

Grabcuts for image segmentation: A comparative study of clustering techniques



University of Fort Hare
Together in Excellence

A thesis submitted in fulfilment of the requirements for the degree of



MASTER OF SCIENCE
University of Fort Hare
IN MATHEMATICS

BY

NOZUKO ZULEIKA MANZI

In the Faculty of Science and Agriculture at the University of Fort Hare

Supervised by Professor B.B. Makamba

Co-supervised by Doctor E.O.D. Andriantiana

2019

DEDICATION

To my beloved mother: Ms Darly Manzi

A true leader, pillar of strength and support that has been loving, faithful and generous even in the darkest of times. My mother is always there for me and pushes me to go after my dreams. She has taught me so much and

I will always be grateful to have her in my life.



University of Fort Hare
Together in Excellence

DECLARATION

I declare that “Grabcuts for image segmentation: A comparative study of clustering techniques” is my own work and that all the sources I have used have been indicated and acknowledged by means of complete references. I further declare that I have not previously submitted this work, or part of it, at any other university for a degree.

Name: NOZUKO ZULEIKA MANZI



Date: 12 APRIL 2019

Signature: *Nz Manzi*

University of Fort Hare
Together in Excellence

ACKNOWLEDGMENTS

I would like to express my overwhelming gratitude to:

- My supervisor Professor B.B Makamba from the Department of Mathematics at the University of Fort Hare for his unceasing motivation and guidance throughout this study.
- My co-supervisor Doctor E.O.D. Andriantiana from Rhodes University for providing me with the knowledge and skills that developed me in many different ways.
- To my beloved fiancée, Mr O.K. Overen and son Mr. O.M. Overen. Also, to my family for their profound love, advice and encouragement.
- My sponsors NRF and CSIR- Defence Peace Safety and Security for their financial support.
- Lastly and most importantly, to God almighty for keeping me alive and strengthening me to fulfil this study.



University of Fort Hare
Together in Excellence

ABSTRACT

Image segmentation is the partitioning of a digital image into small segments such as pixels or sets of pixels. It is significant as it allows for the visualization of structures of interest, removing unnecessary information. In addition, image segmentation is used in many fields like, for instance healthcare for image surgery, construction, etc. as it enables structure analysis. Segmentation of images can be computationally expensive especially when a large dataset is used, thus the importance of fast and effective segmentation algorithms is realised. This method is used to locate objects and boundaries (i.e. foreground and background) in images. The aim of this study is to provide a comparison of clustering techniques that would allow the Grabcuts for image segmentation algorithm to be effective and inexpensive. The Grabcuts based method, which is an extension of the graph cut based method, has been instrumental in solving many problems in computer vision i.e. image restoration, image segmentation, object recognition, tracking and analysis. According to Ramirez,et.al [47], the Grabcuts approach is an iterative and minimal user interaction algorithm as it chooses a segmentation by iteratively revising the foreground and background pixels assignments. The method uses min-cut/ max-flow algorithm to segment digital images proposed by Boykov and Jolly [9]. The input of this approach is a digital image with a selected

region of interest (ROI). The ROI is selected using a rectangular bounding box. The pixels inside the bounding box are assigned to the foreground, while the others are assigned to the background.

In this study, the Grabcuts for image segmentation algorithm designed by [48] with a Gaussian Mixture Model (GMM) based on the Kmeans and Kmedoids clustering techniques are developed and compared. In addition, the algorithms developed are allowed to run on the Central Processing Unit (CPU) under two scenarios. Scenario 1 involves allowing the Kmeans and Kmedoids clustering techniques to the Squared Euclidean distance measures to calculate the similarities and dissimilarities in pixels in an image. In scenario 2, the Kmeans and Kmedoids clustering techniques will use the City Block distance measure to calculate similarities as well as dissimilarities between pixels in a given image. The same images from the Berkeley Segmentation Dataset and Benchmark 500 were used as input to the algorithms and the number of clusters, K , was varied from 2 to 5.

It was observed that the Kmeans clustering technique outperformed the Kmedoids clustering technique under the two scenarios for all the test images with K varied from 2 to 5, in terms of runtime required. In addition, the Kmeans clustering technique obtained more compact and separate clusters under scenario 1, than its counterpart. On the other hand, the Kmedoids obtained more compact and separate clusters than the Kmeans clustering technique under scenario 2. The silhouette validity index favoured the smallest number of clusters for both clustering techniques as it suggested the optimal number of clusters for the Kmeans and Kmedoids clustering techniques under the two scenarios was 2. Although the Kmeans required less computation time than

