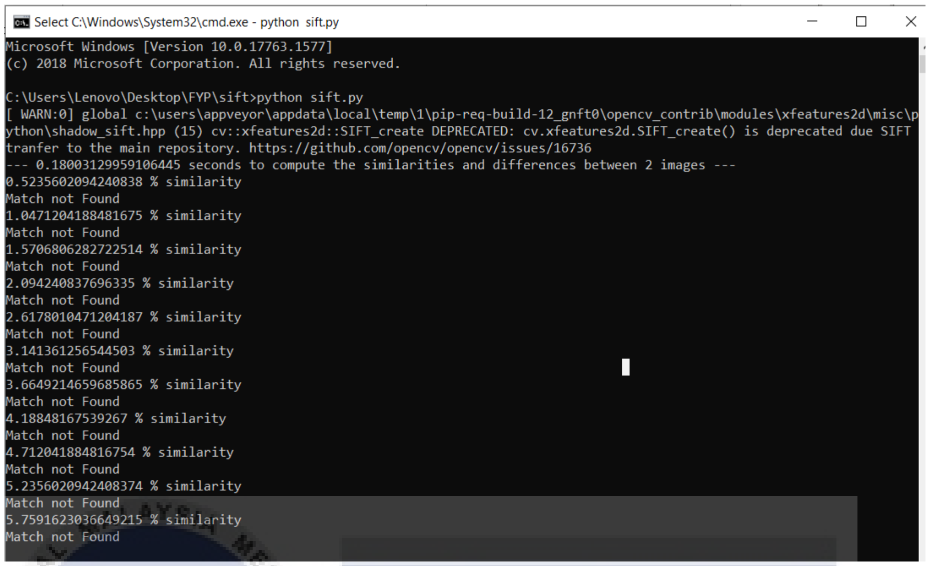


Select C:\Windows\System32\cmd.exe - python sift.py

```
C:\Users\Lenovo\Desktop\FYP\sift>python sift.py
[ WARN:0] global c:\users\appveyor\appdata\local\temp\1\pip-req-build-12_gnft0\opencv_contrib\modules\xfeatures2d\misc\p
ython\shadow_sift.hpp (15) cv::xfeatures2d::SIFT_create DEPRECATED: cv.xfeatures2d.SIFT_create() is deprecated due SIFT
tranfer to the main repository. https://github.com/opencv/opencv/issues/16736
--- 0.1810441017150879 seconds to compute the similarities and differences between 2 images ---
0.5235602094240838 % similarity
Match not Found
1.0471204188481675 % similarity
Match not Found
1.5706806282722514 % similarity
Match not Found
2.094240837696335 % similarity
Match not Found
2.6178010471204187 % similarity
Match not Found
3.141361256544503 % similarity
Match not Found
3.6649214659685865 % similarity
Match not Found
4.18848167539267 % similarity
Match not Found
4.712041884816754 % similarity
Match not Found
5.2356020942408374 % similarity
Match not Found
5.7591623036649215 % similarity
Match not Found
```

Figure 6.7 : Snippet of first SIFT testing time and score

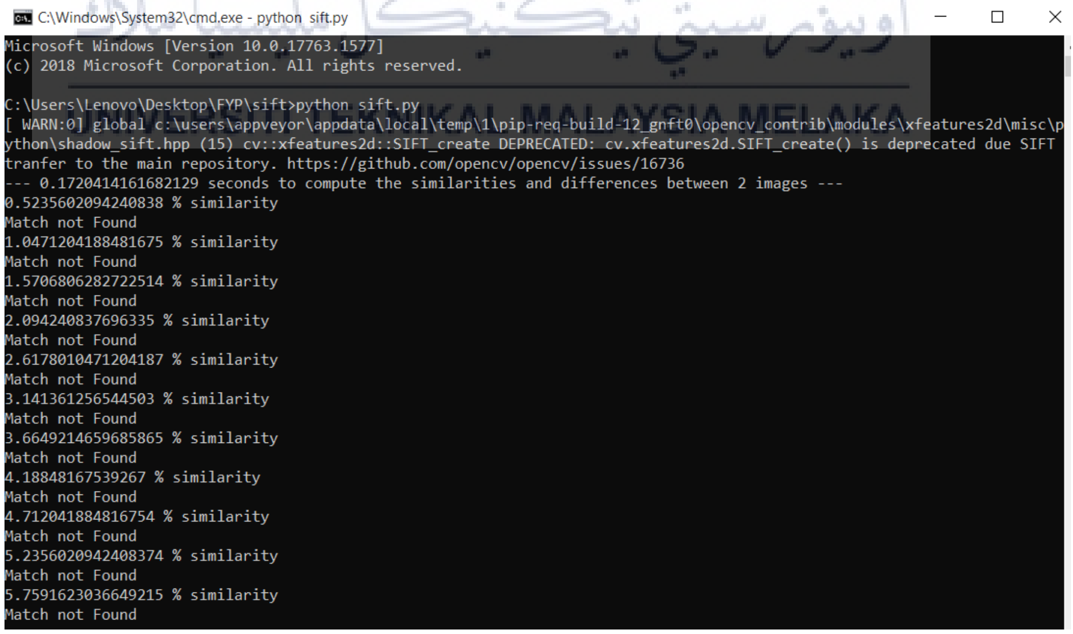
Figure 6.8 is second SIFT testing of rotated image. The score value is 5.76% and the time is 0.18003129959106445 seconds.



```
Select C:\Windows\System32\cmd.exe - python  sift.py
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\sift>python sift.py
[ WARN:0] global c:\users\appveyor\appdata\local\temp\1\pip-req-build-12_gnft0\opencv_contrib\modules\xfeatures2d\misc\p
ython\shadow_sift.hpp (15) cv::xfeatures2d::SIFT_create DEPRECATED: cv.xfeatures2d.SIFT_create() is deprecated due SIFT
tranfer to the main repository. https://github.com/opencv/opencv/issues/16736
--- 0.18003129959106445 seconds to compute the similarities and differences between 2 images ---
0.5235602094240838 % similarity
Match not Found
1.0471204188481675 % similarity
Match not Found
1.5706806282722514 % similarity
Match not Found
2.094240837696335 % similarity
Match not Found
2.6178010471204187 % similarity
Match not Found
3.141361256544503 % similarity
Match not Found
3.6649214659685865 % similarity
Match not Found
4.18848167539267 % similarity
Match not Found
4.712041884816754 % similarity
Match not Found
5.2356020942408374 % similarity
Match not Found
5.7591623036649215 % similarity
Match not Found
```

