DETERMINATION OF ROLLING RETURNS FROM A RASTERIZED GRAPH USING IMAGE PROCESSING

CHEW JIA HAO

UNIVERSITI TUNKU ABDUL RAHMAN

DETERMINATION OF ROLLING RETURNS FROM A RASTERIZED GRAPH USING IMAGE PROCESSING

CHEW JIA HAO

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Electrical and Electronic Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

May 2020

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	[illegible]
Name	:	CHEW JIA HAO
ID No.	:	1502778
Date	:	16/5/2020

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"DETERMINATION OF ROLLING RETURNS FROM A RASTERIZED GRAPH USING IMAGE PROCESSING"** was prepared by **CHEW JIA HAO** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Electrical and Electronic Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	MR NG CHOON BOON
Date	:	16/5/2020

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Mr. Ng Choon Boon for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and given me encouragement.

ABSTRACT

This project is to develop a software application that is able to extract raw data from a rasterized performance graph of a mutual fund using image processing. The extracted raw data are the percentage return of mutual fund with the date. The software is able to calculate rolling return of mutual fund using the extracted raw data. This software application will make investor's work becomes easier and also provides them better insight into the mutual fund performance. The main programming language used in this project is Python. OpenCV and Tkinter are used in this program as image processing library and graphical user interface library respectively. Users will get raw data in an excel file and graph of rolling return of mutual fund after users insert the graph image and some inputs into the software application. Validation of results was carried out after every simulation during developing the program. The average percentage difference obtained by comparing actual value with calculated value is 1.84% for one-year, 2.09% for three-year return and 2.22% for five-year return. There are more works and efforts needed on improving the accuracy of results while reducing the number of inputs provided by users so that the software application will become more efficient and effective.

TABLE OF CONTENTS

DECLARATION				**1**
APPROVAL FOR SUBMISSION				**2**
ACKNOWLEDGEMENTS				**4**
ABSTRACT				**5**
TABLE OF CONTENTS				**6**
LIST OF TABLES				**9**
LIST OF FIGURES				**10**
LIST OF SYMBOLS / ABBREVIATIONS				**12**
LIST OF APPENDIX				**13**
CHAPTER				
1	**INTRODUCTION**			**14**
	1.1	General Introduction		14
	1.2	Importance of the Study		14
	1.3	Problem Statement		15
	1.4	Aims and Objectives		15
	1.5	Scope and Limitation of the Study		16
	1.6	Contribution of the Study		16
	1.7	Outline of the Report		16
2	**LITERATURE REVIEW**			**18**
	2.1	Digital Image Formation		18
		2.1.1	Pixel of image	19
		2.1.2	Colour Image	19
	2.2	Digital Image Processing		20
		2.2.1	Image Filtering	21

2.2.2 Conversion of Colour Image to Grayscale Image 23

2.2.3 Thresholding 26

2.3 Relationship between Investment and Image Processing 28

2.3.1 Absolute Return 28

2.3.2 Compound annual growth rate (CAGR) 29

2.3.3 Rolling Return 30

2.3.4 Summary 30

2.4 Review on others similar software 31

2.4.1 WebPlotDigitizer 31

2.4.2 Graphreader 32

2.4.3 Digitizelt 33

2.4.4 Comparison of software and Conclusion 33

3 METHODOLOGY AND WORK PLAN 35

3.1 Introduction 35

3.2 Programming tools 36

3.2.1 OpenCV 36

3.2.2 Tkinter 37

3.3 Image pre-processing 38

3.4 Determine the pixel location of the line 39

4 RESULTS AND DISCUSSIONS 42

4.1 Introduction 42

4.2 User Interface 42

4.3 Results 44

4.3.1 Example - Dana Makmur Pheim 44

4.3.2 Example - AmBond Fund 47

4.3.3 Example - Apex Dynamic Fund 50

4.3.4 Example - CIMB Principal Strategy Bond Fund 53

4.4 Discussion 56

5 CONCLUSIONS AND RECOMMENDATIONS **60**

5.1 Conclusion 60

5.2 Recommendation 60

REFERENCES **62**

APPENDIX **64**

LIST OF TABLES

Table 2-1: Colour and its pixel value 20

Table 2-2: Colour and their wavelength 25

Table 2-3: Comparison of software 34

Table 4-1: Comparison between actual value and calculated value (Dana Makmur Pheim) 46

Table 4-2: Comparison between actual value and calculated value (AmBond Fund) 50

Table 4-3: Comparison between actual value and calculated value (Apex Dynamic Fund) 53

Table 4-4: Comparison between actual value and calculated value (CIMB Principal Strategy Bond Fund) 56

LIST OF FIGURES

Figure 2.1: Image projected onto a CCD array 18

Figure 2.2: Results of image after sampling 18

Figure 2.3: Grayscale image with different shades 19

Figure 2.4: Image after filtering 21

Figure 2.5: Grayscale Image after Conversion 24

Figure 2.6: Result image after Thresholding 27

Figure 2.7: User Interface for the WebPlotDigitizer 31

Figure 2.8: User interface of Graphreader 32

Figure 2.9: Screenshot from Digitizelt software 33

Figure 3.1: Flowchart of the extraction data 35

Figure 3.2: Example of GUI using Tkinter 37

Figure 3.3: HSV in 2-Dimension 39

Figure 4.1: User Interface 43

Figure 4.2: Error window 43

Figure 4.3: Output window 43

Figure 4.4: Original image of Dana Mukmur Fund 45

Figure 4.5: Segmentation of line (Dana Makmur Fund) 45

Figure 4.6: 1-Year Rolling Return of Dana Makmur Fund 46

Figure 4.7: 3-Year Rolling Return of Dana Makmur Fund 47

Figure 4.8: 5-Year Rolling Return of Dana Makmur Fund 47

Figure 4.9: Original Image of AmBond Fund 48

Figure 4.10: Segmentation of line (AmBond Fund) 48

Figure 4.11 : 1-Year Rolling Return of AmBond Fund 49

Figure 4.12 : 3-Year Rolling Return of AmBond Fund 49

Figure 4.13 : 5-Year Rolling Return of AmBond Fund 50

Figure 4.14: Original Image of Apex Dynamic Fund 51

Figure 4.15: Segmentation of line (Apex Dynamic Fund) 51

Figure 4.16: 1-Year Rolling Return of Apex Dynamic Fund 52

Figure 4.17: 3-Year Rolling Return of Apex Dynamic Fund 52

Figure 4.18: 5-Year Rolling Return of Apex Dynamic Fund 53

Figure 4.19: Original Image CIMB Principal Strategy Bond Fund 54

Figure 4.20: Segmentation of line (CIMB Principal Strategy Bond Fund) 54

Figure 4.21: 1-Year Rolling Return of CIMB Principal Strategy Bond Fund 55

Figure 4.22: 3-Year Rolling Return of CIMB Principal Strategy Bond Fund 55

Figure 4.23: 5-Year Rolling Return of CIMB Principal Strategy Bond Fund 56

Figure 4.24: Two Y-axis Pixels with same X-axis Pixel Location 57

Figure 4.25: Tail part of line 58

LIST OF SYMBOLS / ABBREVIATIONS

σ	Standard deviation of the distribution
CCD	Charge-coupled device
RBG	Red, Blue, Green
HSV	Hue, Saturation, Value
OCR	Optical Character Recognition
CAGR	Compound Annual Growth Rate

LIST OF APPENDIX

Appendix A: Tables 64

Appendix B: Source Code 69

CHAPTER 1

INTRODUCTION

1.1 General Introduction

In the modern era, data have become one of the most important resources. Important information can be analysed from the data by using proper analytics tools and techniques (Basavaprasad, B. and Ravi, M., 2014). Data collection and analytics are commonly used in the business field to obtain valuable insight for better business decisions. With proper analytics method, information extracted from data can be used to predict the future market and best area for new investment. Therefore, data collection and analytics techniques become important for getting a better investment and profit.

In this project, a data analytics software was developed for determination of rolling return of mutual funds from a rasterized line using image processing. Users can insert an image of rasterized graph of mutual fund performance to the software. Data such as the price will be extracted by using image processing techniques. This software will focus on determination of rolling return, so the data extracted will go through a specific algorithm to calculate the desired output.

1.2 Importance of the Study

Image processing has become very common in this century. It has been used in many fields such as medical, financial, and industry. Huge amounts of digital images are produced from these fields. For example, the medical field has x-rays of patients and the performance graph from mutual fund. Human requires things to be faster and more accurate. Studying image processing can create more effective, efficient and innovative applications.

In this project, image processing is used to extract the data from performance graphs of mutual fund. The data extracted must be very accurate so that a better insight of investment can be provided to the investor. Therefore, researching and understanding the image processing techniques become very important in order to develop a good program.

1.3 Problem Statement

In financial services such as stocks market and mutual funds, bulks of data are generated daily. It is impossible for Investors to read and observe for each stock and mutual fund. Investors might need to spend a lot of time on investigating the market price of funds but still have high chances to lose money.

Mutual Fund is also known as Unit Trust in Malaysia. Fact sheet of Unit Trust is a document that provides an overview of a mutual fund retails. Investors usually will read the document before investing in the fund. Fact sheet will provide information such as price and returns over the last 10 years. However, rolling return is not provided in the fact sheet. Some investors might need this information to have a better insight. Calculating rolling return required data of each month over a few years. Investor need to spend time or pay on collecting all the data from the historical price of fund. The graph of the historical price of funds can be captured as an image and the data can be extracted easily by using image processing. Once the data is extracted, it can be used to perform mathematical operation to get the rolling return.

The accuracy of the data extracted is also the difficulty of this project. Users might provide a different quality of image. The lower the quality of images will result in lower accuracy of data extracted. Users are required to provide at least two values within the graph in order to estimate the rest of the value in the graph. However, users would prefer to make the process fully automated when using the software in order to save their time. It is hard to estimate the value from graph without any input information from users. Text recognition technique might need to be performed in order to make the process easier.

The study of image processing techniques become the core to overcome the problem statement stated above.

1.4 Aims and Objectives

The project is developing software that is able to calculate the rolling rate of return or gain from a rasterized line using image processing with minimal human input. The aims and objectives of the project are:

1. Extract data from a rasterized graph using image processing with minimal user input
2. Able to determine the rolling return from extracted data.
3. Develop a graphical user interface that is user-friendly for the software.

1.5 Scope and Limitation of the Study

The project focuses on extraction data from the graph based on the image provided by the user. In this project, the images are obtained from an online unit trust distribution platform. Images with different resolution and pattern are used as data for testing. Only a few techniques of image processing are studied for this application.

Humans always seek for fully automated technology to reduce their burden. However, it is hard to implement on this project due to limitations on computer vision and time. Additional study on text recognition needs to be done in the future so that the objective of minimizing the user input can be achieved.

1.6 Contribution of the Study

There is some similar software for data extraction from images. Some of this existing software required users to insert different kinds of input in order to estimate the output value. This project intends to develop a software that can extract the data with minimal the user input. The data are used to perform specific tasks such as determine the rolling return of a mutual funds. The reason for using image processing is because there are too many mutual funds exists in the market. The software can improve the efficient compare to hand calculation.

1.7 Outline of the Report

Chapter 1 is the introduction of the project. It will explain the concept, problem statement, aim, and objectives of the project.

Chapter 2 is the literature review. It will explain theory about image processing and rolling return. Some similar software is review and compare.

Chapter 3 is the methodology. It will explain the software and image processing techniques used in the project.

Chapter 4 is the result and discussion. It will show and explain the result. It also will discuss the problems encountered in this project.

Chapter 5 is the conclusion and recommendation. It will explain how the aims and objectives will be achieved. It will also recommend for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Digital Image Formation

A digital image can be formed by a digital camera or created by a computer. In digital cameras, any object can be captured as an image if there is a light source such as sunlight or lighting hit on the object. There is an image sensor called charge-coupled device (CCD) inside a digital camera. CCD has a two-dimensional array of cells to sense the intensity of light reflected by the object and convert it into different voltage levels (Felber, 2002). Figure 1 demonstrates image projected onto a CCD array. The value of voltages level for each cell will be sampled and converted into a digital signal. A complete digital image can be formed throughout this process. Figure 2 shows the digital image after sampling.

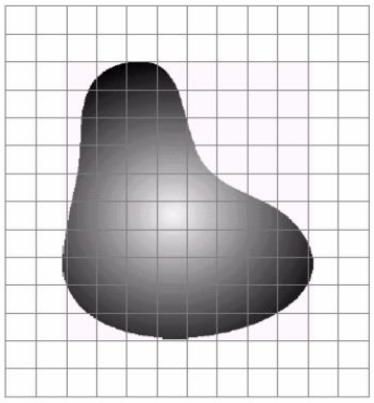


Figure 2.1: Image projected onto a CCD array

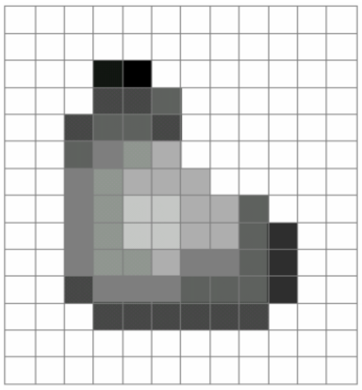


Figure 2.2: Results of image after sampling

2.1.1 Pixel of image

Pixel, also known as picture element, is the smallest element of an image. Pixel can be represented as the two-dimensional array of cells of CCD. Each pixel contains the intensity level of the light hit on the CCD array. The value of each pixel can be represented using binary number with different number of bits. The larger the number of bits per pixel will result in a larger number of different colour. For example, the black and white image has 1 bit per pixel where the pixel value of 1 represents white colour and the pixel value of 0 represents black colour. Greyscale image has the value from 0 to 255 for each pixel. Figure 3 shows the grayscale image with different shades. Value of 0 represents black colour, value of 255 represents white colour, while the value between 1 and 254 represent different shades of grey. Therefore, pixel value can be expressed as a function, $f(x, y)$, where x and y represents the pixel position.

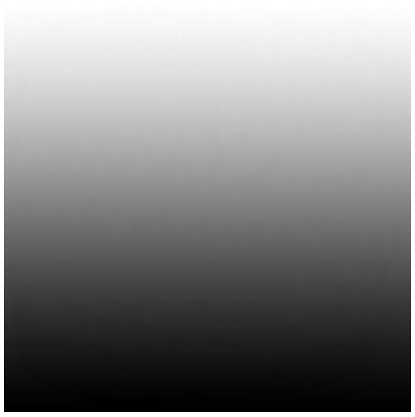


Figure 2.3: Grayscale image with different shades

2.1.2 Colour Image

For a colour image, the pixel contains a vector of three numbers. Each numbers represent the value for red, green and blue colour (RGB). The common pixel format is 24 bits colour format. 24 bits is equally divided on RGB. So, each RGB will be represented by 8 bits. Now the image function, f can be expressed as:

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

For the pixel matrix has the value of $(255 \quad 0 \quad 0)$, the pixel will be represented as red colour. Pixel matrix with the value of $(0 \quad 255 \quad 0)$, the pixel will be represented as green colour and blue colour for the pixel matrix with the value of $(0 \quad 0 \quad 255)$. Some other colours can be shown by changing the pixel value in the matrix. Table 1 shows the colour and its pixel value.

Table 2-1: Colour and its pixel value

Colour	Pixel Value
Cyan	$(0 \quad 255 \quad 255)$
Magenta	$(255 \quad 0 \quad 255)$
Yellow	$(255 \quad 255 \quad 0)$
Black	$(0 \quad 0 \quad 0)$
White	$(255 \quad 255 \quad 255)$

2.2 Digital Image Processing

Digital Image Processing is the programming algorithms used to analyse image in order to extract the useful information or data in the image. These data can be further analysed to estimate the trends or patterns of an event. Applications of image processing are getting wider as the techniques of using image processing are getting more mature. For instance, medical diagnosis, robot visions, intelligent transporting systems, and face recognition.

A digital image is a two-dimensional discrete signal converted from analog signal through sampling and quantization. A two-dimensional signal can be represented as the mathematical function $f(x,y)$ where x and y represented as horizontal and vertical coordinates respectively. The magnitude of the function f is the pixel value of an image, however, the pair of x and y coordinates is the pixel of an image. Therefore, image processing can be done by using different mathematical methods. For example, image filter can be done by using convolution, turn image into grayscale can be done by using the weighted method, and edge detection in an image can be done by using Laplacian operator. Since digital image processing involve in complex mathematic calculations and algorithms, the best way to perform digital image processing is using programming.