its counterpart, the generation of foreground and background took longer for the GMM based on Kmeans than it did for the GMM based on Kmedoids clustering technique. Furthermore, the Grabcuts for image segmentation algorithm with a GMM based on the Kmedoids clustering technique was computationally less expensive than the Grabcuts for image segmentation algorithm with a GMM based on the Kmeans clustering technique. This was observed to be true under both scenario 1 and 2. The Grabcuts for image with the GMM based on the Kmeans clustering techniques obtained slightly better segmentation results when the visual quality is concerned, than its counterpart under the two scenarios considered. On the other hand, the BF-scores showed that the Grabcuts for image segmentation algorithm with the GMM based on Kmedoids produces images with higher BF-scores than its counterpart when K was varied from 2 to 5 for most of the test images. In addition, most of the images obtained the majority of their best segmentation results when $K=2$. This was observed to be true under scenario 1 as well as scenario 2. Therefore, the Kmedoids clustering technique under scenario 2 with $K=2$ would be the best option for the segmentation of difficult images in BSDS500. This is due to its ability to generate GMMs and segment difficult images more efficiently (i.e. time complexity, higher BF-scores, more under segmented rather than over segmented images, inter alia.) while producing comparable visual segmentation results to those obtained by the Grabcuts for image segmentation: GMM-Kmeans.




University of Fort Hare
Pursuing the Frontiers of Knowledge


Contents

List of Figures	xi
List of Tables	xvii
1. Introduction	1
1.1 Background	1
1.2 Problem statement	2
1.3 Research aims and objectives	2
1.4 Contributions of this study	4
1.5 Image segmentation techniques	4
1.5.1 Edge based segmentation	5
1.5.2 Threshold based segmentation	6
1.5.3 Automatic threshold	8
1.5.4 Region based segmentation	8
1.5.5 Watershed based segmentation	8
1.5.6 Energy based segmentation	9
1.5.7 Fuzzy based method	9
2. Literature review	11
2.1 Digital images and their importance	12
2.1.1 Interpretation of binary images	12




University of Fort Hare
Together in Excellence

2.2	Basic Graph Theory	15
2.3	Network Flow	20
2.3.1	Residual Network	21
2.4	Graph cuts	22
2.5	Graph cut based algorithms	24
2.5.1	Minimum cut/ Maximum flow algorithms	25
2.5.2	Push-relabel algorithm	26
2.5.3	Operations of the algorithm	26
2.5.4	Ford-Fulkerson algorithm	29
2.5.5	Push-Relabel versus Ford-Fulkerson algorithm: Similarities and differences	33
2.6	Advantages of Graph based segmentation	35
2.7	Delimitation of graph cut based segmentation	35
 University of Fort Hare <i>Together in Excellence</i>		
3.	Grabcuts: Theory	36
3.1	Energy minimisation framework	36
3.2	Grabcuts for image segmentation	37
3.2.1	Initialisation	41
3.2.2	Iterative minimisation	42
3.2.3	User editing	46
3.3	Grabcuts based algorithms	46
3.4	Advantages of using Grabcuts method	47
3.5	Disadvantages of using Grabcuts method	48
3.6	Clustering Techniques	48
3.7	Distance measures and metrics	53

4. Performance measures	57
4.1 Evaluation metrics	57
4.2 Intra-cluster distance	59
4.3 Inter-cluster distance	59
4.4 Validity Indices	60
4.4.1 Dunn’s validity index	60
4.4.2 Davies-Bouldin validity index	60
4.4.3 Silhouette validity index	61
4.4.4 Rand validity index	61
4.4.5 Jaccard validity index	62
4.5 Precision	63
4.6 Recall	63
4.7 BF-score	64
 University of Fort Hare <i>Together in Excellence</i>	
5. Experimentation and results	66
5.1 Experimental setup	69
5.2 Experimental results	71
5.2.1 Performance of Grabcuts for image segmentation algo- rithm under Scenario 1	71
5.2.2 Optimal number of clusters	82
5.2.3 Segmentation results	94
5.2.4 Performance of Grabcuts for image segmentation algo- rithm under Scenario 2	109
5.2.5 Optimal number of clusters	118
5.2.6 Segmentation results	130

5.3	Discussions	147
6.	Conclusion	155
6.1	Summary	155
6.2	Future work	157
	References	159
Appendix		168
A.	Central processing unit versus the Graphics processing unit	169
A.1	The central processing unit	169
A.2	The graphics processing unit	173
A.2.1	Parallelism in GPU and GPU architecture	174
A.2.2	Influences on evolution of the CPU and GPU	174
A.3	CPU versus GPU <i>Together in Excellence.</i>	174
B.	Grabcuts for image segmentation Matlab code	177
B.1	Grabcuts for image segmentation: GUI	177
B.2	Grabcuts based on GMM-Kmeans algorithm	188
B.3	Local colour model generated by GMM-Kmeans	191
B.4	Grabcuts based on GMM-Kmedoids algorithm	194
B.5	Local colour model generated by GMM-Kmedoids	197