Figure 6.8 : Snippet of second SIFT testing time and score

Figure 6.9 is third SIFT testing of rotated image. The score value is 5.76% and the time is 0.1720414161682129 seconds.



```
C:\Windows\System32\cmd.exe - python  sift.py
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\sift>python sift.py
[ WARN:0] global c:\users\appveyor\appdata\local\temp\1\pip-req-build-12_gnft0\opencv_contrib\modules\xfeatures2d\misc\p
ython\shadow_sift.hpp (15) cv::xfeatures2d::SIFT_create DEPRECATED: cv.xfeatures2d.SIFT_create() is deprecated due SIFT
tranfer to the main repository. https://github.com/opencv/opencv/issues/16736
--- 0.1720414161682129 seconds to compute the similarities and differences between 2 images ---
0.5235602094240838 % similarity
Match not Found
1.0471204188481675 % similarity
Match not Found
1.5706806282722514 % similarity
Match not Found
2.094240837696335 % similarity
Match not Found
2.6178010471204187 % similarity
Match not Found
3.141361256544503 % similarity
Match not Found
3.6649214659685865 % similarity
Match not Found
4.18848167539267 % similarity
Match not Found
4.712041884816754 % similarity
Match not Found
5.2356020942408374 % similarity
Match not Found
5.7591623036649215 % similarity
Match not Found
```

Figure 6.9 : Snippet of third SIFT testing time and score

Figure 6.10 is SIFT testing for similarities area detection. The lines shown in figure 6.10 are the lines indicating similarities between two images. If the similarities value of key descriptors are more than 75% then the system will draw the lines on the descriptors between two images.

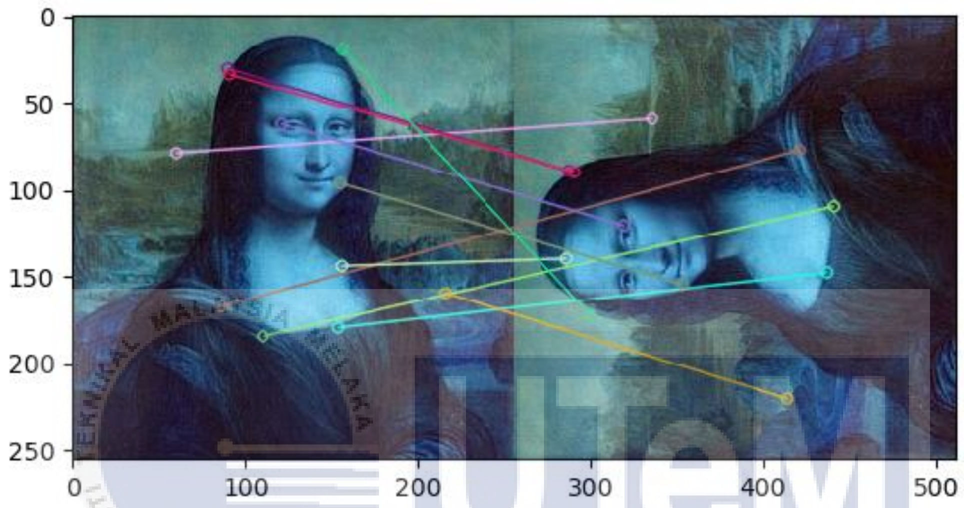
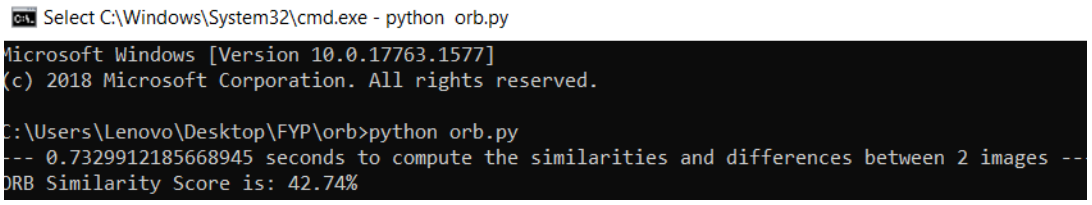


Figure 6.10 : Snippet of Similarities Region from SIFT Test

6.2.3.3 ORB

Figure 6.11 to Figure 6.14 are the snippets of the system console when the rotated image is tested with the ORB.

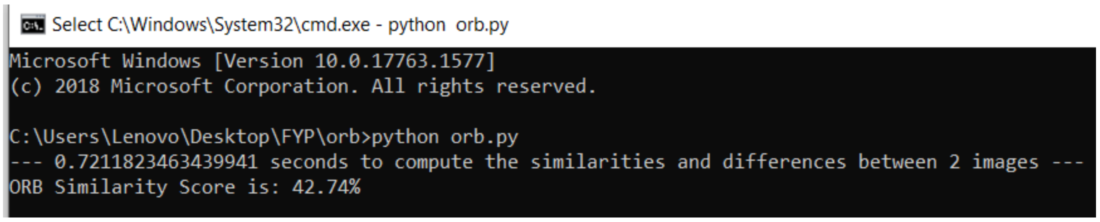
Figure 6.11 is first ORB testing of rotated image. The score value is 42.74% and the time is 0.7329912185668945 seconds.



```
Select C:\Windows\System32\cmd.exe - python  orb.py
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\orb>python orb.py
--- 0.7329912185668945 seconds to compute the similarities and differences between 2 images ---
ORB Similarity Score is: 42.74%
```

Figure 6.11 : Snippet of first ORB testing time and score

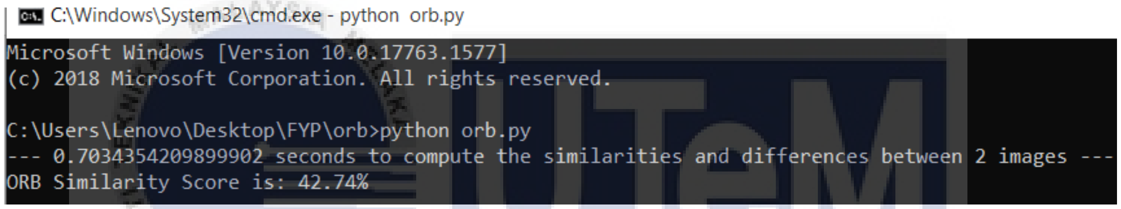
Figure 6.12 is second ORB testing of rotated image. The score value is 42.74% and the time is 0.7211823463439941 seconds.



```
Select C:\Windows\System32\cmd.exe - python  orb.py
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\orb>python orb.py
--- 0.7211823463439941 seconds to compute the similarities and differences between 2 images ---
ORB Similarity Score is: 42.74%
```