2.2.1 Image Filtering

Since images are considered as signals, it might consist of different type of noises. Image Filtering is one of the most important techniques in image processing to smooth the image by reducing the noise and improve the quality of the image. Kernel is used with a filter that performs average smoothing. Figure 2.4 shows the image after filtering. Another purpose of using an image filter is to detect the edge or sharpness of images. There are different types of image filtering algorithms with different kernels available. Each algorithm can be studied in order to understand their effect so that the proper technique is used in specific applications to ensure an efficient process.

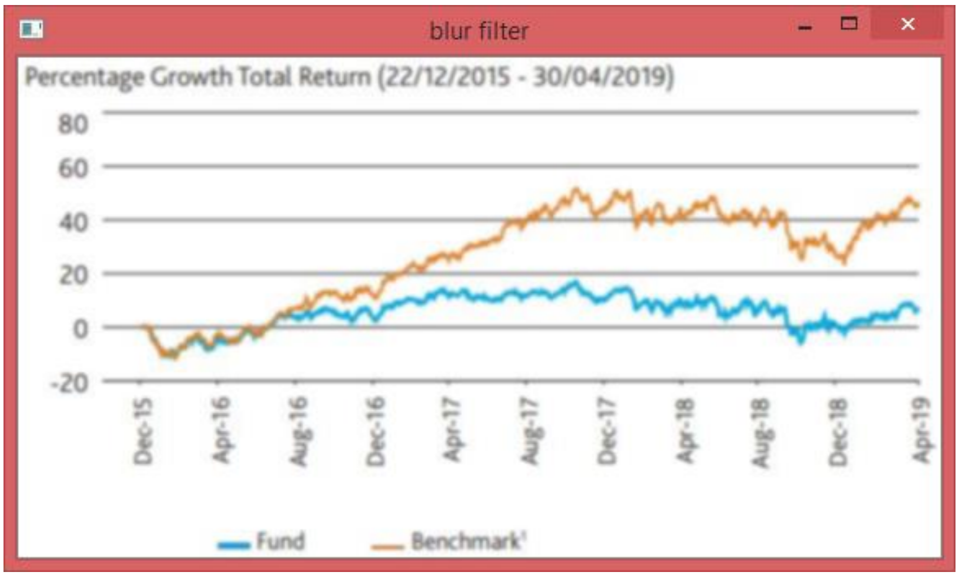


Figure 2.4: Image after filtering

2.2.1.1 Moving Average Filters

Moving average filters is the most common and simple filter in digital signal processing. It used with a square kernel that contains equal coefficient:

$$Kernel = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

(2-1)

The equation for the moving average filter can be expressed as:

$$y[i] = \frac{1}{M}\sum_{j=0}^{M-1} x[i+j] \qquad (2\text{-}2)$$

The concept of implementation of moving average filter is shown below:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 90 & 90 & 90 \end{bmatrix} \rightarrow \begin{bmatrix} & & \\ & 30 & \\ & & \end{bmatrix}$$

It calculates and replaces each pixel with average pixel value of the surrounding pixel. In this example, the pixel value surrounds by the middle pixel is sum up and divide with the total number of pixels to get the average value. The middle pixel now replaced with the average value. This process repeated for every pixel and results in a smooth and blurred image.

2.2.1.2 Gaussian Filter

Gaussian Filter is a low pass filter that used to reduce the noise of image and blur the image by reducing the high-frequency component. The Gaussian function for one dimension can be expressed by the equation below:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \qquad (2\text{-}3)$$

where

$G(x)$ is the Gaussian Function

σ is the standard deviation of the distribution

Gaussian function with two-dimension is the product of two one-dimensional Gaussian functions which decomposed into a series of one-dimensional filtering for rows and columns. It can be expressed by the equation below:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2-4}$$

Convolution of a kernel is expressed by a Gaussian function in Gaussian Filter (Tyagi, T. and Mishra, V., 2016). The two-dimensional Gaussian function is used to generate the kernel. Convolution kernel is used to approximate the Gaussian distribution. The kernel is rotationally symmetric with no directional bias. Example of the Gaussian kernel is shown below:

$$Gaussian\ kernel = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{2-5}$$

Convolution matrix can be obtained by using the value from Gaussian distribution. The weighted average of the pixel neighbourhood is replaced for each pixel as a new value respectively. Thus, the original pixel’s value is replaced with the larger Gaussian value and neighbourhood pixels is replaced with smaller Gaussian value as their distance refer to the original pixel become further (Gedraite and Hadad, 2011). The standard deviation of the Gaussian function is used to define the filter since the kernel coefficients depend on it.

2.2.2 Conversion of Colour Image to Grayscale Image

Conversion of colour image to grayscale image usually used by different applications. For example, printing a grayscale image from a colour image. As discussed above, colour image is a function of a vector with 3 numbers however, the grayscale image is a function consist of only one-pixel value. Figure 2.5 shows the grayscale image after converted from colour image. Conversion of colour image to grayscale image helps to simplify and make the computation easier. Tasks like edge detection will be easier to perform in grayscale image. However, some information will be lost during the conversion (Saravanan, 2010). The quality of the grayscale image will be affected. Therefore, a good algorithm needs to be used for the conversion to maintain the good quality of grayscale image in order to perceive the information as in colour image.

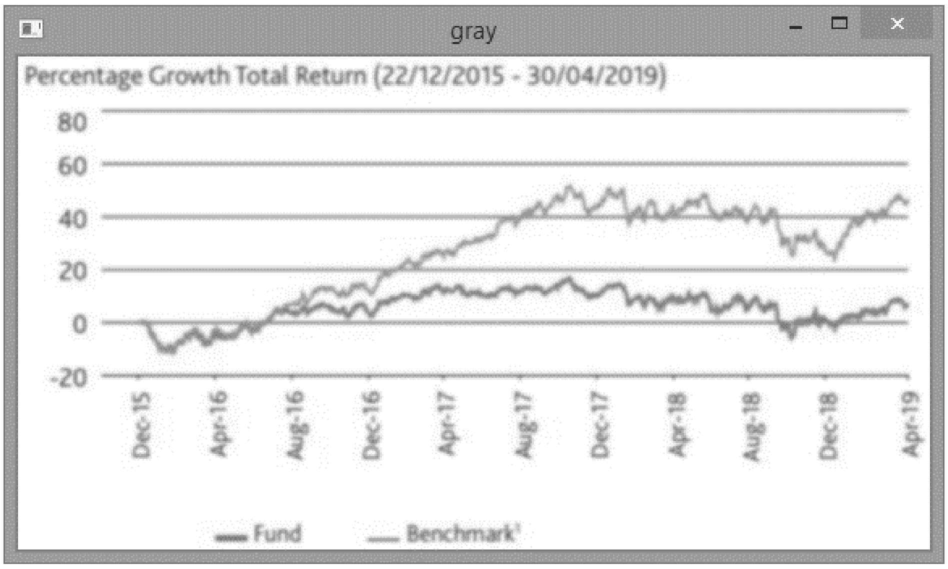


Figure 2.5: Grayscale Image after Conversion

2.2.2.1 Average Method

The average method is the basic algorithm used to convert colour image to grayscale image. The average value of three colour in a pixel is calculated and become the latest value of the pixel. The algorithm is demonstrated and shown below:

$$Pixel\ value\ of\ a\ colour\ image, f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix} = \begin{bmatrix} 255 \\ 255 \\ 0 \end{bmatrix} \quad \text{(2-6)}$$

Taking the average value of three colour,

$$Average = \frac{255+255+0}{3} = 170 \qquad (2\text{-}7)$$

Therefore, the new value of the pixel become:

$$f(x,y) = [170]$$

Where value of 170 can be referred to one of the intensities of grayscale.

This method is the easiest method but not an effective method. Since the average is taken from the three colours, each colour will contribute around 33%. However, different colour has different wavelength contribution in formation of image. The result of grayscale image will become very dim and the information in colour image cannot be perceived.

2.2.2.2 Luminosity Method

Since the average method is not an effective method for converting colour image to grayscale image, another method called Luminosity Method is used. The luminosity method has a better consideration in the contribution of wavelength of each colour. It also takes the average value of three colours but a weighted average is formed to account for human perception.

Table 2-2: Colour and their wavelength

Colour	Wavelength
Red	700
Green	530
Blue	470

Based on Table2-2, red colour has a larger wavelength while blue colour has a shorter wavelength among the three colours. Therefore, the weight of the colour needs to be adjusted. Humans are more sensitive to green colour. Green colour gives a soothing effect to human eyes compare to other colours (Cook, 2009). So, the new equation for the conversion is expressed as below:

$$f(x,y) = (0.3 * r(x,y) + 0.59 * g(x,y) + 0.11 * b(x,y)) \qquad (2\text{-}8)$$

The grayscale image using the luminosity method will result in a brighter image and higher quality. Further application on the grayscale image can be carried on with better efficiency.

2.2.3 Thresholding

Thresholding is a method used to produce a black and white image from grayscale image. The meaning of Thresholding is to assign the pixel value to either one or zero if the pixel value is greater than a threshold value (Sezgin, M. and Sankur, B, 2004). This approach can be expressed as the equation below:

$$f(x,y) = \begin{cases} 0, & x < threshold \\ 255, & x \geq threshold \end{cases} \qquad (2\text{-}9)$$

For grayscale image, the grayscale level of the object is substantially different from the background. Thresholding becomes important in this situation to extract the object from the background. Figure 2.6 shows the results image after Thresholding. Some common applications of Thresholding are document image analysis, map processing, and x-ray computed tomography. The purpose of using the Thresholding is to extract the important object from the image and separate it with the background. Therefore, the threshold value needs to be selected properly in order to get the desired object from the image. There are six groups of Thresholding method are categorized based on the information they are exploiting. Four groups among the six groups will be discussed in the following paragraph.

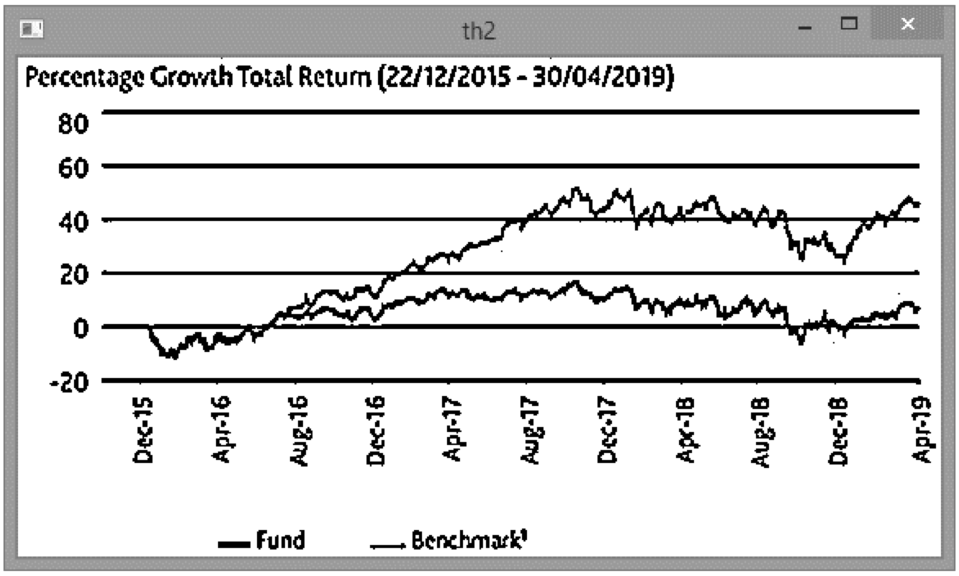


Figure 2.6: Result image after Thresholding

The first category of the Thresholding method is Histogram Shape-Based Thresholding. The value of threshold is choose based on the peaks, valleys, and curvatures of the smoothed histogram. Image histogram needs to be analysed and studied to get a proper threshold value.

The second category of the Thresholding method is clustering-based methods. Clustering-based methods cluster the pixel into object and background based on the grayscale level. Each cluster corresponds to a lobe of the image histogram. There are several clustering methods can be further sub-classified. For example, iterative-based method, Otsu’s method, minimum error Thresholding, and fuzzy clustering Thresholding.

The third category of the Thresholding method is the Entropy-based method. The Entropy-based method is used to measures spontaneous distributing energy based on the second law of Thermodynamics. It was used in the communications field for measuring the effectiveness in data transmission through a channel with noises. Great information transfer indicated by high entropy. Minimizing cross-entropy between input grayscale image and binary output image can achieve the preservation of information.

The forth category of the Thresholding method is Local Adaptive Thresholding. Depend on attribute quality, local parameters, or similarity measure, the threshold of each pixel is selected. This method works well on unequal illumination because each pixel has its own threshold (Chandel, R.

and Gupta, G., 2013). One of the common sub-categories of local adaptive Thresholding is the Local Variance method. The threshold is selected based on the mean, standard deviation, and local window size. This method widely used in Optical Character recognition tasks because it performs effectively in these tasks.

2.3 Relationship between Investment and Image Processing

Image processing technology becomes very important nowadays. Multiple transformations on the image can be done by using image processing techniques. The purpose of performing image processing is to mine the data and the data can be analysed to predict the outcome. Image processing steps are shown as:

1. Image pre-processing
2. Extraction of pattern or features
3. Image clustering and classification
4. Analysis and data mining

Mutual Fund Company usually just provide a graph of the percentage growth. Therefore, image processing becomes important to extract the data from the graph for further analysis in order to find out the performance of it and any future insight for investment. There are several methods to find out the performance of stocks or mutual funds such as absolute return, rolling return, and compound annual growth rate. These methods will be discussed later to determine which method has a better insight for investors.

2.3.1 Absolute Return

Absolute return can be explained as the return calculated based on specific period of time (Chen, 2019). However, the calculation of absolute return does not consider the time frame. For example, Investor bought a stock with 1000 units at a price of RM 1. After one week, the stock price rises to RM 1.2. The absolute return can be calculated using the formula as shown below:

$$Absolute\ Return = \frac{Ending\ Value - Begining\ value}{Begining\ Value} \times 100\% \quad \text{(2-10)}$$

Calculation of absolute return is much more simple compare to the calculation of rolling return. It shows the investor the total return gained for a certain period but not a clear insight for investment. False appearance might be given by the absolute return. Sometimes the absolute return shows impressive results when the stock price rises rapidly (Maheshwari, 2017). The value of the absolute return calculated based on the time period of its rising might be very high but when it comes to measure the other returns such as rolling return, compound annual growth rate (CAGR), and trailing return, the value might be not that impressive.

2.3.2 Compound annual growth rate (CAGR)

The compound annual growth rate is not an actual rate of return compared to the absolute return. CAGR calculates the annual return of an investment for a certain period. An example is illustrated to have a better understand on CAGR. An investor invests RM 10000 on a mutual fund on Jan 1, 2016. On next year, Jan 1, 2017, the fund grows to RM 12000 and reached RM 13000 on Jan 1, 2018. The equation to calculate CAGR is expressed as:

$$CAGR\ (\%) = (\frac{Ending\ Value}{Beginning\ Value})^{\frac{1}{n}} - 1 \times 100\% \qquad (2\text{-}11)$$

$$= \left(\frac{13000}{10000}\right)^{\frac{1}{2}} - 1 \times 100\% = 14.10\%$$

This shows that the mutual fund grows at an average rate of 14.10% year on year. The estimation of CAGR spoke to the rate at which a speculation would have developed on the off chance that it had developed a similar rate each year and the benefits were reinvested toward the part of the bargain. It tends to be utilized to smooth returns with the goal that they might be all the more effectively comprehended when contrasted with elective speculations.

There are some limitations on CAGR. First, the volatility of the mutual fund is ignored. CAGR assumes that the mutual fund growing steadily year on year. The second limitation is the growth of the mutual fund might not grow as the CAGR. Another limitation is it might not be that accurate and wrong insight might provide since it only considers the ending value and beginning value through the tenure (Murphy, 2019).

2.3.3 Rolling Return

Rolling return is also known as rolling period return which is the annualized average return for a certain period such as days, months, and years at a given starting date to the latest date. Rolling return provides more accurate insight into a portfolio’s performance for investors to evaluate. Sunil Subramaniam, a managing director of Sundaram mutual fund said that “Rolling returns is a bullet proof method to understand market volatility as against trailing return. This method helps you make the decision by looking at the probability of the past by looking at the probability of future” (Kaur, 2018).