Figure 6.12 : Snippet of second ORB testing time and score

Figure 6.13 is third ORB testing of rotated image. The score value is 42.74% and the time is 0.7034354209899902 seconds.



```
C:\Windows\System32\cmd.exe - python  orb.py
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\orb>python orb.py
--- 0.7034354209899902 seconds to compute the similarities and differences between 2 images ---
ORB Similarity Score is: 42.74%
```

Figure 6.13 : Snippet of third ORB testing time and score

Figure 6.14 is ORB testing for similarities area detection. The lines shown in figure 6.14 are the lines indicating similarities found between two images. The lines are drawn based on good matching key descriptors found between two images.

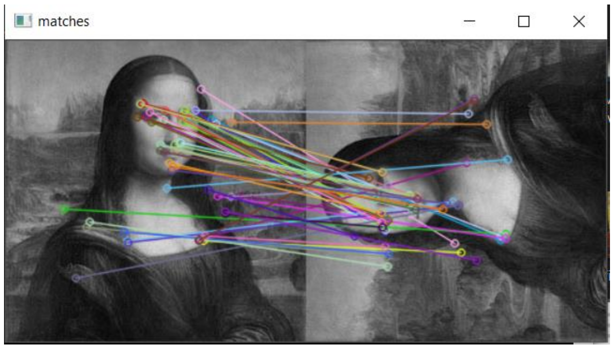


Figure 6.14 : Snippet of Similarities Region from ORB Test

6.3 Copy Move Forgery Images

These two images are almost similar but the are some differences between both images. The image has been performed copy move forgery which is there are some additional pictures or parts are included in the image. It is very hard for human to identify the differences with naked eyes. If examine closely, in the manipulated image there are some extra leaves have been pasted over the original image. This kind of images are used to test the efficiency of the image forgery detection algorithms and to test whether the algorithm can detect the similarities and differences between two images.

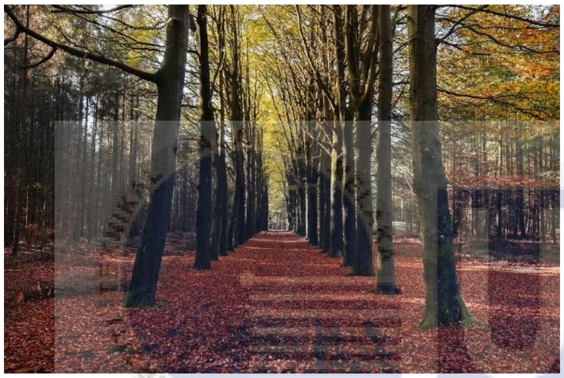


Figure 6.15 : Original Image (references)

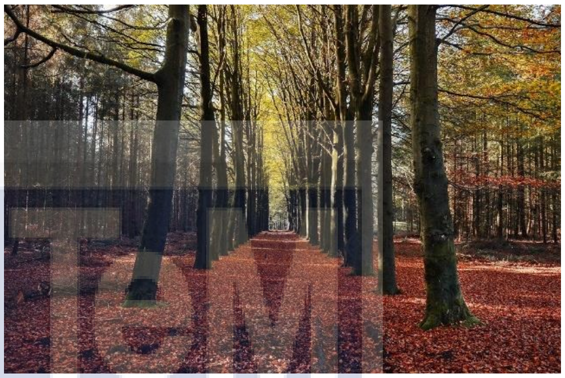


Figure 6.16 : Manipulated Image (references)

6.3.1 Results based on Similarities Scores

Table 6.3 shows the similarities score results of image forgery detection for copy move forgery image using the 3 algorithms. The value is presented in percentage format. So, the higher the score value, the higher the similarities between two images.

Table 6.3 : Results for copy move forgery image testing in terms of score value

Results for Copy-Move Forgery Images (Score Value)			
Algorithms	SSIM	SIFT	ORB
Similarities Algorithm Score (%)	96.82	77.77	99.57

From the Table 6.3 results, it clearly shows that ORB has the highest similarities score value compare to SSIM and SIFT which is 99.57%. Second highest score value is SSIM which is 96.82% and followed by SIFT with 77.77% score.

Table 6.4 shows the running time results of image forgery detection for rotated image using the 3 algorithms. The value is presented in seconds format. So, the lower the time, the faster the algorithm works.

Table 6.4 : Results for copy move forgery testing in terms of time

Results for Copy-Move Forgery Images (Running time)			
Algorithms / Tests (Seconds)	SSIM	SIFT	ORB
1st Test	0.7111790180206299	4.218065500259399	0.8212058544158936
2nd Test	0.6491646766662598	3.978957176208496	0.8832218647003174
3rd Test	0.632143497467041	3.5078842639923096	0.8882086277008057
Average	0.664162397384644	3.9016356468200682	0.8642121156056722

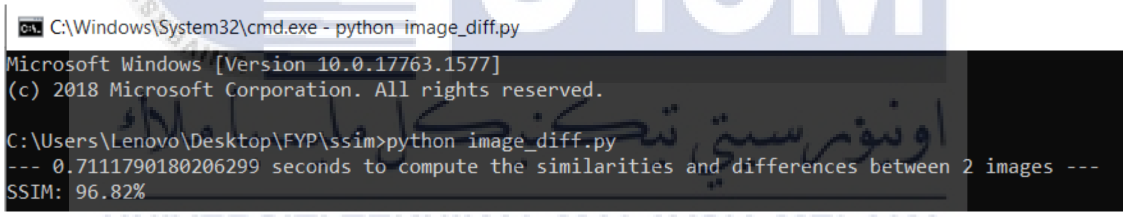
From the Table 6.4 results, it clearly shows that SIFT has the longest average running time compare to SSIM and SIFT which is 3.90 seconds. Second longest average running time is ORB which is 0.86 seconds and followed by SSIM with 0.66 seconds.

6.3.2 Testing results images (System Console)

6.3.2.1 SSIM

Figure 6.17 to Figure 6.20 are the snippets of the system console when the copy move forgery image are tested with the SSIM.

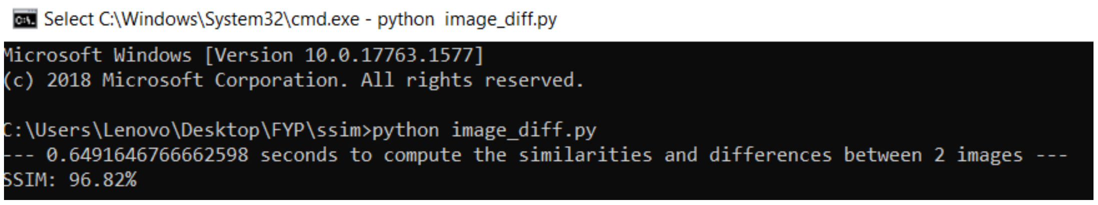
Figure 6.17 is first SSIM testing of copy move forgery image. The score value is 96.82% and the time is 0.7111790180206299 seconds.



```
C:\Windows\System32\cmd.exe - python  image_diff.py
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\ssim>python image_diff.py
--- 0.7111790180206299 seconds to compute the similarities and differences between 2 images ---
SSIM: 96.82%
```

Figure 6.17 : Snippet of first SSIM testing time and score

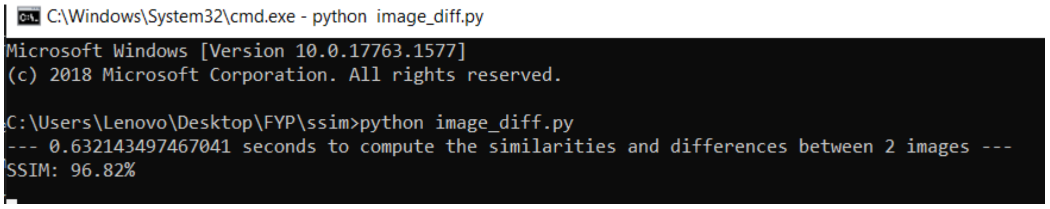
Figure 6.18 is second SSIM testing of copy move forgery image. The score value is 96.82% and the time is 0.6491646766662598 seconds.



```
Select C:\Windows\System32\cmd.exe - python  image_diff.py
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\ssim>python image_diff.py
--- 0.6491646766662598 seconds to compute the similarities and differences between 2 images ---
SSIM: 96.82%
```

Figure 6.18 : Snippet of second SSIM testing time and score

Figure 6.19 is third SSIM testing of copy move forgery image. The score value is 96.82% and the time is 0.632143497467041 seconds.



```
C:\Windows\System32\cmd.exe - python  image_diff.py
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\ssim>python image_diff.py
--- 0.632143497467041 seconds to compute the similarities and differences between 2 images ---
SSIM: 96.82%
```

Figure 6.19 : Snippet of third SSIM testing time and score

Figure 6.20 is SSIM testing for similarities and differences area detection. The red boxes are the detected region of similarities and differences area between two images. The grayscale system console are the part of the algorithm process wehre SSIM will convert the coloured images to grayscale image to identify the similarities and differences. It is more easier to identify the regions in grayscale images.

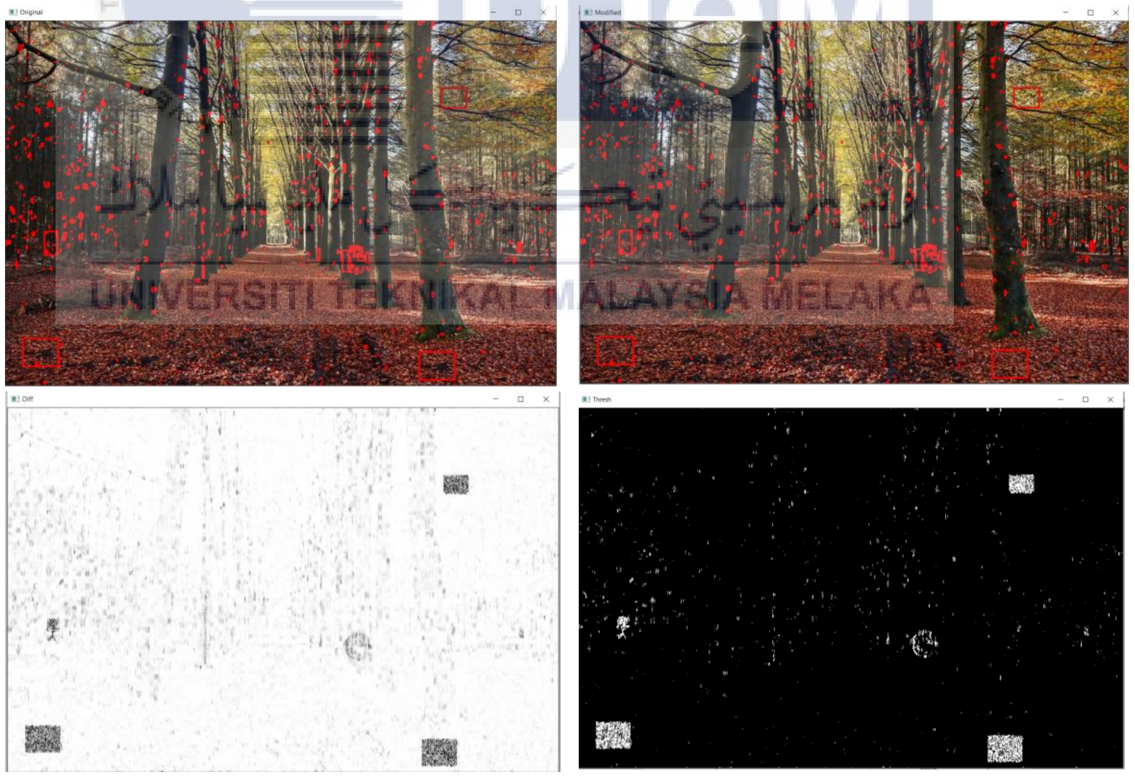
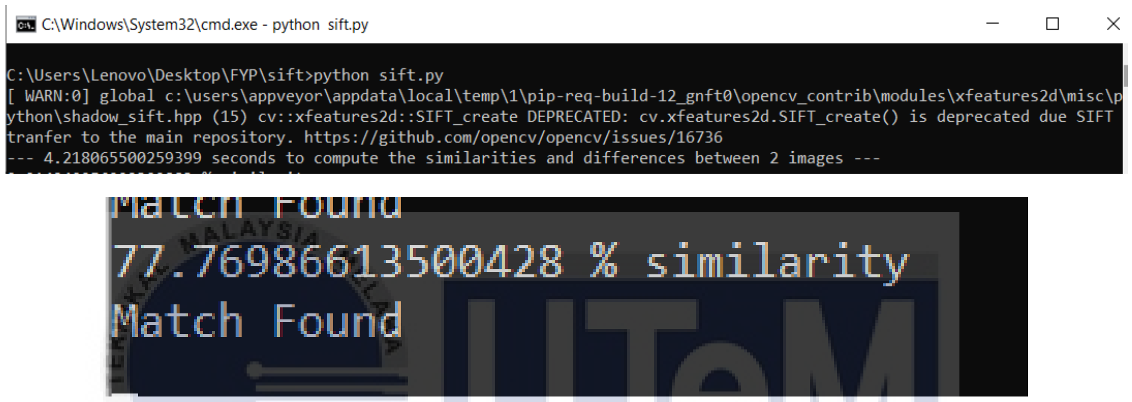