Steps to calculate the rolling return for a mutual fund are described below:

1. Select a starting date and interval period (day, month, or year).
2. Calculate the return percentage for the interval period.
3. Repeat the calculation for every interval until latest date.
4. Rolling return is plotted on a graph for analysis.

Equation of calculate return percentage is:

$$Return\ percentage = \frac{Y-X}{X} \times 100\% \qquad (2\text{-}12)$$

Where

Y is the end of the period’s price

X is the beginning of the period’s price

2.3.4 Summary

In this project, image processing techniques are used to extract data from the graph of mutual funds. The data is analyses by computing some calculation such as rolling return to let the investor has a better insight and decide whether it is worth to invest in the fund. Mutual fund is one of the investment. Mutual Fund Company pool money from the investor and use that money to invest on a number of securities such as stocks and bonds. The manager of the Mutual

Fund Company will do the research and monitor the performance of the securities. The performance of the portfolio is the main factor that investors will refer to.

The reason for choosing rolling return as a method to determine the performance of mutual fund is because the return is calculated at different intervals of time in detail. Any bias due to the return can be avoided. Therefore, the determination of rolling return from the graph of mutual funds is more reliable on giving better insight into investment.

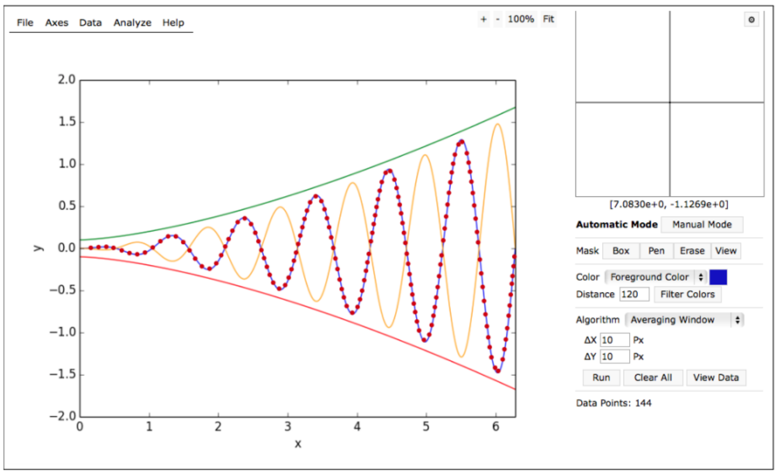
2.4 Review on others similar software

There are some similar software available in the internet which are used to extract value from the graph. The concept from these software can be used to develop for this project.

2.4.1 WebPlotDigitizer

WebPlotDigitizer was developed by a student at the University of Notre Dame. This software can be download from the URL: https://automeris.io/WebPlotDigitizer/.

Basic image processing techniques are used to develop an automatic detection algorithm. HTML5 APIs are used in this software to create a web-based software. Thousand people used this software every day and the developer is still continued to maintain and upgrade the software (Rohatgi, 2015).



[7.0830e+0, -1.1269e+0]

Color Foreground Color

Distance 120

Algorithm Averaging Window

ΔX 10 Px

ΔY 10 Px

Data Points: 144

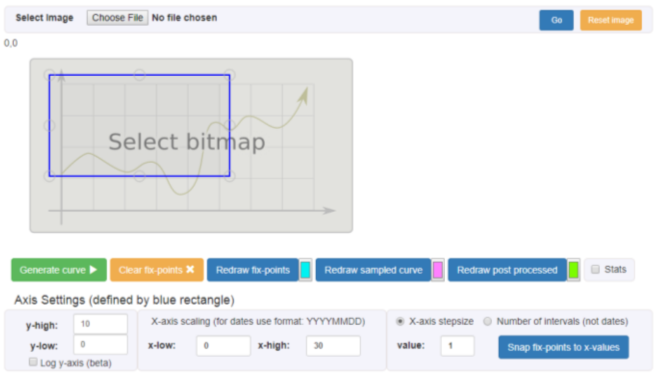
Figure 2.7: User Interface for the WebPlotDigitizer

Figure 2.4 shows the user interface for the software. Users are required to upload or load the image into the software and it can be done by several methods such as drag & drop operation, browse a file on a hard disk, copy-paste from clipboard, and webcam capture. It is able to support different image formats like JPEG, PNG, BMP, and GIF. This software is able to extract the value from a graph, bar chart, polar diagram, ternary diagram and maps. Users are required to insert input based on their image. For example, the x-axis and y-axis scale, colour of the line, data points.

There are two modes in this software, one is the manual mode and another one is the automatic mode. In manual mode, users required insert point one by one or draw a line along the graph on the image using the tools in the software. Data will be extracted based on the x-axis and y-axis scale and it can be exported to excel file or CSV file. In automatic mode, Users required to select the two x-axis points and two y-axis points. Value for these points need to be inserted too. After that, users need to select the colour of the object in the image so that the software can recognize the desired object. As manual mode, output data can be exported to the CSV file.

2.4.2 Graphreader

Graphreader is also a software used to extract the data from a graph. The steps to extract the data is almost the same as the manual mode of WebPlotDigitizer. Users need to upload an image file, draw a blue rectangle to set a ruler for axis scaling, insert the value for the x-axis and y-axis, and click on the image to insert the curve-fix point.



Select Image: No file chosen

Axis Settings (defined by blue rectangle)

y-high: 10
y-low: 0
☐ Log y-axis (beta)

X-axis scaling (for dates use format: YYYYMMDD)
x-low: 0 x-high: 30

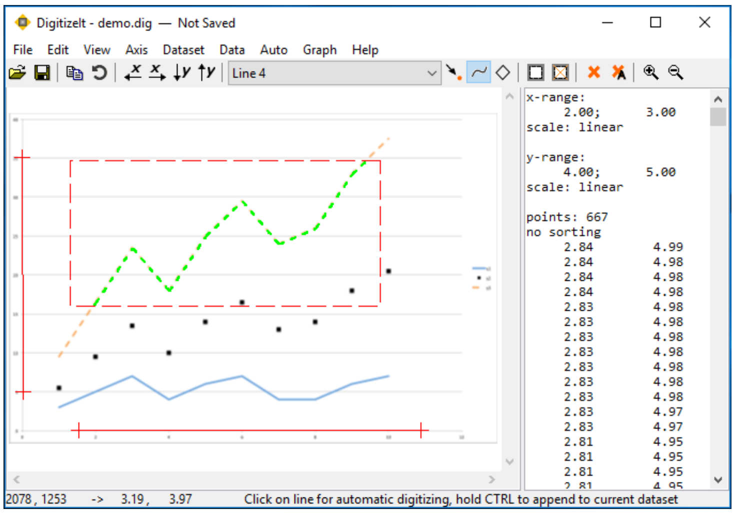
[✓ X-axis stepsize] Number of intervals (not dates)
value: 1

Figure 2.8: User interface of Graphreader

Figure 2.5 shows the user interface of the Graphreader. It is a web-based software and developed using web-programming and python (Larsen, n.d.). It available on URL: http://www.graphreader.com/. This software does not have the ability to automatic detection of a line in the graph. User required to insert the curve-fix points one by one and it is time-consuming.

2.4.3 DigitizeIt

Another similar software named DigitizeIt is available on the internet. It has the same function as other software mentioned above. This software has two digitizing modes which are automatic and manual digitizing mode (Bormann, 2012). In manual mode, users required to select data using mouse clicks. For automatic mode, line or scatter plots will be detected automatically. Data extracted will be exported in CSV file also.


DigitizeIt - demo.dig — Not Saved
File Edit View Axis Dataset Data Auto Graph Help
Line 4

```
x-range:
     2.00;      3.00
scale: linear

y-range:
     4.00;      5.00
scale: linear

points: 667
no sorting
     2.84        4.99
     2.84        4.98
     2.84        4.98
     2.84        4.98
     2.83        4.98
     2.83        4.98
     2.83        4.98
     2.83        4.98
     2.83        4.98
     2.83        4.98
     2.83        4.98
     2.83        4.97
     2.83        4.97
     2.81        4.95
     2.81        4.95
     2.81        4.95
     2.81        4.95
```

```
2078 , 1253    ->    3.19 ,    3.97        Click on line for automatic digitizing, hold CTRL to append to current dataset
```

Figure 2.9: Screenshot from DigitizeIt software

2.4.4 Comparison of software and Conclusion

The software mentioned above are compared with each other. Table 2.3 shows the comparison of these software.

Table 2-3: Comparison of software

	WebPlotDigitizer	**Graphreader**	**Digitizelt**
Image format	JPEG, PNG, BMP, GIF	PNG, JPG	INCL, GIF, PNG, TIFF, JPEG, BMP
Plot Type	Graph, Bar Chart, Plot diagram, Ternary Diagram, Maps	Graph	Graph
Output method	CSV files	CSV file	CSV file
Price	Free	Free	Free trial version for 21days.
Features	-Manual and automatic mode -Able to work with different type of chart -Open source and cross-platform	-Simple and easy to use	-Manual and automatic mode -Accept for all common image format -Different axes system: linear, logarithmic and reciprocal

Each software has its own features and advantages. Based on the table, WebPlotDigitizer and Digitizelt has more features compare to Graphreader. WebPlotDigitizer has the capability for more plot type compares with the other two software. Although Digitizelt is only capable of line graph, but it has different features to make the extraction data from the graph more complete. Both WebPlotDigitizer and Digitizelt has been cited by publications in their works. Therefore, this two software will have better reliable and trusty.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

The flowchart of extraction data from image is shown in figure 3.1.

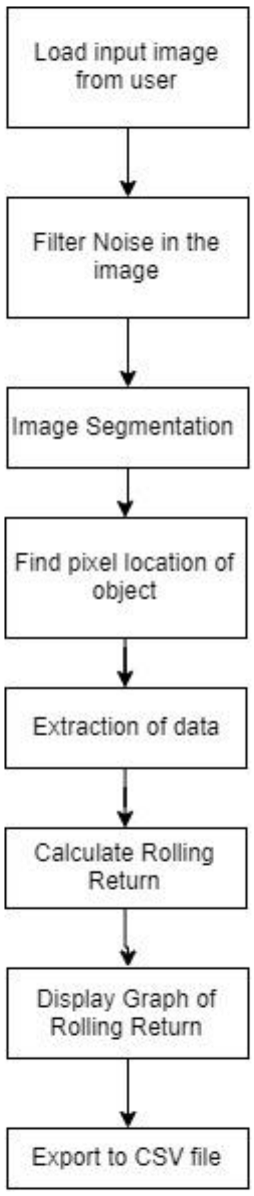

Load input image from user
Filter Noise in the image
Image Segmentation
Find pixel location of object
Extraction of data
Calculate Rolling Return
Display Graph of Rolling Return
Export to CSV file

Figure 3.1: Flowchart of the extraction data

In this project, data from graph of the mutual funds need to be extracted for calculation of rolling return. Image processing techniques are required to develop the software. Graphical user interface also needs to be developed to achieve a user-friendly environment. User load a graph of the mutual fund into the program and it able to calculate the rolling return of the mutual fund based on the user's requirement such as the period of rolling return, rolled for how many years, and the interval period of every rolling.

3.2 Programming tools

There are different programming languages available for image processing in the market. For example, python, MATLAB, C++, and Java. However, the most commonly used programming languages for image processing are python and MATLAB.

MATLAB is a Math-and matrix-oriented language and also a high-level language. It provides matrix manipulations, function, and data plotting. It also has the features for creation of a user interface. It is commonly used because of its high performance of scientific computing especially on matrix calculation since every digital image consists of a matrix of colour and colour intensity.

Python is categorized as a high-level and general-purpose programming language. It can be used for web applications, scientific computing, and numeric operation. It supports extensive library, open-source, and community development. OpenCV is one of the most popular libraries used in python to compute image processing.

In this project, Python is used for developing the program, and the OpenCV library is used for image processing because it is easier to learn, open-source, and free.

3.2.1 OpenCV

OpenCV also known as OPEN source Computer vision library and acquired by Intel now. This library is written in C++ programming languages. However, it has bindings with Python, Java, and MATLAB. The purpose of OpenCV is to provide the tools for solving computer vision problems. High-level algorithms and low-level image processing functions are included in this library. These functions are useful for face and pedestrian detection as well as matching and tracking of features. With the aids of this library, image processing techniques such as the Thresholding, filtering, and contour detection. Therefore, OpenCV is the core of this project for developing a software that can determine the rolling return of mutual funds using image processing.

3.2.2 Tkinter

Another library used in this project is Tkinter. Tkinter is a standard Graphical User Interface (GUI) library for Python. An easy and effective way to create GUI applications is provided by this library when using python because it is a powerful object-oriented interface to the Tk GUI toolkit.

Tkinter module can be downloaded in any python IDE. After downloaded and installed the Tkinter module, a GUI application main window can be created by typing the initializing code. Any event or widget can be added to the main window.

Some of the Tkinter widgets will be used commonly in this project such as button widget, entry widget, and list box widget. All the widgets can be placed at any desired location within the created window. Fonts, colour, and sizes of the widgets can be changed as wanted.

Figure 3.2 shows the example of GUI using Tkinter. Users can select input, load graph, and generate the results in this window providing convenience to users. This is the basic structure of the GUI.

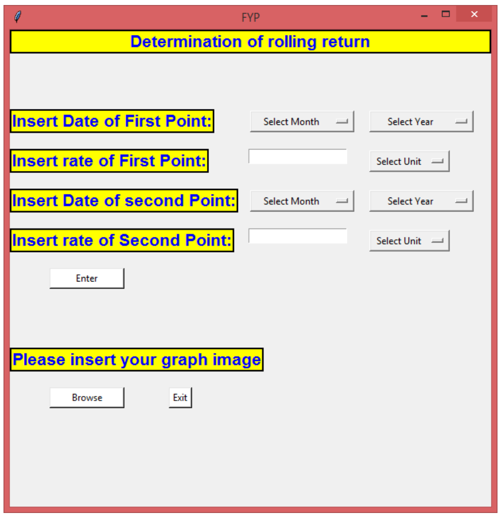

FYP
Determination of rolling return
Insert Date of First Point:
Select Month
Select Year
Insert rate of First Point:
Select Unit
Insert Date of second Point:
Select Month
Select Year
Insert rate of Second Point:
Select Unit
Enter
Please insert your graph image
Browse
Exit

Figure 3.2: Example of GUI using Tkinter

3.3 Image pre-processing

Image pre-processing in this project involved filtering and image segmentation. Filtering is used to remove the noise of the image or blur the image. In this project, the Gaussian filter is used. The original image is a colour image. The colour Image will go through the Gaussian filter and the output image from the filter will be blurred. The function in OpenCV that performs Gaussian filtering is called *GaussianBlur ()*. The function requires 3 parameters which is input image, kernel size and sigmaX. SigmaX is the standard deviation in X direction. In this case, sigmaX is equal to 0 which means it will be calculated from kernel size.

Image segmentation is a process that segment the object in the image. In this project, the main object needs to be segmented is the line that represents the percentage growth of fund in the image. The graph of mutual funds usually contains two lines. One of the lines is the benchmark of percentage growth of fund and another line is the actual value of the percentage growth. These two lines will be represented as two different colours. Image segmentation needs to differentiate these two lines and segment the line of the actual value of the percentage growth. First, the colour of the line needs to be identified. When users insert the image, first point, and last point into the software, the original image will pop out in a new window, user is able to click on the line that needed to segment. At the same time when user click on the line, the colour of the line are read in the form of Hue, Saturation, and Value (HSV). The X and Y pixel locations are also recorded. The operation is done by using a function called *setMouseCallBack()*. This function has two parameters which are window name and function name to be called when click event detected. Once the HSV value of the line graph is get from the click event, then Hue in HSV will go through a series of if-else conditions in order to get values of lower and upper boundaries for the filtering process afterward. According to the Figure 3.3, Hue represented as X-axis and Saturation represented as Y-axis while Value keeps at 255.