Figure 6.20 : Snippet of Similarities and Differences Region from SSIM Test

6.3.2.2 SIFT

Figure 6.21 to Figure 6.24 are the snippets of the system console when the copy move forgery image are tested with the SIFT.

Figure 6.21 is first SIFT testing of copy move forgery image. The score value is 77.77% and the time is 4.218065500259399 seconds.

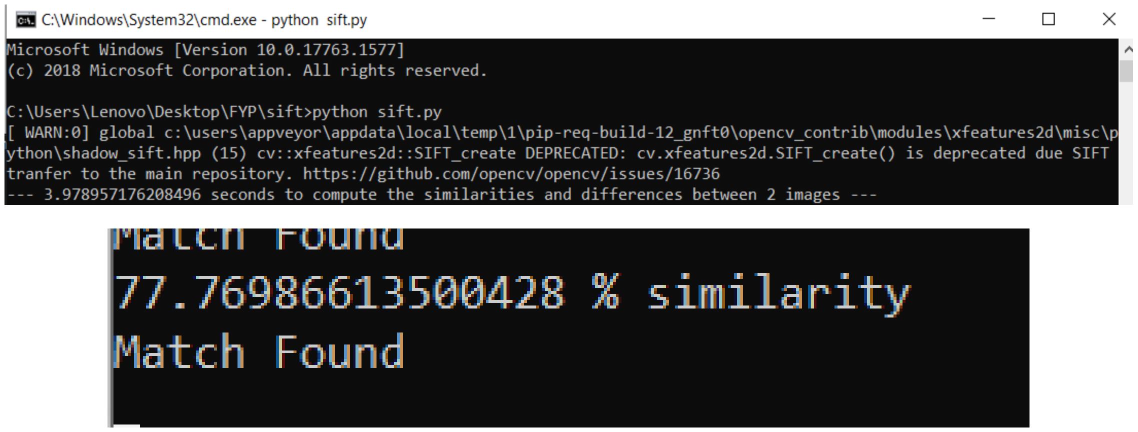


```
C:\Windows\System32\cmd.exe - python  sift.py

C:\Users\Lenovo\Desktop\FYP\sift>python sift.py
[ WARN:0] global c:\users\appveyor\appdata\local\temp\1\pip-req-build-12_gnft0\opencv_contrib\modules\xfeatures2d\misc\p
ython\shadow_sift.hpp (15) cv::xfeatures2d::SIFT_create DEPRECATED: cv.xfeatures2d.SIFT_create() is deprecated due SIFT
tranfer to the main repository. https://github.com/opencv/opencv/issues/16736
--- 4.218065500259399 seconds to compute the similarities and differences between 2 images ---
```

```
77.76986613500428 % similarity
Match Found
```

Figure 6.21 : Snippet of first SIFT testing time and score

Figure 6.22 is second SIFT testing of copy move forgery image. The score value is 77.77% and the time is 3.978957176208496 seconds.

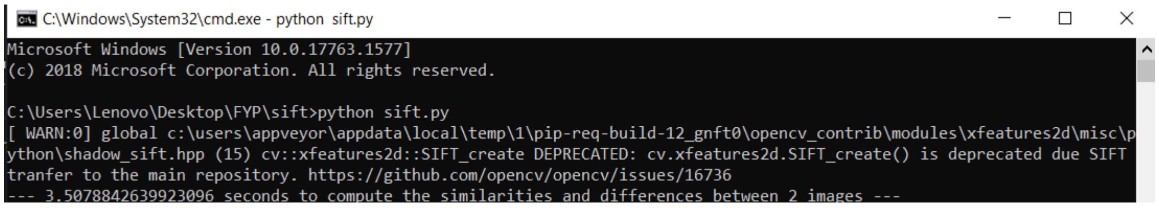


```
C:\Windows\System32\cmd.exe - python  sift.py
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\sift>python sift.py
[ WARN:0] global c:\users\appveyor\appdata\local\temp\1\pip-req-build-12_gnft0\opencv_contrib\modules\xfeatures2d\misc\p
ython\shadow_sift.hpp (15) cv::xfeatures2d::SIFT_create DEPRECATED: cv.xfeatures2d.SIFT_create() is deprecated due SIFT
tranfer to the main repository. https://github.com/opencv/opencv/issues/16736
--- 3.978957176208496 seconds to compute the similarities and differences between 2 images ---
```

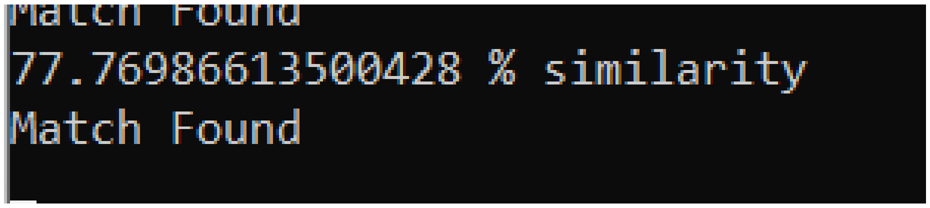
```
77.76986613500428 % similarity
Match Found
```

Figure 6.22 : Snippet of second SIFT testing time and score

Figure 6.23 is third SIFT testing of copy move forgery image. The score value is 77.77% and the time is 3.5078842639923096 seconds.



```
C:\Windows\System32\cmd.exe - python  sift.py
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\sift>python sift.py
[ WARN:0] global c:\users\appveyor\appdata\local\temp\1\pip-req-build-12_gnft0\opencv_contrib\modules\xfeatures2d\misc\p
ython\shadow_sift.hpp (15) cv::xfeatures2d::SIFT_create DEPRECATED: cv.xfeatures2d.SIFT_create() is deprecated due SIFT
tranfer to the main repository. https://github.com/opencv/opencv/issues/16736
--- 3.5078842639923096 seconds to compute the similarities and differences between 2 images ---
```



```
Match Found
77.76986613500428 % similarity
Match Found
```

Figure 6.23 : Snippet of third SIFT testing time and score

Figure 6.24 is SIFT testing for similarities area detection. The lines shown in figure 6.24 are the lines indicating similarities between two images. If the similarities value of key descriptors are more than 75% then the system will draw the lines on the descriptors between two images.

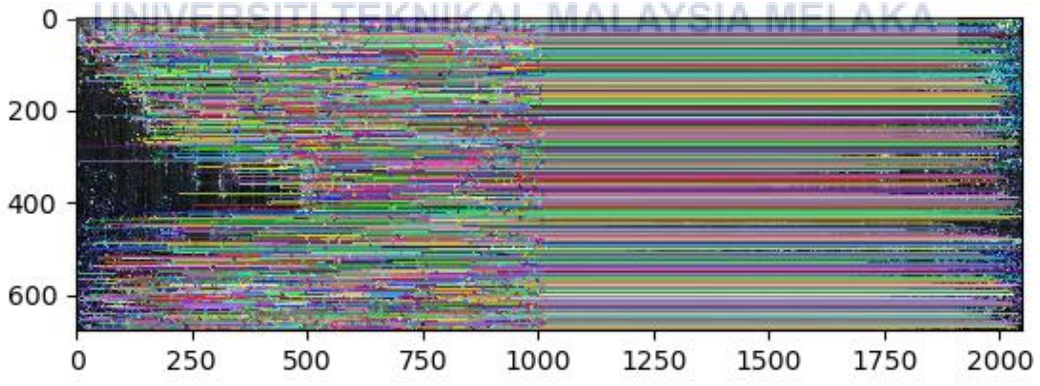
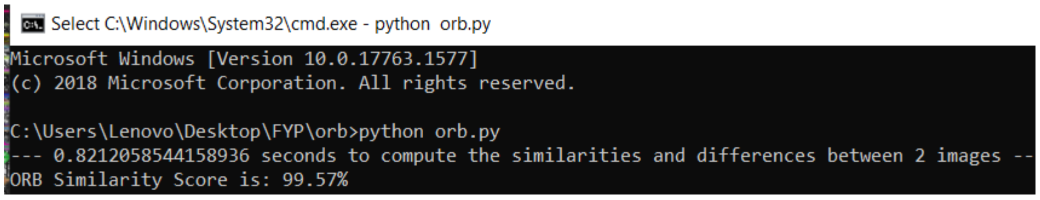


Figure 6.24 : Snippet of Similarities Region from SIFT Test

6.3.2.3 ORB

Figure 6.25 to Figure 6.24 are the snippets of the system console when the copy move forgery image are tested with the ORB.

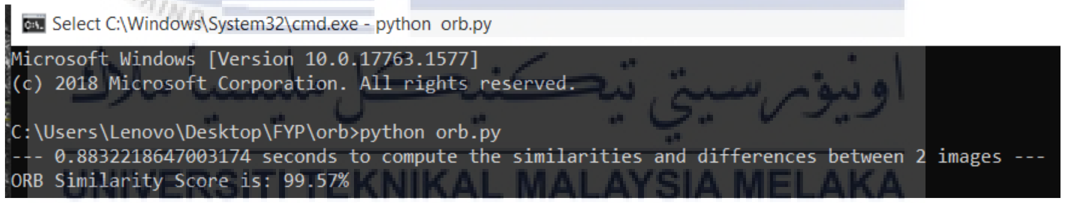
Figure 6.25 is first ORB testing of copy move forgery image. The score value is 99.57% and the time is 0.8212058544158936 seconds.



```
Select C:\Windows\System32\cmd.exe - python  orb.py
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\orb>python orb.py
--- 0.8212058544158936 seconds to compute the similarities and differences between 2 images --
ORB Similarity Score is: 99.57%
```

Figure 6.25 : Snippet of first ORB testing time and score

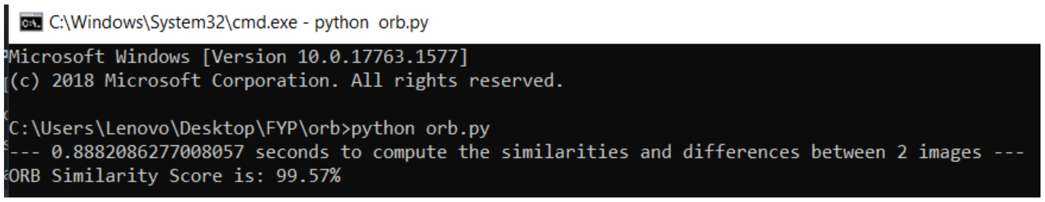
Figure 6.26 is second ORB testing of copy move forgery image. The score value is 99.57% and the time is 0.8832218647003174 seconds.



```
Select C:\Windows\System32\cmd.exe - python  orb.py
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\orb>python orb.py
--- 0.8832218647003174 seconds to compute the similarities and differences between 2 images ---
ORB Similarity Score is: 99.57%
```

Figure 6.26 : Snippet of second ORB testing time and score

Figure 6.27 is third ORB testing of copy move forgery image. The score value is 99.57% and the time is 0.8882086277008057 seconds.



```
C:\Windows\System32\cmd.exe - python  orb.py
Microsoft Windows [Version 10.0.17763.1577]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\FYP\orb>python orb.py
--- 0.8882086277008057 seconds to compute the similarities and differences between 2 images ---
ORB Similarity Score is: 99.57%
```

Figure 6.27 : Snippet of third ORB testing time and score

Figure 6.28 is ORB testing for similarities area detection. The lines shown in figure 6.28 are the lines indicating similarities found between two images. The lines are drawn based on good matching key descriptors found between two images.