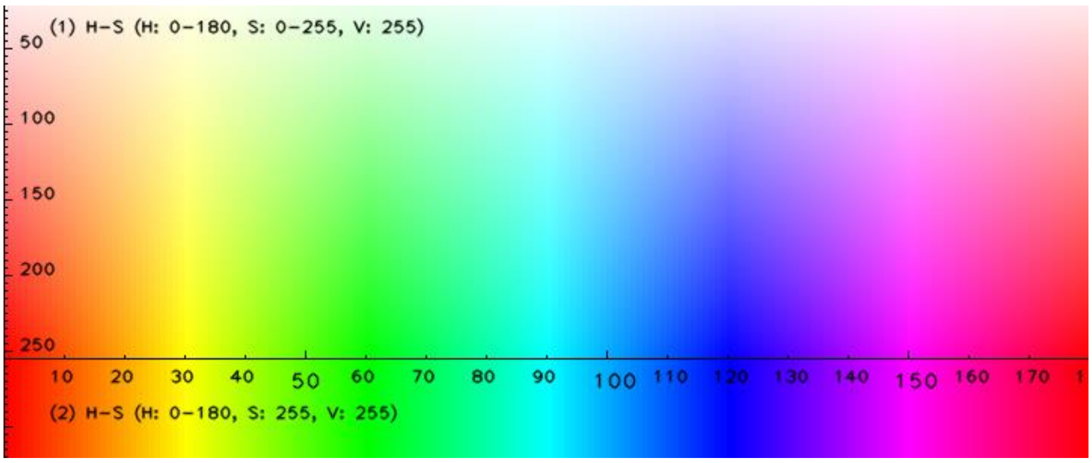


Figure 3.3: HSV in 2-Dimension

Different ranges of Hue represent for different colours. For instance, Hue value gets from the click event is 60, it is located in range of green colour where green colour has range between 36 and 83. The lower boundary is 36 and the upper boundary is 83 for green colour. Both boundary will be used as parameters in the function called *inRange()*. It contains three parameters which is input image, lower boundary, and upper boundary. This function is used to generate a mask that segmented object is white colour while others are black colour. The logic of this function is it compares all pixel value in the input image with the boundaries, if the pixel value is inside the boundaries then the pixel value will reset as 255 which is white colour, else, will be reset as 0 which is black colour. Therefore, the mask will become a black and white image with only the segmented line in white colour.

3.4 Determine the pixel location of the line

After the selected line in the graph has been segmented, the pixel location of along the line can be found by finding the contours of the line. Contours are commonly used for applications such as shape detection and recognition. Contours are the curve along the boundary of the lines. OpenCV has a function called *findCountours()* that can find out the contours of the object and store the pixel location of every contours point in an array. The function takes three parameters which are input source, mode, and method. In this case, the input source used is the mask generated earlier during the segmentation process. *CV_RETR_TREE* had been chosen for the mode. This mode means all contours along the line will be retrieved based on the hierarchy of nested contours. *CV_CHAIN_APPROX_NONE* is used for the method which it will stores absolutely all the contour points. Therefore, all pixel locations of the line can be obtained continuously by using this function and parameters.

The pixel locations from the contours are stored in the form of an array. Now the input values given by the user can be mapped to the array. The first element and last element in the array will mapped to the first point and last point that provided by user respectively. The remaining value along the lines can be determined by calculation. The steps of the calculation are shown as below:

1. Find Y-interval (Percentage return per pixel) by dividing the total percentage return within the date with the total number of Y-axis pixels in contours array. The formula is shown in equation 3-1.
2. Find X-interval (Days per pixel) by dividing the total days within the date with the total number of X-axis pixels in contours array. The formula is shown in equation 3-2.
3. Estimate the percentage return and date for each pixel in contours array by multiplying pixel location with the intervals. The formula is shown in equation 3-3 and 3-4.

$$Y - interval = \frac{Last\ Percentage\ return - Fisrt\ Percentage\ return}{Total\ number\ of\ y{-}axis\ pixels\ in\ contours\ array.} \tag{3-1}$$

$$X - interval = \frac{Last\ Date - Fisrt\ Date}{Total\ number\ of\ x{-}axis\ pixels\ in\ contours\ array.} \tag{3-2}$$

$$Percentage\ return\ for\ Yth\ pixel = \left((Y_{TH} - Y_0) * Y_{int}\right) * 100\%) + P_0 \tag{3-3}$$

where

Y_{TH} = Y th Y-axis Pixel Location

Y_0 = First Y-axis Pixel Location

Y_{int} = Y-Interval

P_0 = Initial Percentage Return

$$Date\ for\ Xth\ pixel = \big((X_{TH} - X_0) * X_{int}\big) + D_0 \qquad (3\text{-}4)$$

where

X_{TH} = X th X-axis Pixel Location

X_0 = First X-axis Pixel Location

X_{int} = X-Interval

D_0 = Initial Percentage Return

The next step after extraction of data is calculating the rolling return. The step of calculating rolling return is explained earlier in Chapter 2.3.3. The software program is designed to calculate rolling return for one, three, and five years. It depends on the available data of the mutual fund. For example, the graph of mutual fund starts from 2014 to 2020 where there are total six years data available. The program will calculate the rolling return for one, three, and five years. However, if the mutual fund has data for four years only, the program will only calculate the rolling return for one and three years. All graphs of rolling return will be plotted and shown in a new window. Matplotlib library is used to plot all graphs. One of the most common functions in Matplotlib to plot graph is *plot ()*. This function takes few parameters but only three parameters are used in this case which are x-axis data, y-axis data, and colour of line.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

The application can detect the line of graph and convert pixel location of the line into each of the data. Each data consists of date and price/percentage return. An excel file will be exported by the application. There are few tables can refer in Appendix A. First table consists of all the converted date and price/percentage return. Second table consists of one-year rolling returns for the period selected by user. Third table and fourth table will be three-year and five-year rolling return respectively if available. Graph of rolling return will be plotted out and user can make use of these graphs for references. Every part of the program from user interface to output will be discussed as below.

4.2 User Interface

Figure 4.1 shows the user interface of the software application. Users are required to insert the initial date, initial rate, final date, and final rate. Rate is the price or percentage return on the specific date. The unit for extracted data will be converted to percentage return if the input unit is "RM". The minimum duration of the date that allows the users to insert is 2 years. If the users insert dates with duration less than two years, an "Error" window will pop up and require users to re-insert the date again. Figure 4.2 shows the "Error window" that alerts users for invalid input. Users can insert the image by clicking the "Browse" button and clicking "Exit" button to terminate the application. After users insert all the inputs, a new window with the original image will be pop up. Users are required to left-click on the line that user want to extract. Once the data extracted from the image, a window will pop up to tell the user the data is exported to excel file. Figure 4.3 shows the "Output" window that mentions excel file is exported.

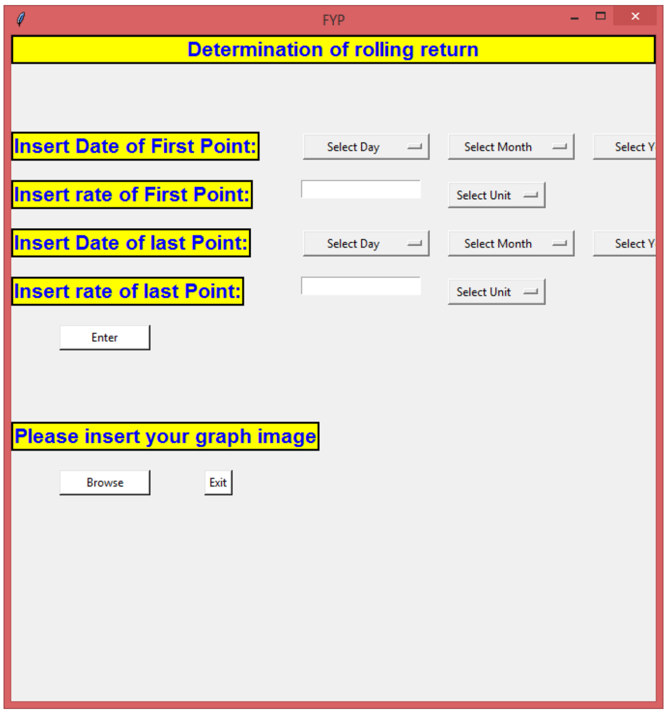
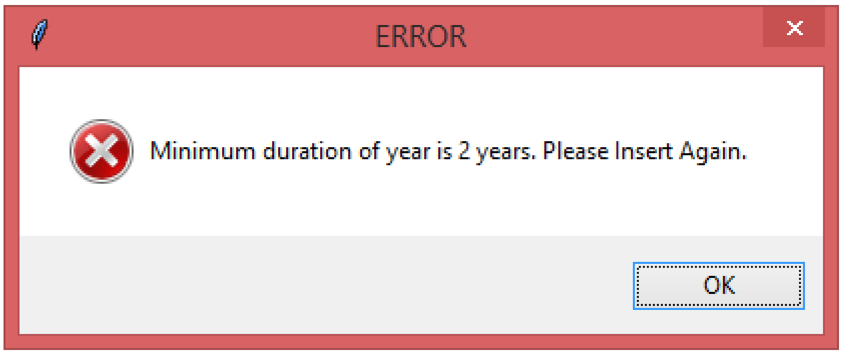
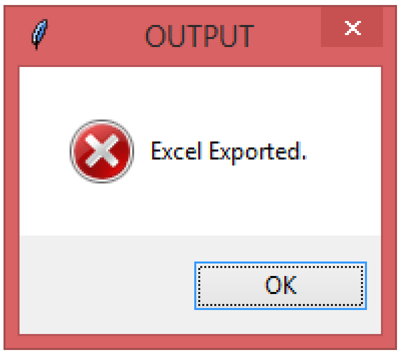

FYP
Determination of rolling return
Insert Date of First Point:
Select Day
Select Month
Select Y
Insert rate of First Point:
Select Unit
Insert Date of last Point:
Select Day
Select Month
Select Y
Insert rate of last Point:
Select Unit
Enter
Please insert your graph image
Browse
Exit

Figure 4.1: User Interface


ERROR

Minimum duration of year is 2 years. Please Insert Again.

Figure 4.2: Error window


OUTPUT

Excel Exported.

Figure 4.3: Output window

4.3 Results

There are 30 samples tested using the software application. The image size of these samples are selected at around 600 pixels for its width and any value for its height. Rolling returns for different years will be calculated and recorded. The results will be compared with the actual value from the fact sheet in order to determine the accuracy of the software application. A sample mutual fund named “Dana Makmur Pheim” is used to demonstrate the details procedure of getting results and the remaining samples are focus on percentage differences. Outputs of the program are the excel file and graph of rolling return. Excel file contains a few tables as mentioned in the introduction of this chapter. In the sub chapter, it will show some of the problems encounter in the simulation results.

4.3.1 Example - Dana Makmur Pheim

The example of the output excel file is shown in Appendix A. Graph of rolling return will pop up after an excel file is exported. After users insert image a window that shows the original image will pop up. User left-click on the line need to be extract and the output excel file will be exported. Figure 4.4 shows the original image that users inserted. For study purposes, a window named “mask” will be pop up after user left-click the line on the image. This mouse click event will retrieve the HSV value of that particular pixel. The mask is the segmentation of the selected line by using the HSV value. In this example, users clicked on the green line and all of the green color pixels will be segmented as white color while other pixels as black color. Figure 4.5 shows the segmentation of the line.

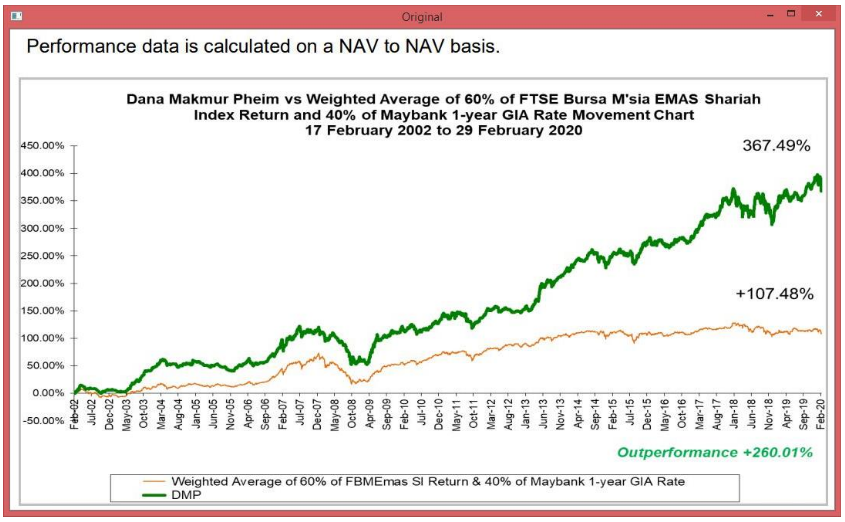


Figure 4.4: Original image of Dana Mukmur Fund

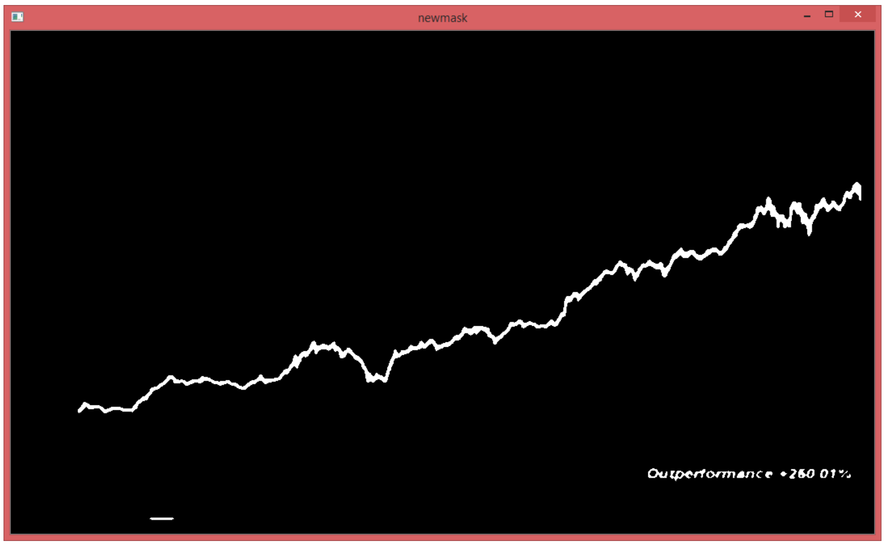


Figure 4.5: Segmentation of line (Dana Makmur Fund)

Rolling return will be calculated after all of these processes. The graph of rolling return will be plotted using matplotlib function. Figure 4.6 shows the one-year rolling return of the mutual fund. For experienced investors, they can understand the performance of the mutual fund based on the rolling return. Rolling return provide a more visual and accurate information for investors rather than annualized return.

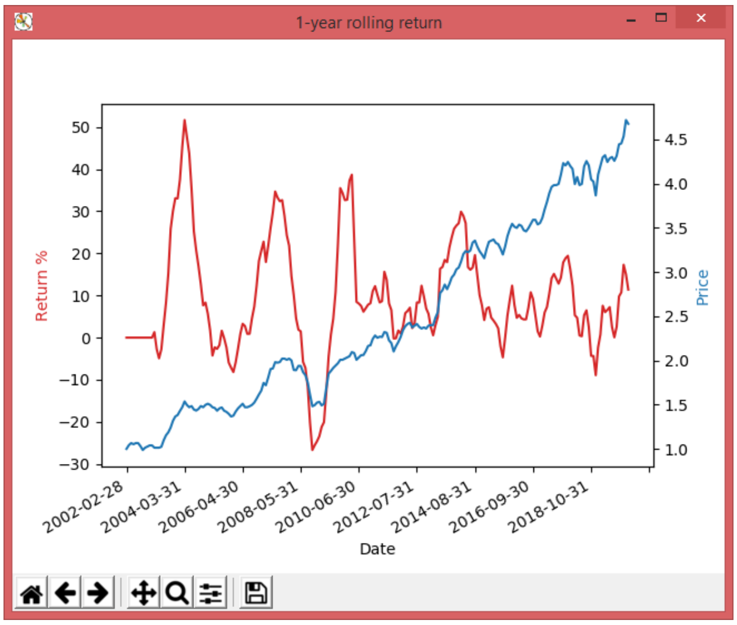


Figure 4.6: 1-Year Rolling Return of Dana Makmur Fund

The accuracy of this application can determine by comparing the 1-year return in the excel file with the actual value in the factsheet. In this example, the actual 1-year return mentioned in the factsheet is +4.73% (28/2/19 – 29/02/20) and the calculated one-year return in the excel file is 8.83%. If the duration of the mutual fund is more than three or five years, the application will also calculate the rolling return for three years and five years. So, three years and five years annualized return can be obtained for comparison. These values are tabulated in Table 4.1 and the three-year and five-year rolling return are shown in Figure 4.7 and 4.8 respectively.