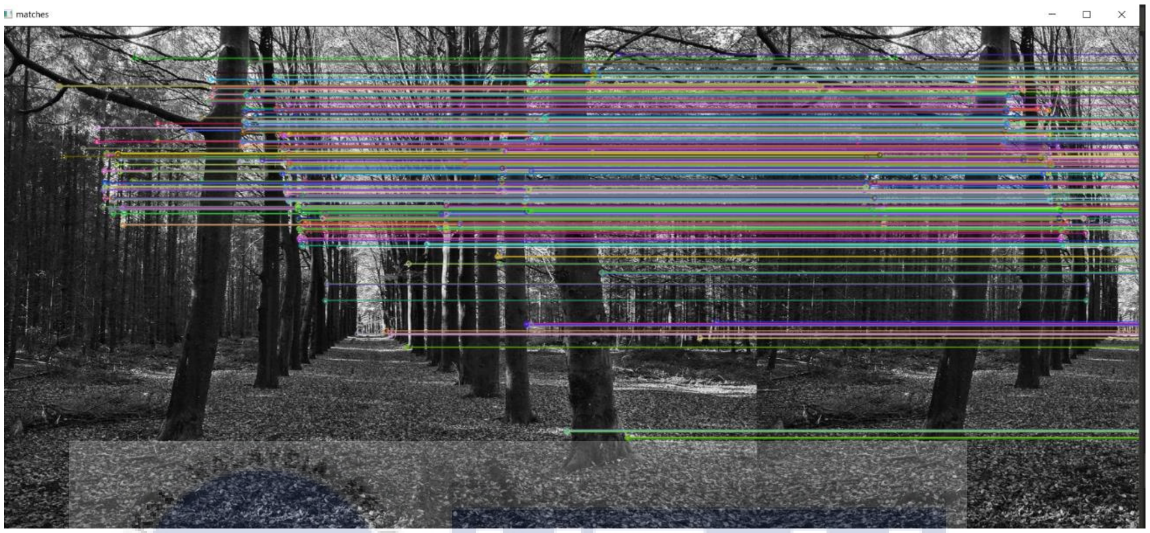


Figure 6.28 : Snippet of Similarities Region from ORB Test

6.4 Conclusion

In conclusion, ORB is the best image forgery detection algorithm for rotated images. Table 6.5 showing that overall result findings for both types of images using 3 algorithms. For rotated image, the highest value of similarities score is 42.74% by ORB and for the lowest time taken to compute the detection is 0.18 seconds by SIFT. Since images under rotated category are same in terms of the content except for their orientation and rotation, So, theoretically a good algorithm can detect the edges and orientation of the images and produce a high similarities score. Although SIFT has the least time taken, but the efficient of an algorithm mainly depends on the detection level. Since the score value for each algorithm is much different, we can clearly see the best detection algorithm for rotated images.

As for copy move forgery images, it is proved that SSIM is the best image forgery detection algorithm. At the same time, for copy move forgery image testing, the results shows that ORB has the highest similarities score value (99.57%) compared to two other algorithms. In terms of running time, SSIM has the least amount of running time which is 0.23 seconds. In this case, the results showing that score value for ORB and SSIM are almost similar. When comes to area detection, the result shows that SSIM has higher accuracy detection on differences area between original image and manipulated image while ORB fails to detect the differences between both images. ORB only able to compute the similarities between two images while SSIM able to identify the differences with exact coordinates between two images. When comes to functionality, SSIM has better job in identifying differences between two images. It is also undeniable that ORB is also a good algorithm for detecting the similarities of copy move forgery images because of the highest similarities score value.

Table 6.5 : Overall Result for the Testing

Types of Images	Algorithms	Similarities Score Value (%)	Time taken (seconds)
Rotated Image	SSIM	27.30	0.23
	SIFT	5.76	0.18
	ORB	42.74	0.72
Copy Move Forgery Image	SSIM	96.82	0.66
	SIFT	77.77	3.90
	ORB	99.57	0.86

CHAPTER 7 : CONCLUSION

7.1 Introduction

This chapter is the conclusion for the entire project. This chapter will summarize all the implementations, testing and analysis that has been done throughout the entire project. In addition, this chapter also includes strength and weakness of the project along with the project contributions. All the mentioned objectives has been achieved on the end of the project and several recommendations and betterment has been suggested for this project for the future better implementations.

7.2 Project Summarization

This research or image forgery system has been designed to identify the differences and similarities between two images to test whether the image has been forged or not. The system will process the two inputted images and calculate the similarities score and running time between both images and the system will find the good matching similar keypoints and descriptors to calculate the similarities scores. The inputted images will process through 3 algorithms which are Structural Similarity Index (SSIM), Oriented FAST and Rotated BRIEF (ORB) and Scale-Invariant Feature Transform (SIFT). In conclusion, this research showing that ORB is the best algorithm for identify rotated images while SSIM is the best algorithm to detect copy move forgery images. The research mainly focuses on analysis and identify the best algorithms among the 3 algorithms which can be used in practical industry world especially in digital forensic department for identifying the image forgery cases. Furthermore, the 3 inputted images will analyze through each algorithm specifically and each matching technique.

To summarize everything, this project has 7 chapters. Starting from chapter 1 discussing about project introduction and background which focuses more on problem statement, research questions, research objectives, research scopes and research contributions. Next is chapter 2 where explanation and table of literature review has been included. It covers the similar previous research and studies about the algorithms that have been done by other researches. This chapter ends with a proposed solution. Next is chapter 3 about project methodology which methodology of this project has been discussed. Prototypes, diagrams and milestones are included in this chapter. Chapter 4 is analysis and design where the logical and physical design of the system architecture and modules and packages needed has been discussed and explained. Next is chapter 5 which is implementation chapter. In this chapter the designed system integrated with algorithms has been implemented and the steps and activities been discussed in this chapter. The programming coding of the system and algorithms also been included and explained in detail in this chapter. The following chapter is chapter 6 which is about testing and analysis of the system and algorithms. In this chapter, the 3 algorithms been tested with two categories of images which are rotated images and copy move forgery images. The final results are been shown in a table to compare between the algorithms. Finally, chapter 7 is the project conclusion where it summarizes the entire project and give the conclusion.