Table 4-1: Comparison between actual value and calculated value (Dana Makmur Pheim)

	1-year Percentage Return (%)	3-year Percentage Return (%)	5-year Percentage Return (%)
Actual	4.73	19.29	31.38
Calculated	12.33	26.10	39.9
Percentage Different Between Both	7.6%	6.81	8.52

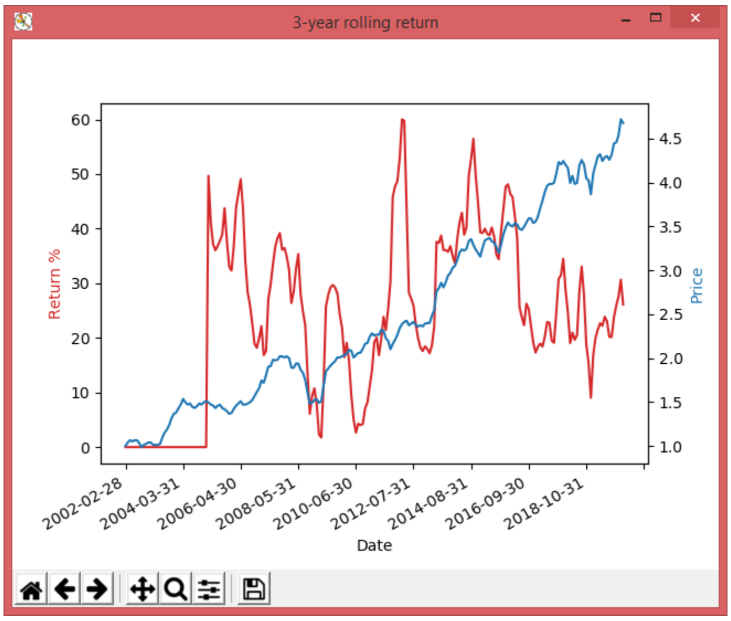


Figure 4.7: 3-Year Rolling Return of Dana Makmur Fund

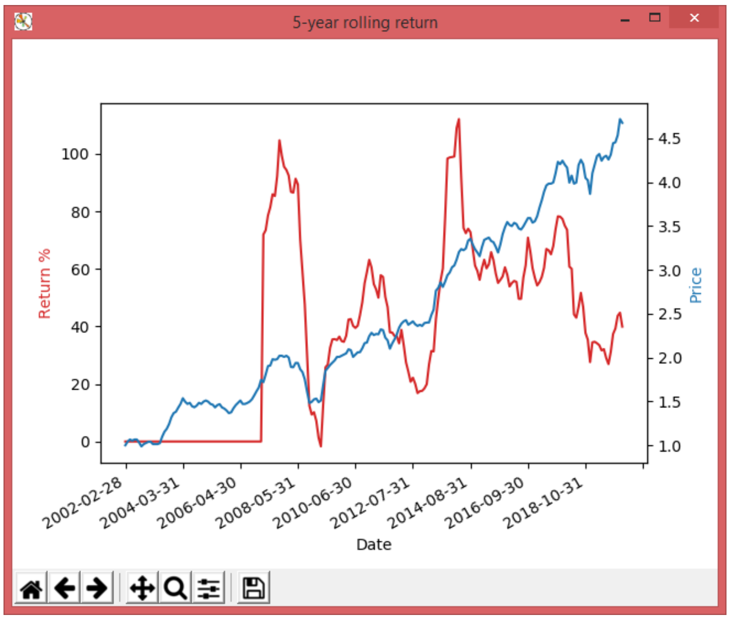


Figure 4.8 : 5-Year Rolling Return of Dana Makmur Fund

4.3.2 Example - AmBond Fund

Figure 4.9 shows the original image of AmBond Fund. Figure 4.10 shows segmentation of line. Figure 4.11 shows the 1-year rolling return of AmBond Fund. Figure 4.12 shows the 3-year rolling return of AmBond Fund. Figure 4.13 shows the 5-year rolling return of AmBond Fund.

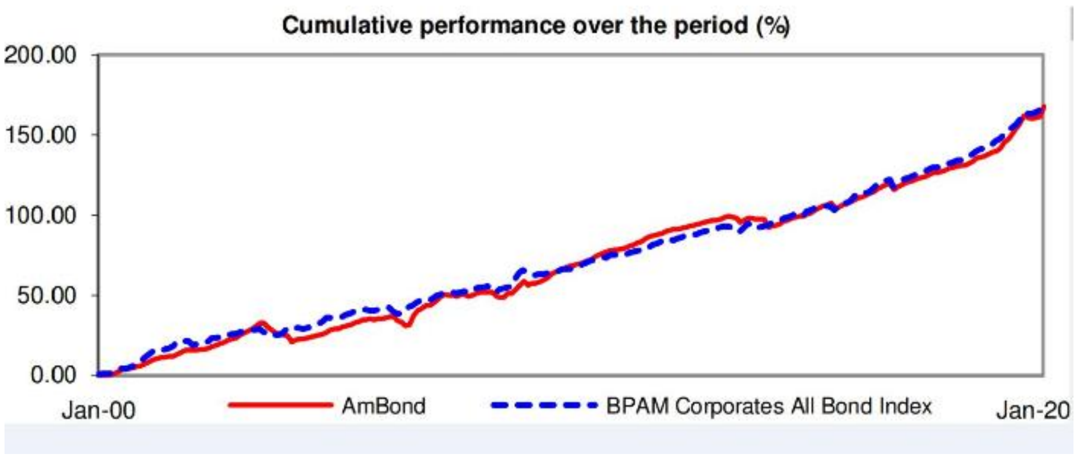


Figure 4.9: Original Image of AmBond Fund

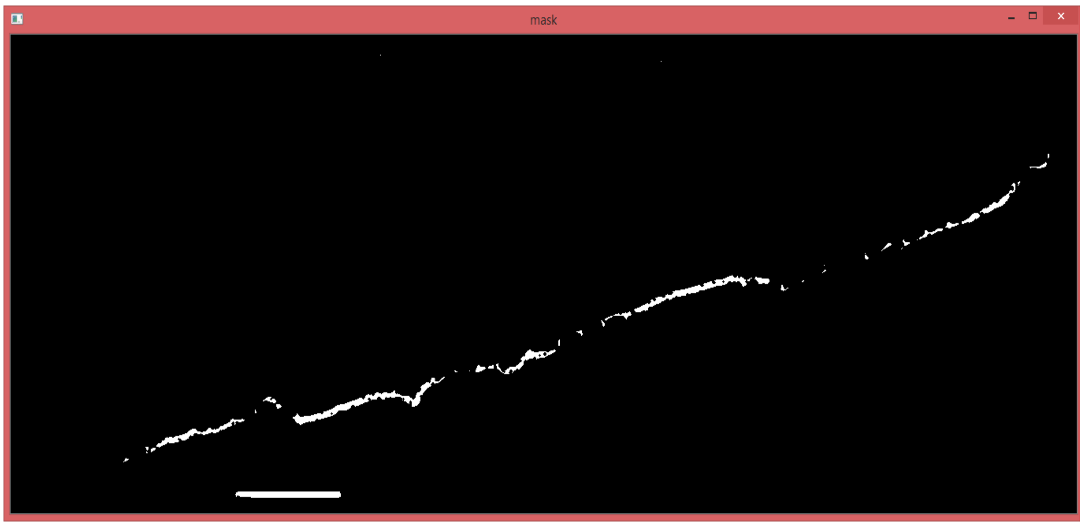


Figure 4.10: Segmentation of line (AmBond Fund)

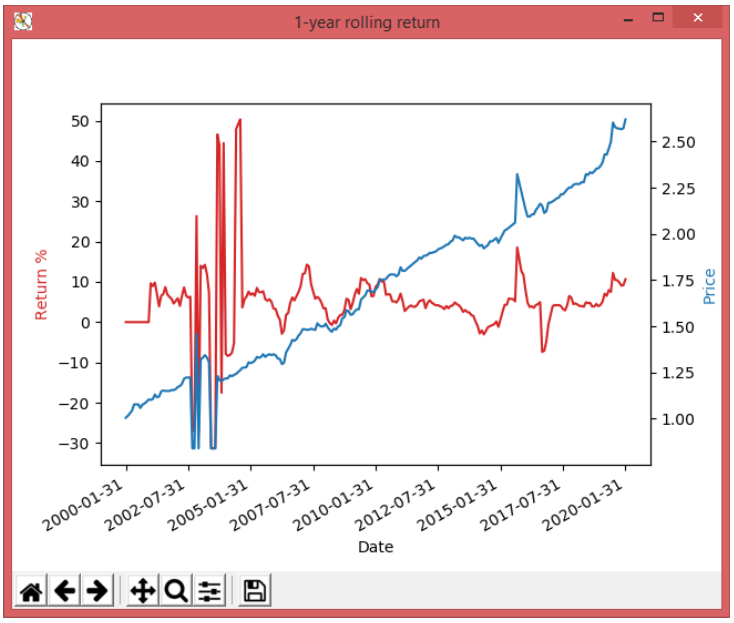


Figure 4.11: 1-Year Rolling Return of AmBond Fund

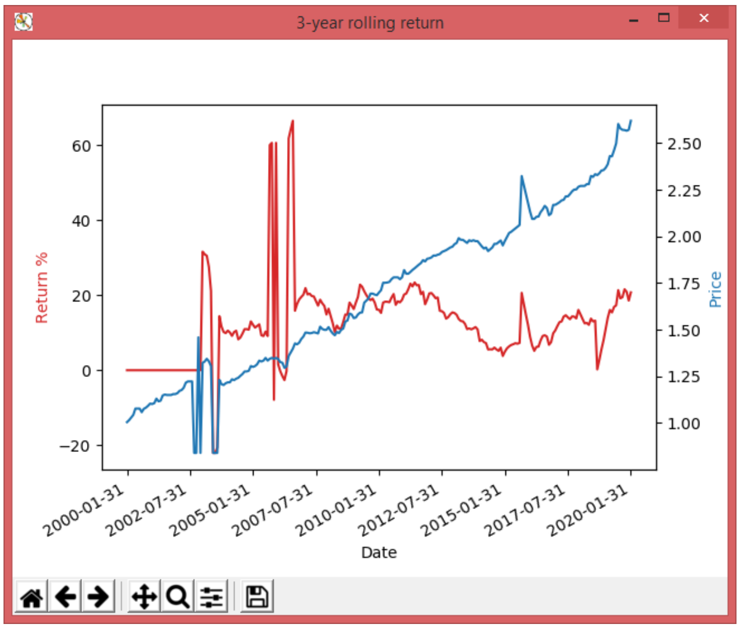


Figure 4.12: 3-Year Rolling Return of AmBond Fund

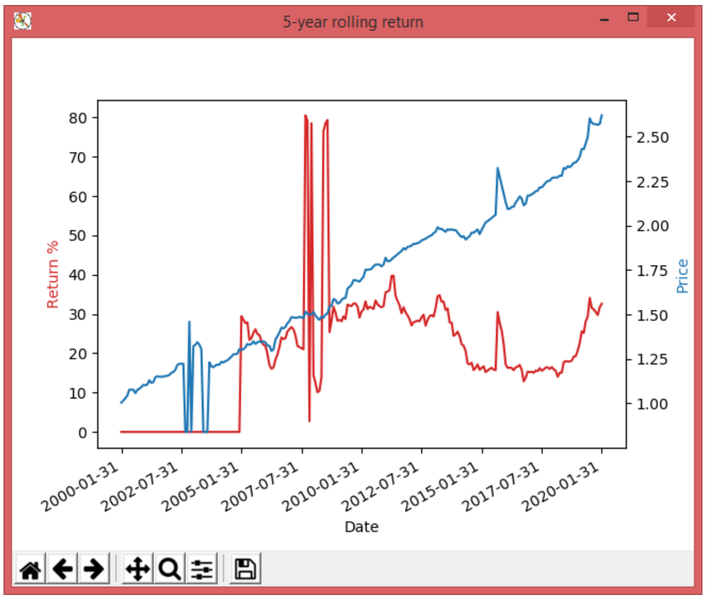


Figure 4.13: 5-Year Rolling Return of AmBond Fund

Table 4-2: Comparison between actual value and calculated value (AmBond Fund)

	1-year Percentage Return (%)	3-year Percentage Return (%)	5-year Percentage Return (%)
Actual	11.41	22.06	33.40
Calculated	10.65	20.76	32.59
Percentage Different Between Both	0.76	1.30	0.81

4.3.3 Example - Apex Dynamic Fund

Figure 4.14 shows the original image of Apex Dynamic Fund. Figure 4.15 shows segmentation of line. Figure 4.16 shows the 1-year rolling return of Apex Dynamic Fund. Figure 4.17 shows the 3-year rolling return of Apex Dynamic Fund. Figure 4.18 shows the 5-year rolling return of Apex Dynamic Fund.

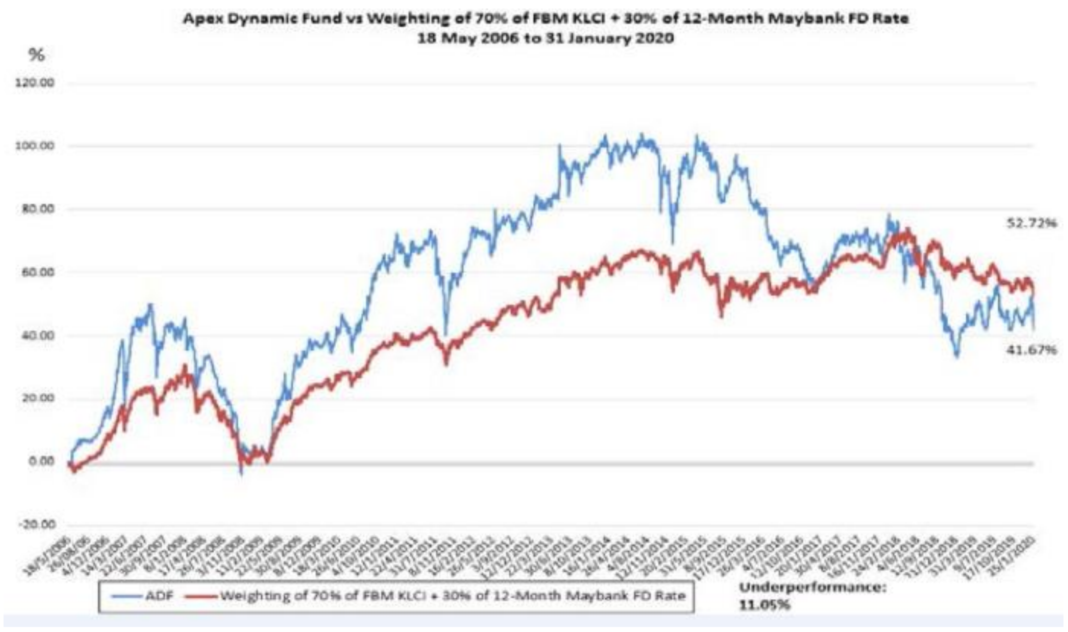


Figure 4.14: Original Image of Apex Dynamic Fund

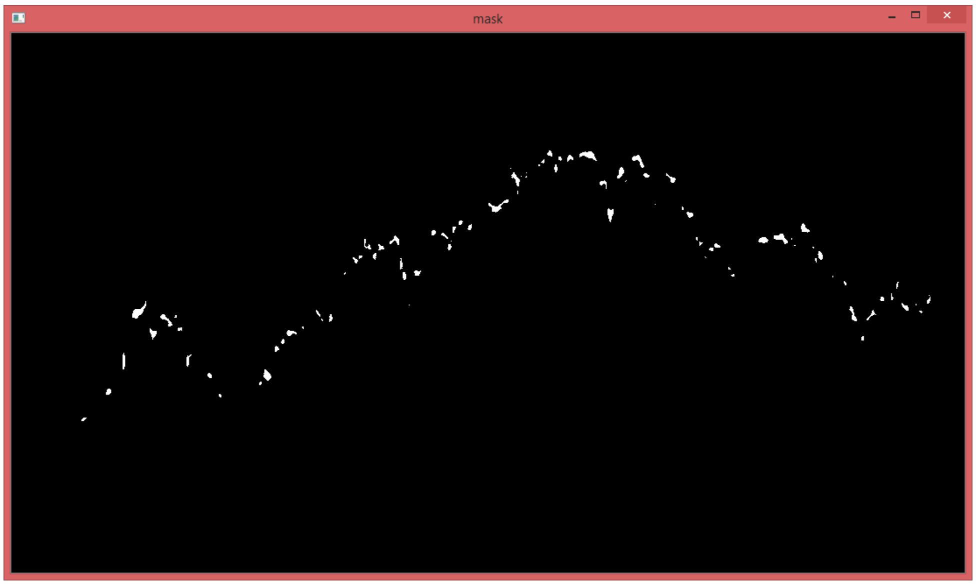


Figure 4.15: Segmentation of line (Apex Dynamic Fund)

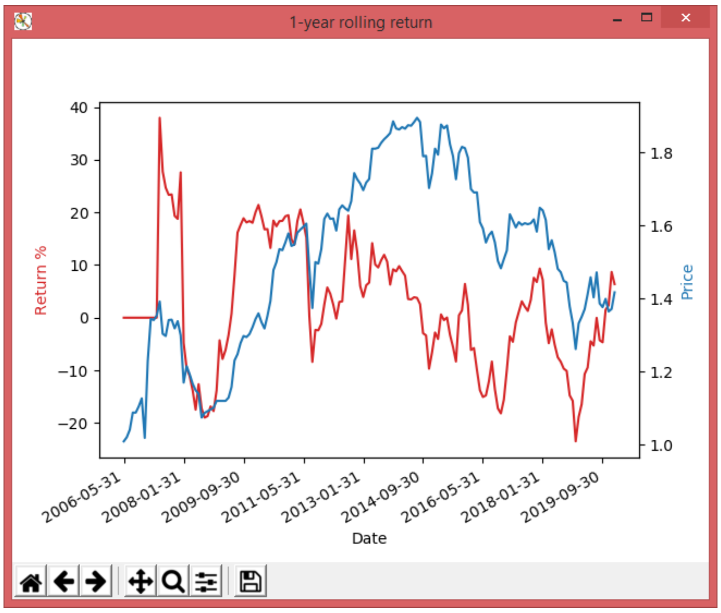


Figure 4.16: 1-Year Rolling Return of Apex Dynamic Fund

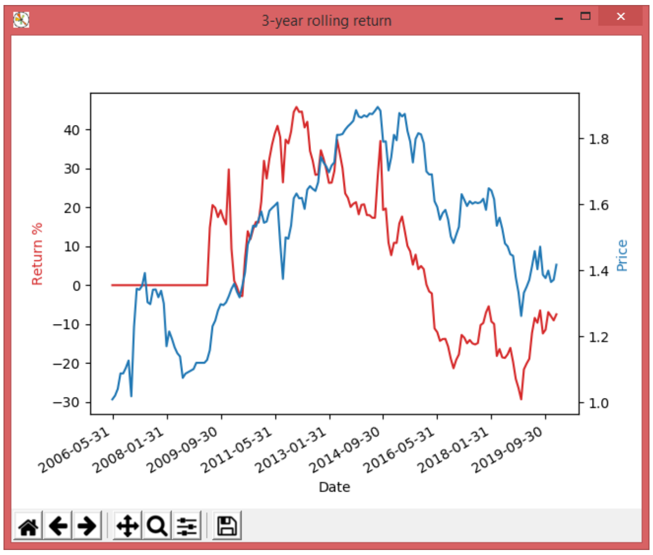


Figure 4.17: 3-Year Rolling Return of Apex Dynamic Fund

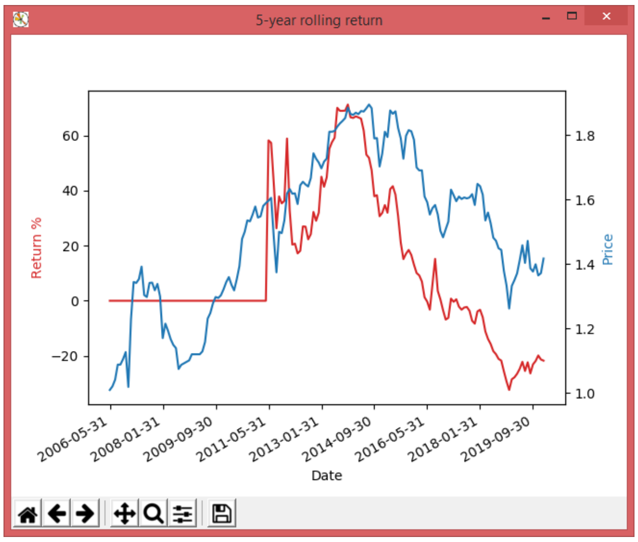


Figure 4.18: 5-Year Rolling Return of Apex Dynamic Fund

Table 4-3: Comparison between actual value and calculated value (Apex Dynamic Fund)

	1-year Percentage Return (%)	3-year Percentage Return (%)	5-year Percentage Return (%)
Actual	-0.74	-11.26	-26.01
Calculated	6.38	-7.47	-21.73
Percentage Different Between Both	7.12	3.79	4.28

4.3.4 Example - CIMB Principal Strategy Bond Fund

Figure 4.19 shows the original image of CIMB Principal Strategy Bond Fund. Figure 4.20 shows segmentation of line. Figure 4.21 shows the 1-year rolling return of CIMB Principal Strategy Bond Fund. Figure 4.22 shows the 3-year rolling return of CIMB Principal Strategy Bond Fund. Figure 4.23 shows the 5-year rolling return of CIMB Principal Strategy Bond Fund.

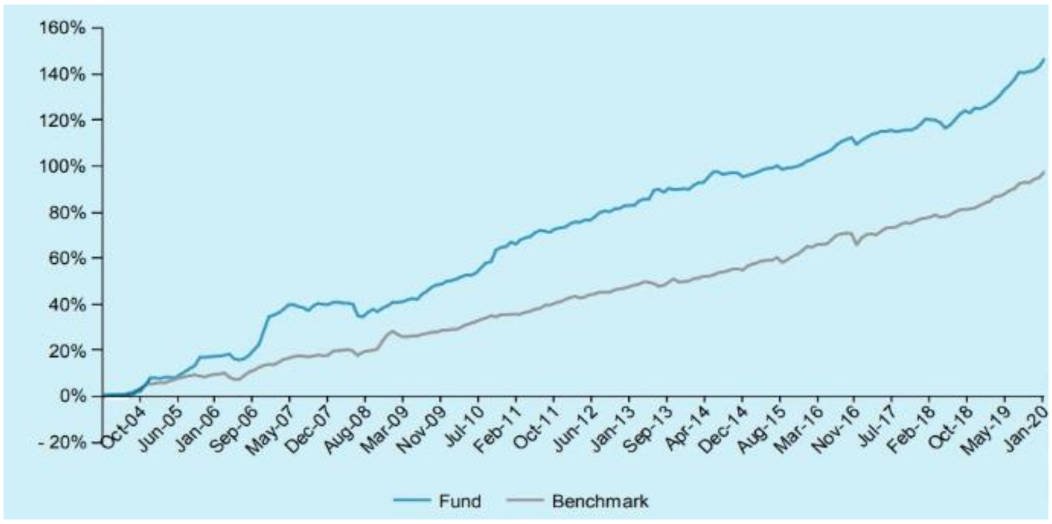


Figure 4.19: Original Image CIMB Principal Strategy Bond Fund

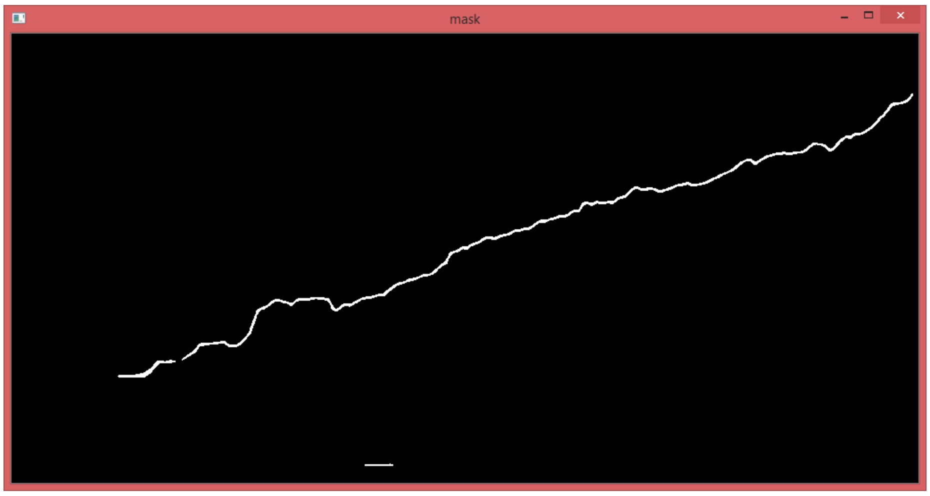


Figure 4.20: Segmentation of line (CIMB Principal Strategy Bond Fund)

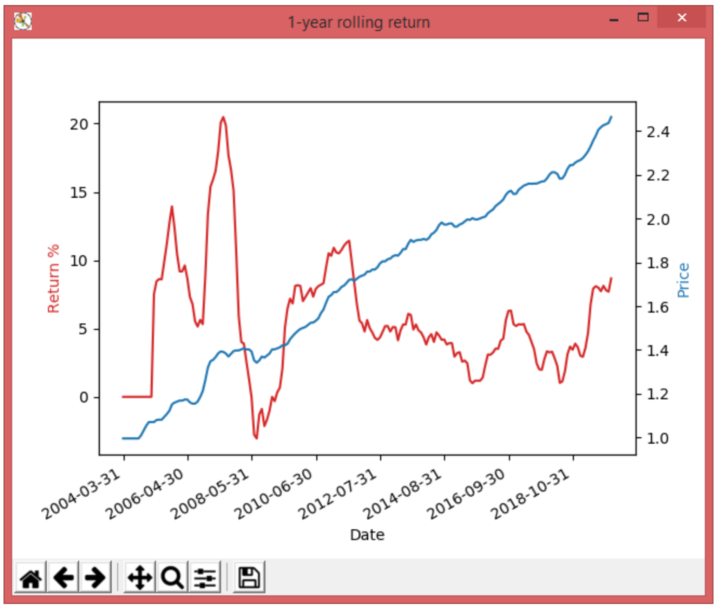


Figure 4.21: 1-Year Rolling Return of CIMB Principal Strategy Bond Fund

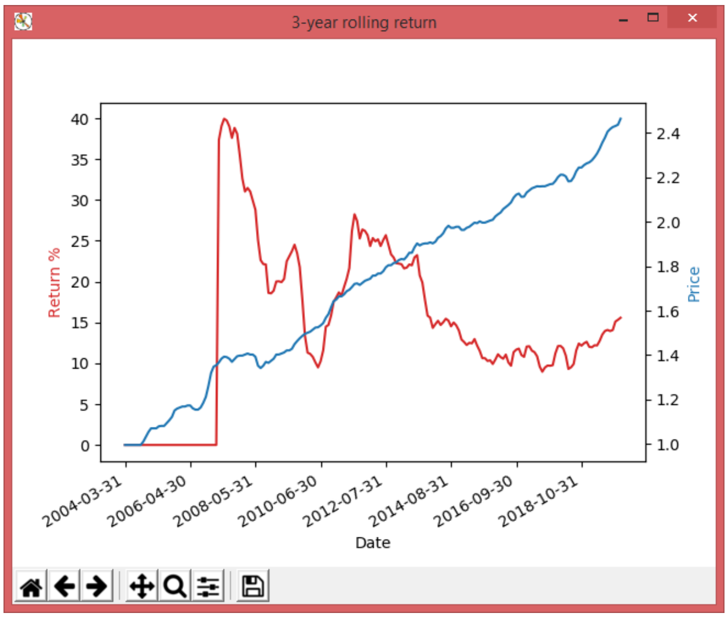


Figure 4.22: 3-Year Rolling Return of CIMB Principal Strategy Bond Fund

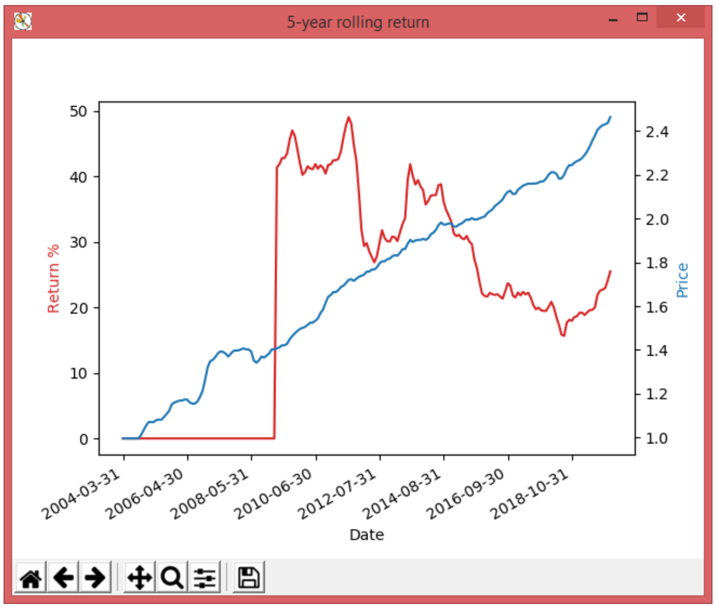


Figure 4.23: 5-Year Rolling Return of CIMB Principal Strategy Bond Fund

Table 4-4: Comparison between actual value and calculated value (CIMB Principal Strategy Bond Fund)

	1-year Percentage Return (%)	3-year Percentage Return (%)	5-year Percentage Return (%)
Actual	9.15	15.96	25.68
Calculated	8.66	15.57	25.49
Percentage Different Between Both	0.49	0.39	0.19

4.4 Discussion

Total of thirty samples were tested by using the software application. Four samples were shown above as representative of the major situation encountered. Other samples were shown in Appendix B. Percentage different between both actual and calculated value was calculated in order to determine the accuracy of the software application. The summary of the percentage difference is shown in Table A-3 to Table A-5 in Appendix A. These tables also contain actual value and the calculated value of these samples. The average of the percentage difference is used to determine the accuracy of the software application. Based on the table, the average percentage difference for one-Year Return is 1.84%, 2.09% for three-Year Return, and 2.22% for five-Year Return. The larger the number of years, the higher the percentage difference. This shows that the application becomes inaccurate when due with higher number of years.

There are few possible problems that cause inaccurate of results. The first problem is incomplete of segmentation. For example, the segmented line is incomplete and discontinuous compare with the original image which shows in Figure 4.14 and Figure 4.15. When this kind of problem happened, the software will determine the missing or discontinuous value by using interpolation. Interpolation is a mathematical calculation for estimating a value between two values. Since interpolation is used for estimation, it might cause some variation when compared with the original value.

The next problem is misidentifying other objects with the same colour as the line. This happened in the samples shown in Figure 4.10. The object in the image such as legend will have the same colour as the line. When performing segmentation, the legend will be segmented together with the line which shown in Figure 4.24 as an example where it will be two y-axis pixels in one x-axis pixel.

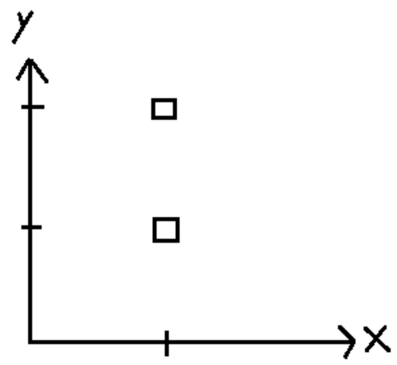


Figure 4.24: Two Y-axis Pixels with same X-axis Pixel Location

According to the segmentation algorithm in the software, it will select the higher pixel location as one of the pixels of the line since the majority of the lower pixel location usually will be pixels of legend. Therefore, misidentify of pixels will occur when the pixel of the line did not segment completely and lead to select the pixel at lower location as the pixel of the line.

It will cause a glitch in the extracted data and also affected the graph of rolling return. Figure 4.11 shows one of the examples of the problem.

However, in some cases, even though the line is almost completely segmented, the percentage difference results in high value. For example in Figure 4.5, the line looks perfectly segmented but the percentage different for one, three, five years returns are 7.6 %, 6.81 %, 8.52 % respectively which is higher than the average percentage different. This is due to the pattern of line. In figure 4.25, the tail part of the line consists of pixels that arrange in vertical. As mentioned above, the software part will select the highest pixel location as the pixel of the line and ignore the others at a lower pixel location. In this situation, the pixel label as "2" in Figure 4.25 should has 367.49 % return but the software will select the pixel label as "1" as 367.49 % return. Therefore, this causes the high percentage difference between actual and calculated value even though it is completely segmented.