There are 3 objectives in this project. First is to explore and apply Structural Similarity Index (SSIM), Scale-Invariant Feature Transform (SIFT) and Oriented FAST and Rotated BRIEF (ORB). This objective has been achieved in Chapter 2 and the following chapters. In chapter 2, the way of the algorithms works is been explained and the algorithms are been implemented in the systems in following chapters. Second objective is to identify and detect the differences between original and manipulated images. This objective has been achieved in chapter 5 and chapter 6 where the system is developed using the algorithms is able to identify the coordinates of the similarities and differences between two images. Last objective in this research is to recommend and propose the best and efficient algorithms between the 3 algorithms. This objective has been achieved in chapter 6 and chapter 7. Once the testing are carried out, we can conclude the best and efficient algorithms among them with the result analysis. So, based on the analysis from the result table, it is concluded that ORB is the best algorithm to detect image forgery for rotated images category while SSIM is the best algorithm to detect the image forgery for copy move forgery images category.

7.3 Project Contribution

This project contributes mainly to Cyber Security departments and Digital Forensics departments. This system mainly developed to help Digital Forensics department in cybercrime cases especially on image forgery cases. Besides that, this system also can be implemented in various types of field offices such as banks. Banks can use the system to identify any changes in signatures. It is because these systems can identify exactly the exact coordinates of differences and similarities of two images. So, this feature can help to identify any forgeries easily. So, the contribution at the end of the project is this project proved and proposed the best algorithms for each type of image.

7.4 Project Limitations

There are several limitations in this projects that need to be highlighted for future improvisation. First limitation that occurs during the systems developments is comes from the algorithms itself. For an example, Structural Similarity Index (SSIM) and Scale-Invariant Feature Transform (SIFT) algorithms cannot detect rotated images with high percentage accuracy. Both algorithms are not able to detect the rotated images. So, both algorithms are not applicable for rotated images. At the same time, some algorithms cannot detect scaled images as well. In addition, some algorithms for example Scale-Invariant Feature Transform (SIFT) need to purchase since it is patented algorithm but for academic and research purpose the algorithm is free to use.

7.5 Future Works

The future of technology usage is Internet of Things (IoT). So, for future improvements these systems can be implemented in any IoT devices or even in smartphones applications for more user-friendly and easier to use. In addition, these systems can be embedded with a database system where the system will have images in the database as original copy and the user can input the forgery images into the system and the system will automatically find the original images based on the similarities and differences scores values. Moreover, the system can be improvised in terms of usage of searching and matching techniques. The system can accurately match and draw some boxes or circles around the identified forged parts in the image in exact coordinates. Furthermore, combining ORB and SSIM algorithms together produce a new algorithm will be great in identifying image forgery.

7.6 Conclusion

As for conclusion, this final year project is successfully achieved its objectives. The systems are successfully developed and the algorithms are successfully implemented. The systems are able to identify the coordinates of different and similar parts between two inputted images and finally produce a score value. Although the systems have some drawbacks, but it is good to have such image forgery detection systems in digital forensic department to combat with cybercrimes such as image forgery cases. Moreover, all the information gathered and analysis performed in this project can be used as future improvements for a better systems and algorithms.

References

Ansari, A. (2020, October 13). Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/getting-started-scikit-image-image-processing-python/

Atefeh Shahroudnejad, M. R. (2016). Copy - Move Forgery Detection in Digital Images Using Affine SIFT. *ICSPIS 2016, 14 - 15 Dec. 2016, Amirkabir University of Technology Tehran, Iran*.

Chuan Luo, W. Y. (2019). Overview of Image Matching Based on ORB Algorithm. *Journal of Physics: Conference Series*.

Datta, P. (2020, September 3). *All about Structural Similarity Index (SSIM): Theory + Code in PyTorch*. Retrieved from Medium: https://medium.com/srm-mic/all-about-structural-similarity-index-ssim-theory-code-in-pytorch-6551b455541e

Fahime Hakimi, M. H. (2015). Image Splicing Forgery Detection Using Local Binary Pattern abd Discrete Wavelet Transform. *International Conference on Knowledge Based Engineering and Innovation*.

J.Lukas, J. a. (2003). Estimation of primary quantization matrix in double compressed JPEG images. *Digital Forensic Research Workshop* .

J.Wang, G. Z. (2009). Fast and robust forensics for image region-duplication forgery. *Acta Autimatica Sinica*, 95-1488.

Lv, G. (2015). Robust and Effective Techniques for Multi-modal Image Registration.

Matthias Carnein, P. R. (2015). Forensics of High Quality JPEG Images with Color Subsampling. *IEEE International Workshop on Information Forensic and Seucirty (WIFS)*.

Ms Jayshri Charpe. (2015). Revealing Image Forgery through Image Manipulation Detection. *Global Conference on Communication Technologies*.

P. Sutthiwan, Y. S. (2010). Rake transform and edge statistic for image forgery detection. *Internation conference on multimedia*, 8 - 1463 .

P.Xunyu, L. (2011). Region duplication detection using image feature matching. *Transactions on Information Forensics security*, 67 - 857.

python. (2021, January 15). Retrieved from https://pypi.org/project/imutils/

Reshma P.D, A. C. (2015). Image Forgery Detection Using SVM Classifier. *International Conference on Innovations in Information Embedded and Communication Systems*.

Rosebrock, A. (2017, June 19). *Image Diiference with OpenCV and Python*. Retrieved from pyimagesearch: https://www.pyimagesearch.com/2017/06/19/image-difference-with-opencv-and-python/

Snigdha K. Mankar, P. D. (2015). Image Forgery Types and Their Detection : A Review. *Internation Journal of Advanced Research in Computer Science and Software Engineering*.

Varsha Karbhari Sanap, V. M. (2015). Region Duplication Forgery Detection in Digital Images Using 2D - DWT and SVD. *2015 IEEE*.

W.Luo, F. J. (2007). A survey of passive technology for digital image forensic. *Front computer science China*, 79 -166.

Z.Lint, R. X. (2005). Detecting Doctored Images using Camera Response Normality and Consistency. *IEEE Computer Society Conference on computer vision and Pattern Recognition*, 92 - 1087.

Zhou Wang, A. C. (2004). Image Quality Assessment : From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*.