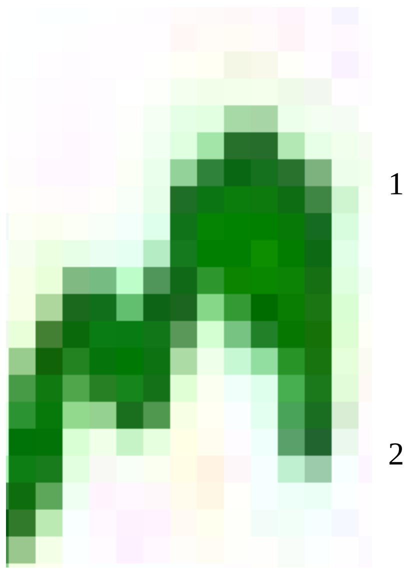


Figure 4.25: Tail part of line

Besides, the most challenging parts of this project are minimizing the user input and improving the accuracy of results. In this project, users are required to insert six inputs which are first date, first price, last date, last price, image, and HSV of the line. All the inputs are necessary in order to estimate and extract the data, none of them can be excluded in the process. Low accuracy of result is very critical for user or investor. Even though the percentage difference of the results are around 1 to 2 %, it will also affect decision of the user or investor. There are too many different types and patterns of graphs. It is very hard to consider every situation at the same time when improving the accuracy.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusion

As conclusion, image processing act as a reverse-engineer solution to extract the raw data from the image and perform further analysis or calculation to achieve certain goals in this project. It reduces the work done by the users or investors in getting raw data or information that is not for free in the market. Image processing makes human work become more easily and effectively especially in the world that generates tons of data daily. Advanced technology such as computer vision is developed based on the image processing algorithms and brings more benefits to the world. Different kind of applications can be created or developed using image processing with more creative and innovative ways not only in the financial field.

This project is able to achieve the aims and objectives mentioned in Chapter 1. The software has a graphical user interface which is user-friendly. The steps to use the software are simple and easy. Users only need to insert inputs and click on the line of the graph that wanted to extract. The software application is able to extract data from a rasterized graph using image processing and determine rolling return. The accuracy of the data needs to improve in the future in order to provide an accurate insight to users. On the other hand, users need to insert 6 inputs to extract the data. User might need to take some time to insert the inputs. The number of inputs still can be minimized by using more advanced techniques. However, this software is considered to have lesser steps for inserting inputs when compared with the software discussed before in Chapter 2.4. This software requires users to click on the line for segmentation but the other software requires insert as many points as possible or highlight the line using tools provided.

5.2 Recommendation

Recommendation on improving the accuracy of data is to improve the quality of the image during image pre-processing. For example, using the image with higher pixel resolution. The higher the pixel resolution means the larger the number of pixels in the image. During the simulation, an input image with higher resolution will get better results because high resolution has more details of the image. The trade-off of using high pixel resolution is longer processing time. It will take longer time to process the image compare with low pixel resolution due to the large number of pixels involved in the calculation. Moreover, the recommendation on minimizing the user input is using text recognition such as Optical Character Recognition (OCR). OCR will read and recognize text or word in the image. It can be used to recognize the date and percentage return in the input image which will be used as x and y-axis. Users no longer need to insert the input date and percentage return if the OCR can works well. However, this method still needs to consider the situation that will causes low accuracy due to different types and patterns of the image. Machine learning will be able to solve the problem caused by different types and patterns of the image. A large number of data set will be needed to feed into a program that able to learn the different types and patterns of the image. Then, the program can detect the line automatically and extract the desired data. The power of machine learning is it able to learn, predict, and improve itself.

REFERENCES

Basavaprasad, B. and Ravi, M., 2014. A Study on The Importance of Image Processing and Its Application. *IJRET: International Journal of Research in Engineering and Technology*, 03(03), pp. 155-160.

Bormann, I., 2012. *DigitizeIt.* [Online] Available at: https://www.digitizeit.de/ [Accessed 22 August 2019].

Chandel, R. and Gupta, G., 2013. Image Filtering Algorithms and Techniques: A review. *International Journal of Advanced Research in Computer Science and Software Engineering,* 3(10).

Chen, J., 2019. *Absolute Return.* [Online] Available at: https://www.investopedia.com/terms/a/absolutereturn.asp
[Accessed 23 August 2019].

Cook, J., 2009. *Three algorithms for converting color to grayscale.* [Online] Available at: https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/
[Accessed 20 August 2019].

Felber, P., 2002. *Charged-Coupled Devices.* [Online] Available at: http://www.ece.iit.edu/~pfelber/ccd/project.pdf

Gedraite, E.S. and Hadad, M., 2011. Investigation on the Effect of a Gaussian Blur in Image Filtering and Segmentation. *Proceedings ELMAR-2011,* pp. 393-396.

Kaur, A., 2018. *Is 'Rolling Return' the Best Way to Measure Performance of Mutual Funds?.* [Online] Available at: https://economictimes.indiatimes.com/mf/analysis/is-rolling-return-the-best-way-to-measure-performance-of-mutual-funds/articleshow/66968371.cms?from=mdr
[Accessed 19 April 2020].

Larsen, K. P., n.d. *Graphreader.* [Online] Available at: http://www.graphreader.com/
[Accessed 22 August 2019].

Maheshwari, S., 2017. *Understanding Absolute, Trailing & Rolling Returns to Measure MF Performance.* [Online] Available at: https://www.moneycontrol.com/news/business/mutual-funds/understanding-absolute-trailing-rolling-returns-to-measure-mf-performance-3838141.html
[Accessed 23 August 2019].

Murphy, C., 2019. *Compound Annual Growth Rate.* [Online] Available at: https://www.investopedia.com/terms/c/cagr.asp
[Accessed 23 August 2019].

Rohatgi, A., 2015. *WebPlotDigitizer*. [Online] Available at: https://automeris.io/WebPlotDigitizer/index.html
[Accessed 22 August 2019].

Saravanan, C., 2010. Color Image to Grayscale Image Conversion. *In 2010 Second International Conference on Computer Engineering and Applications,* Volume 2, pp. 196-199.

Sezgin, M. and Sankur, B, 2004. Survey over Image Thresholding Techiniques and Quantitiative Performance Evaluation. *Journal of Electronic imaging,* 13(1), pp. 146-165.

Tyagi, T. and Mishra, V., 2016. 2D Gaussian Filter for Image Processing: A Study. 3(06), pp. 22-24.

APPENDIX

Appendix A: Tables

Table A-1: Sample of extracted data shown in excel file

X	Y	Date	Price
83	471	2002-02-17	1
84	472	2002-02-23	0.986922
85	471	2002-03-02	1
86	471	2002-03-09	1
87	470	2002-03-16	1.013078
88	469	2002-03-23	1.026156
89	468	2002-03-29	1.039234
90	467	2002-04-05	1.052312
91	466	2002-04-12	1.06539
92	465	2002-04-19	1.078468
93	464	2002-04-26	1.091546
94	465	2002-05-02	1.078468
95	466	2002-05-09	1.06539
96	467	2002-05-16	1.052312
97	467	2002-05-23	1.052312
98	467	2002-05-30	1.052312
99	467	2002-06-05	1.052312
100	467	2002-06-12	1.052312
101	467	2002-06-19	1.052312
1031	214	2019-10-15	4.36103
1032	212	2019-10-22	4.387185
1033	207	2019-10-29	4.452575
1034	206	2019-11-05	4.465653
1038	207	2019-12-02	4.452575
1039	207	2019-12-09	4.452575
1040	206	2019-12-16	4.465653
1041	202	2019-12-22	4.517965
1042	201	2019-12-29	4.531043
1043	200	2020-01-05	4.544121
1044	199	2020-01-12	4.557199
1045	199	2020-01-19	4.557199
1046	200	2020-01-25	4.544121
1047	187	2020-02-01	4.714134
1048	187	2020-02-08	4.714134
1049	188	2020-02-15	4.701056
1050	189	2020-02-22	4.687978
1051	190	2020-02-29	4.6749

Table A-2: Sample of Rolling Return data

Date	Price	1-Year Rolling Return
2019-02-28	4.196401636	-1.54161687
2019-03-31	4.295639858	5.084950679
2019-04-30	4.319551869	4.960834308
2019-05-31	4.243328114	8.483225424
2019-06-30	4.28435701	6.220032015
2019-07-31	4.299647036	2.457747915
2019-08-31	4.25640605	1.07975315
2019-09-30	4.317339779	1.248130515
2019-10-31	4.455394808	6.675811343
2019-11-30	4.452575089	10.80306538
2019-12-31	4.533413651	15.2036065
2020-01-31	4.714133808	10.55342179
2020-02-29	4.6749	12.33752668

Table A-3: Summary of Percentage Different for 1 Year

	1 - Year Percentage Return		
	Actual	Calculated	Difference
Dana Makmur Pheim	4.73	12.33	7.6
Pheim Income Fund	0.19	3.61	3.42
Kenaga Bond Fund	5	4.61	0.39
Kenaga Bond Extra Fund	9.15	8.74	0.41
Kenaga Premier Fund	15.86	17.54	1.68
AmSean Equity Fund	-0.49	1.51	2
Am Bond Fund	11.41	10.65	0.76
Apex Asian(Ex-Japan)Fund	-3.64	6.85	10.49
Apex Dynamic Fund	-0.74	6.38	7.12
EastSpring Investment Asia Pacific(Ex-Japan) Target Return	8.81	9.45	0.64
EastSpring Bond Fund	8.66	8.71	0.05
Pacific Dana Aman	5.05	6.72	1.67
Pacific Pearl Fund	10.43	11.13	0.71
Pacific Premier Fund	-1.17	0.59	1.76
United ASEAN Discovery	17.86	21.24	3.38
United Bond Equity Strategy	13.94	14.22	0.28
United Global Quality Equity - MYR Hedged	21.08	22.39	1.31
United Gloden Opprtunity MYR Hedged	19.67	18.84	0.83
United Income Plus	12.02	12.67	0.65
United Target Income Bond Fund	3.05	2.64	0.41
Opus Cash Extra Fund	4.1	4.6	0.5
Opus Dynamic Fund	12.39	9.98	2.41
Opus Global Income	3.39	2.95	0.44
Aberdeen Standard Islam Pacific Ex Japan Equity	0.19	1.65	1.46
Affin Hwang Absolute Return Fund	12.8	14.8	2
Affin huang Growth Fund	2.1	3.97	1.87
CIMB Principal Strategy Bond Fund	9.15	8.66	0.49
CIMB Principal Small Cap Fund	17.63	18.56	0.93
CIMB Islamic PRS Plus Conservative-Class A	6.44	6.8	0.36
CIMB Islamic PRS Plus Conservative-Class C	6.25	6.49	0.14
		Average:	1.872

Table A-4: Summary of Percentage Different for 3 Year

	3 - Year Percentage Return		
	actual	calculated	difference
Dana Makmur Pheim	19.29	26.1	6.81
Pheim Income Fund	4.91	7.79	2.88
Kenaga Bond Fund	12.76	12.89	0.13
Kenaga Bond Extra Fund	19.01	18.84	0.17
Kenaga Premier Fund	6.8	9.77	2.97
AmSean Equity Fund	3.06	-1.2	4.26
Am Bond Fund	22.06	20.76	1.3
Apex Asian(Ex-Japan)Fund	5.42	3.66	1.76
Apex Dynamic Fund	-11.26	-7.47	3.79
EastSpring Investment Asia Pacific(Ex-Japan) Target Return	13.91	13.98	0.07
EastSpring Bond Fund	17.83	17.53	0.3
Pacific Dana Aman	-11.41	-9.92	1.49
Pacific Pearl Fund	-16	-14.72	1.27
Pacific Premier Fund	-5.11	-4.91	0.2
United ASEAN Discovery			
United Bond Equity Strategy			
United Global Quality Equity - MYR Hedged			
United Gloden Opprtunity MYR Hedged			
United Income Plus			
United Target Income Bond Fund			
Opus Cash Extra Fund	10.75	12.56	1.82
Opus Dynamic Fund	22.39	21.18	1.21
Opus Global Income	-1	1.26	2.26
Aberdeen Standard Islam Pacific Ex Japan Equity	-4.9	0.1	5
Affin Hwang Absolute Return Fund	21.1	20.24	0.86
Affin huang Growth Fund	9.1	1.7	7.4
CIMB Principal Strategy Bond Fund	15.96	15.57	0.39
CIMB Principal Small Cap Fund	5.44	7.72	2.28
CIMB Islamic PRS Plus Conservative-Class A	13.23	14.08	0.85
CIMB Islamic PRS Plus Conservative-Class C	13.1	13.02	0.08
			2.1543478
		Average:	3

Table A-5: Summary of Percentage Different for 5 Year

	5 - Year Percentage Return		
	actual	calculated	difference
Dana Makmur Pheim	31.38	39.9	8.52
Pheim Income Fund	14.79	18.29	3.5
Kenaga Bond Fund	20.36	19.76	0.6
Kenaga Bond Extra Fund	35.29	36.06	0.77
Kenaga Premier Fund	9.15	11.07	1.92
AmSean Equity Fund	-4.17	-3.83	0.34
Am Bond Fund	33.4	32.59	0.81
Apex Asian(Ex-Japan)Fund	15.66	22.06	6.4
Apex Dynamic Fund	-26.01	-21.73	4.28
EastSpring Investment Asia Pacific(Ex-Japan) Target Return	19.16	21.73	2.57
EastSpring Bond Fund	37.28	38.26	0.98
Pacific Dana Aman	-15.42	-14.49	0.93
Pacific Pearl Fund	-27.43	-28.59	1.16
Pacific Premier Fund	-7.59	-8.9	1.31
United ASEAN Discovery			
United Bond Equity Strategy			
United Global Quality Equity - MYR Hedged			
United Gloden Opprtunity MYR Hedged			
United Income Plus			
United Target Income Bond Fund			
Opus Cash Extra Fund	16.75	20.88	4.13
Opus Dynamic Fund	32.29	32.56	0.17
Opus Global Income			
Aberdeen Standard Islam Pacific Ex Japan Equity			
Affin Hwang Absolute Return Fund			
Affin huang Growth Fund			
CIMB Principal Strategy Bond Fund	25.68	25.49	0.19
CIMB Principal Small Cap Fund	4.03	6.09	2.06
CIMB Islamic PRS Plus Conservative-Class A	18.26	20.58	2.32
CIMB Islamic PRS Plus Conservative-Class C	18.11	17.7	0.41
		Average:	2.28263158

Appendix B: Source Code

```
from tkinter import *
from tkinter import messagebox as mb
from PIL import Image,ImageTk
from tkinter import ttk
from tkinter import filedialog
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
import matplotlib.dates as mdates
from numpy import array
import xlsxwriter
from datetime import datetime
from datetime import timedelta
from datetime import date
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import KMeans
import colorsys
import calendar

window=Tk()
window.geometry("2048x600")  #size of window
window.title("FYP") #title of window
#variable to store the value of first and last point
fd = StringVar()
fm = StringVar()
fy = StringVar()
fr = StringVar()
fr_unit = StringVar()
sd = StringVar()
sm = StringVar()
sy = StringVar()
sr = StringVar()
sr_unit = StringVar()
input_img = np.zeros((512, 512, 3),np.uint8)
upper_RGB = []
lower_RGB = []

def print1():
    first_day = int(fd.get())
    first_month = int(fm.get())
    first_year = int(fy.get())
    first_rate = float(fr.get())
    first_unit = fr_unit.get()
    sec_day = int(sd.get())
    sec_month = int(sm.get())
    sec_year = int(sy.get())
    sec_rate = float(sr.get())
    sec_unit = sr_unit.get()
    date1 = date(first_year, first_month, first_day)
    date2 = date(sec_year, sec_month, sec_day)
    t_days = date2 - date1
    year_interval = timedelta(days=365)
    total_year = int(t_days.days / year_interval.days)
    #print(total_year)
    if (total_year <= 2):
        mb.showerror("ERROR","Minimun duration of year is 2 years.
Please Insert Again.")
        #print(f"First point is is
({first_month},{first_year},{first_rate}{first_unit}).")
        #print(f"Last point is ({sec_month},{sec_year},{sec_rate}{sec_unit}).")
    else:
        return (first_rate,sec_rate,first_year, sec_year, first_month, sec_month,
first_unit, first_day, sec_day, total_year)

def exit1():
    exit()

def click_event(event, x, y, flags, params):
    global lower_RGB
    global upper_RGB
    if event == cv.EVENT_LBUTTONDOWN:
        upper_blue = input_img[y, x, 0]
        upper_green = input_img[y, x, 1]
        upper_red = input_img[y, x, 2]
        #upper_RGB = np.array([upper_red, upper_green, upper_blue])
        print(f"UB is {upper_blue}, UG {upper_green},UR {upper_red}")
    if event == cv.EVENT_RBUTTONDOWN:
        lower_blue = input_img[y, x, 0]
        lower_green = input_img[y, x, 1]
        lower_red = input_img[y, x, 2]
        #lower_RGB = np.array([lower_blue, lower_green, lower_red])
        print(f"LB is {lower_blue}, LG {lower_green},LR {lower_red}")
        #cv.imshow('Original', input_img)
        img2 = cv.GaussianBlur(input_img, (7, 7), 0)
        #cv.imshow('blur filter', img2)
        gray = cv.cvtColor(img2, cv.COLOR_BGR2GRAY)
        #cv.imshow('gray', gray)
        th2 = cv.adaptiveThreshold(gray, 255,
cv.ADAPTIVE_THRESH_GAUSSIAN_C, cv.THRESH_BINARY, 5, 2)
        #cv.imshow('th2', th2)
        hsv = cv.cvtColor(img2, cv.COLOR_BGR2HSV)
        #cv.imshow('hsv', hsv)
        RGB_2_HSV = colorsys.rgb_to_hsv(lower_red/255, lower_green/255,
lower_blue/255)
        hue = int(RGB_2_HSV[0]*179)
        #print(hue)
        if (hue>= 0 and hue <=10):
            hue_low = 0
            hue_hi = 10
        elif (hue>= 11 and hue <=25):
            hue_low = 11
            hue_hi = 25
        elif (hue>= 26 and hue <=35):
            hue_low = 26
            hue_hi = 35
        elif (hue>= 36 and hue <=83):
            hue_low = 36
            hue_hi = 83
        elif (hue>= 84 and hue <=105):
            hue_low = 84
            hue_hi = 105
        elif (hue>= 106 and hue <=135):
            hue_low = 106
            hue_hi = 135
        elif (hue>= 136 and hue <=144):
            hue_low = 136
            hue_hi = 144
        elif (hue>= 145 and hue <=158):
            hue_low = 145
            hue_hi = 158
        elif (hue>= 159 and hue <=166):
            hue_low = 159
            hue_hi = 166
        elif (hue>= 167 and hue <=179):
            hue_low = 167
            hue_hi = 179
        red_lower = np.array([hue_low, 100, 10])
        red_upper = np.array([hue_hi, 255, 255])
        mask2 = cv.inRange(hsv, red_lower, red_upper)
        new_mask2 = cv.morphologyEx(mask2, cv.MORPH_CLOSE,
cv.getStructuringElement(cv.MORPH_OPEN, (3, 3)))

        cv.imshow("mask", new_mask2)
        contours, hierarchy = cv.findContours(new_mask2, cv.RETR_TREE,
cv.CHAIN_APPROX_NONE)
        contours = array(contours)
        #print(len(contours))
        new_con = []
        area = []
        #for hier in range(len(contours)):
            #area.append(int(cv.contourArea(contours[hier])))
        #average = sum(area) / len(contours)
        #for aa in range(len(contours)):
            #if (area[aa] >= average):
                #new_con.append(contours[aa])

        combined = np.concatenate([contours[p].reshape(-1,2) for p in
range(len(contours))])
        #array1 = contours[0].reshape(-1, 2)
        #array2 = contours[1].reshape(-1, 2)
        #array3 = contours[2].reshape(-1, 2)
        #array4 = contours[3].reshape(-1, 2)
        #array5 = contours[4].reshape(-1, 2)
        #combined = np.concatenate((array1, array2, array3, array4, array5))
        #print(combined)
        #print(len(combined))
        sort_combined = sorted(combined, key=lambda x: x[0])
        pixel_location = []
        price = []
        tdate = []
        new_date = []
        new_pl = []
        pl_xcoor = []
        pos_ret = 0
        neg_ret = 0
        no_ret = 0
        total_ret = 0
        probability_loss = 0
        # take out the similar point
        for i in range(len(sort_combined) - 1):
            if sort_combined[i + 1][0] != sort_combined[i][0]:
                pixel_location.append(sort_combined[i])
        outWorkbook = xlsxwriter.Workbook("imgpixel.xlsx")
        outSheet = outWorkbook.add_worksheet()
        # write headers
        outSheet.write("A1", "X")
        outSheet.write("B1", "Y")
        outSheet.write("C1", "Date")
        outSheet.write("D1", "Price")
        outSheet.write("F1", "Date")
        outSheet.write("G1", "Price")
        outSheet.write("H1", "1-Year Rolling Return")
        for row in range(len(pixel_location)):
            outSheet.write(row + 1, 0, pixel_location[row][0])
            outSheet.write(row + 1, 1, pixel_location[row][1])
        first_rate, sec_rate, first_year, sec_year, first_month, sec_month ,
first_unit, first_day, sec_day, total_year= print1()
        #print (first_rate,sec_rate)
        #print(pixel_location[row][0],pixel_location[0][0])
        y_interval = abs(((sec_rate - first_rate)/(pixel_location[row][1]-
pixel_location[0][1])))
        #print(f"Y_interval is {y_interval}")
        date1 = date(first_year,first_month,first_day)
        date2 = date(sec_year,sec_month,sec_day)
        t_days = date2 - date1
        #************find total years to calculate for year-rolling return
        ppd = abs((pixel_location[row][0] - pixel_location[0][0]) / t_days.days)
        #month_interval = timedelta(days = 30)
        #total_month = abs(t_days.days/month_interval.days)
        #ppm = month_interval.days * ppd

        #for i in range(int(total_month)):

#new_date.append(datetime.strftime((date1+(month_interval*(i+1))),"%Y-
%m-%d"))
            #new_pl.append(int(round(new_pl[i]+ppm)))
        for j in range(len(pixel_location)):
            pl_xcoor.append(pixel_location[j][0])
            #*********change percentage to price
            if(first_unit == 'Percentage'):
                price.append(((pixel_location[0][1]-
pixel_location[j][1])*y_interval)*0.01+ (first_rate*0.01+1))
            else:
                price.append(((pixel_location[0][1] - pixel_location[j][1]) *
y_interval) + first_rate)
            outSheet.write(j + 1, 3, price[j])
            tdelta = timedelta(days=(pixel_location[j][0] - pixel_location[0][0]) /
ppd)
            tdate.append(datetime.strftime((date1 + tdelta),"%Y-%m-%d"))
            outSheet.write(j + 1, 2, tdate[j])

        # make list with end month date
        start_month = date1.month
        end_month = (date2.year - date1.year) * 12 + date2.month
        #print(start_month, end_month)

        for month in range(start_month, end_month + 1):
            year = int((month) / 12 + date1.year)
            month = int((month) % 12 + 1)
            new_date.append(datetime.strftime(datetime(year, month, 1) -
timedelta(days=1), "%Y-%m-%d"))
            diff = (date(year, month, 1) - timedelta(days=1)) - date1
            new_pl.append(diff.days * ppd + pixel_location[0][0])

        new_price = [None] * (len(new_pl))
        percentage_ret1 = [0] * (len(new_pl))
        percentage_ret3 = [0] * (len(new_pl))
        percentage_ret5 = [0] * (len(new_pl))

        for k in range(len(new_pl)):
            if (new_price[k] == None):
                #find nearest date index
                res = min(enumerate(tdate), key = lambda
x:abs(datetime.strptime(new_date[k], "%Y-%m-%d") -
datetime.strptime(x[1], "%Y-%m-%d")))
                #print(res)
                delta = datetime.strptime(new_date[k], "%Y-%m-%d") -
```

```
datetime.strptime(tdate[res[0]], "%Y-%m-%d")
                if (delta.days > 0):
                    interpolation = ((price[res[0] + 1] - price[res[0]]) /
(pixel_location[res[0] + 1][0] - pixel_location[res[0]][0]))
                    new_price[k] = ((interpolation*(new_pl[k]-
pixel_location[res[0]][0]))+price[res[0]])
                elif (delta.days == 0):
                    new_price[k] = price[res[0]]
                elif (delta.days < 0):
                    interpolation = ((price[res[0] + 1] - price[res[0]]) /
(pixel_location[res[0] + 1][0] - pixel_location[res[0]][0]))
                    new_price[k] = (price[res[0]] - (interpolation *
(pixel_location[res[0]][0] - new_pl[k])))

            #print(new_date[k], new_pl[k], new_price[k])
            outSheet.write(k + 1, 5, new_date[k])
            outSheet.write(k + 1, 6, new_price[k])
        #1 year rolling return
        for item in range(len(new_price) - 12):
            percentage_ret1[item+12] = ((new_price[item+12] -
new_price[item])/new_price[item])*100

        #3 year rolling return
        if (total_year >= 4):
            for item3 in range(len(new_price) - 36):
                percentage_ret3[item3 + 36] = ((new_price[item3 + 36] -
new_price[item3]) / new_price[item3]) * 100

            outSheet.write("I1", "3-Year Rolling Return")
        #5 year rolling reutn
        if (total_year >= 6):
            for item5 in range(len(new_price) - 60):
                percentage_ret5[item5 + 60] = ((new_price[item5 + 60] -
new_price[item5]) / new_price[item5]) * 100

            outSheet.write("J1", "5-Year Rolling Return")

        for m in range(len(percentage_ret1)):
            outSheet.write(m + 1, 7, percentage_ret1[m])
            outSheet.write(m + 1, 8, percentage_ret3[m])
            outSheet.write(m + 1, 9, percentage_ret5[m])
        outWorkbook.close()
        mb.showerror("OUTPUT", "Excel Exported.")
        plotgraph(new_date, percentage_ret1, percentage_ret3,
percentage_ret5,new_price)
        print ('Excel Exported')
        #cv.waitKey()
        #cv.destroyAllWindows()

def plotgraph(x, y1, y3, y5, price):
    #Plot graph for 1-year return
    color = 'tab:red'
    fig1 = plt.figure(num="1-year rolling return")
    ax1 = fig1.add_subplot()
    ax1.plot(x, y1, color = color)
    ax1.set_xlabel('Date')
    ax1.set_ylabel('Return %',color = color)
    ax1.xaxis.set_major_locator(plt.MaxNLocator(10))
    color = 'tab:blue'
    ax2 = ax1.twinx()
    ax2.plot(x, price, color = color)
    ax2.set_ylabel('Price',color = color)
    #ax2.xaxis.set_major_formatter(plt.NullFormatter())
    ax2.xaxis.set_major_locator(plt.MaxNLocator(10))
    fig1.autofmt_xdate()

    #Plot graph for 3-year return
    color = 'tab:red'
    fig3 = plt.figure(num="3-year rolling return")
    ax3 = fig3.add_subplot()
    ax3.plot(x, y3, color = color)
    ax3.set_xlabel('Date')
    ax3.set_ylabel('Return %',color = color)
    ax3.xaxis.set_major_locator(plt.MaxNLocator (10))
    color = 'tab:blue'
    ax4 = ax3.twinx()
    ax4.plot(x, price, color = color)
    ax4.set_ylabel('Price',color = color)
    ax4.xaxis.set_major_locator(plt.MaxNLocator(10))
    fig3.autofmt_xdate()

    #Plot graph for 5-year return
    color = 'tab:red'
    fig5 = plt.figure(num="5-year rolling return")
    ax5 = fig5.add_subplot()
    ax5.plot(x, y5, color = color)
    ax5.set_xlabel('Date')
    ax5.set_ylabel('Return %',color = color)
    ax5.xaxis.set_major_locator(plt.MaxNLocator(10))
    color = 'tab:blue'
    ax6 = ax5.twinx()
    ax6.plot(x, price, color = color)
    ax6.set_ylabel('Price',color = color)
    ax6.xaxis.set_major_locator(plt.MaxNLocator(10))
    fig5.autofmt_xdate()
    #show all graph
    plt.show()

def get_image():
    filename=filedialog.askopenfilename(initialdir="/",title="Select A
File",filetype=(("jpeg","*.jpg"),("png","*.png"),("All Files","*.*")))
    label_dir=Label(text=filename)
    label_dir.place(x=200,y=500)
    load=Image.open(filename)
    render=ImageTk.PhotoImage(load)
    img=Label(window,image=render)
    img.image = render
    #img.place(x=450,y=50)
    global input_img
    input_img = cv.imread(filename)
    testing()

def testing():
    cv.namedWindow('Original')
    cv.setMouseCallback('Original', click_event)
    while (1):
        cv.imshow('Original', input_img)
        if cv.waitKey(20) == 27:
            break
    cv.destroyAllWindows()



def graphical_UI():
    label0=Label(window,text="Determination of rolling
return",fg="blue",bg="yellow",relief="solid",font=("arial",16,"bold"))
    label0.pack(fill="x")
    label1=Label(window,text="Please insert your graph
image",fg="blue",bg="yellow",relief="solid",font=("arial",16,"bold"))
    label1.place(x=0,y=400)

    #user insert the first point
    label2=Label(window,text="Insert Date of First
Point:",fg="blue",bg="yellow",relief="solid",font=("arial",16,"bold"))
    label2.place(x=0,y=100)
    #First Month
    list_month = ['1','2','3','4','5','6','7','8','9','10','11','12']
    droplist_fm = OptionMenu(window,fm,*list_month)
    fm.set("Select Month")
    droplist_fm.config(width=15)
    droplist_fm.place(x=450,y=100)
    #First Year
    list_year =
['1995','1996','1997','1998','1999','2000','2001','2002','2003','2004','2005','
2006','2007','2008','2009','2010','2011','2012','2013','2014','2015','2016','2
017','2018','2019','2020']
    droplist_fy = OptionMenu(window,fy,*list_year)
    fy.set("Select Year")
    droplist_fy.config(width=15)
    droplist_fy.place(x=600,y=100)
    #First day
    list_day =
['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20
','21','22','23','24','25','26','27','28','29','30','31']
    droplist_fday = OptionMenu(window, fd, *list_day)
    fd.set("Select Day")
    droplist_fday.configure(width=15)
    droplist_fday.place(x= 300,y=100)
    #First rate
    label3 = Label(window, text="Insert rate of First Point:", fg="blue",
bg="yellow", relief="solid",
                font=("arial", 16, "bold"))
    label3.place(x=0, y=150)
    entry_fr = Entry(window, textvar=fr)
    entry_fr.place(x=300, y=150)
    list_unit = ['RM','Percentage']
    droplist_unit = OptionMenu(window,fr_unit,*list_unit)
    fr_unit.set("Select Unit")
    droplist_unit.config(width=10)
    droplist_unit.place(x=450,y=150)

    #user insert last point
    label4 = Label(window, text="Insert Date of last Point:", fg="blue",
bg="yellow", relief="solid",
                font=("arial", 16, "bold"))
    label4.place(x=0, y=200)
    #Last Month
    droplist_sm = OptionMenu(window, sm, *list_month)
    sm.set("Select Month")
    droplist_sm.config(width=15)
    droplist_sm.place(x=450, y=200)
    #Second Year
    droplist_sy = OptionMenu(window,sy,*list_year)
    sy.set("Select Year")
    droplist_sy.config(width=15)
    droplist_sy.place(x=600,y=200)
    #Last Day
    droplist_sday = OptionMenu(window, sd, *list_day)
    sd.set("Select Day")
    droplist_sday.config(width=15)
    droplist_sday.place(x=300,y=200)
    #Last Rate
    label5 = Label(window, text="Insert rate of last Point:", fg="blue",
bg="yellow", relief="solid",
                font=("arial", 16, "bold"))
    label5.place(x=0, y=250)
    entry_sr = Entry(window, textvar=sr)
    entry_sr.place(x=300, y=250)
    list_unit = ['RM','Percentage']
    droplist_unit2 = OptionMenu(window,sr_unit,*list_unit)
    sr_unit.set("Select Unit")
    droplist_unit2.config(width=10)
    droplist_unit2.place(x=450,y=250)

graphical_UI();
button1=Button(window,text="Browse",width=12,fg="black",bg="white",
command=get_image)
button1.place(x=50,y=450)
button3=Button(window,text="Enter",width=12,fg="black",bg="white",co
mmand=print1)
button3.place(x=50,y=300)
button2=Button(window,text="Exit",fg="black",bg="white",command=exi
t1)
button2.place(x=200,y=450)
window.mainloop()
```