List of Figures

1.1	Segmentation techniques	5
2.1	RGB colour space as shown by [52]	14
2.2	A graph G with its subgraph H	16
2.3	The graphs K_4 and K_5	17
2.4	Examples of digraphs	18
2.5	Network and possible flow [58] 	21
2.6	The Push-relabel algorithm as opposed to the Ford-Fulkerson algorithm [34]. <i>Together in Excellence.</i>	34
3.1	An image with selected ROI	39
5.1	Images from BSDS500 database	68
5.2	Grabcuts for image segmentation: A comparative study of clustering techniques GUI	69
5.3	Selection of ROI from image loaded in the GUI	70
5.4	Number of iterations: Kmeans versus Kmedoids clustering technique	75
5.5	Silhouette validity index versus Number of clusters for the Butterfly image	85

5.6	Silhouette validity index versus Number of clusters for the Insect image	85
5.7	Silhouette validity index versus Number of clusters for the Soldier image	86
5.8	Silhouette validity index versus Number of clusters for the Kids image	87
5.9	Silhouette validity index versus Number of clusters for the Swimmer image	87
5.10	Silhouette validity index versus Number of clusters for the Car image	88
5.11	Silhouette validity index versus Number of clusters for the Aeroplane image	88
5.12	Silhouette validity index versus Number of clusters for the Flowers image	89
5.13	Silhouette validity index versus Number of clusters for the Swan image	90
5.14	Silhouette validity index versus Number of clusters for the Statues image	90
5.15	Silhouette validity index versus Number of clusters for the Star-fish image	91
5.16	Silhouette validity index versus Number of clusters for the Horses image	92
5.17	Average runtime GMM-Kmeans versus GMM-Kmedoids	93
5.18	Average runtimes for Grabcuts for image segmentation algo- rithm: GMM-Kmeans versus GMM-Kmedoids	94



5.19 Segmentation results for Grabcuts for image segmentation:	
GMM-Kmeans versus GMM-Kmedoids	96
5.19 Segmentation results for Grabcuts for image segmentation:	
GMM-Kmeans versus GMM-Kmedoids	97
5.19 Segmentation results for Grabcuts for image segmentation:	
GMM-Kmeans versus GMM-Kmedoids	98
5.19 Segmentation results for Grabcuts for image segmentation:	
GMM-Kmeans versus GMM-Kmedoids	99
5.20 Average BF-scores for Grabcuts for image segmentation algo-	
rithm: GMM-Kmeans versus GMM-Kmedoids	108
5.21 Number of iterations: Kmeans versus Kmedoids clustering	
technique	112
5.22 Silhouette validity index versus Number of clusters for the	
Butterfly image	121
5.23 Silhouette validity index versus Number of clusters for the	
Insect image	122
5.24 Silhouette validity index versus Number of clusters for the	
Soldier image	123
5.25 Silhouette validity index versus Number of clusters for the	
Kids image	123
5.26 Silhouette validity index versus Number of clusters for the	
Swimmer image	124
5.27 Silhouette validity index versus Number of clusters for the Car	
image	124



5.28	Silhouette validity index versus Number of clusters for the Aeroplane image	125
5.29	Silhouette validity index versus Number of clusters for the Flowers image	126
5.30	Silhouette validity index versus Number of clusters for the Swan image	126
5.31	Silhouette validity index versus Number of clusters for the Statues image	127
5.32	Silhouette validity index versus Number of clusters for the Star-fish image	128
5.33	Silhouette validity index versus Number of clusters for the Horses image	128
5.34	Average runtime GMM-Kmeans versus GMM-Kmedoids . . .	130
5.35	Average runtimes for Grabcuts for image segmentation algo- rithm: GMM-Kmeans versus GMM-Kmedoids	131
5.36	Segmentation results for Grabcuts for image segmentation: GMM-Kmeans versus GMM-Kmedoids	133
5.36	Segmentation results for Grabcuts for image segmentation: GMM-Kmeans versus GMM-Kmedoids	134
5.36	Segmentation results for Grabcuts for image segmentation: GMM-Kmeans versus GMM-Kmedoids	135
5.36	Segmentation results for Grabcuts for image segmentation: GMM-Kmeans versus GMM-Kmedoids	136
5.37	Average BF-scores for Grabcuts for image segmentation algo- rithm: GMM-Kmeans versus GMM-Kmedoids	146



5.38	Average Silhouette values for Kmeans and Kmedoids clustering techniques across all test images	150
5.39	Average Silhouette values for Kmeans clustering techniques across all test images	151
5.40	Average Silhouette values for Kmedoids clustering techniques across all test images	151
A.1	Machine cycle of a CPU [38].	172



University of Fort Hare
Together in Excellence

List of Tables

3.1	T-link weights for pixel m [54].	45
3.2	Advantages and disadvantages of Kmeans	51
3.3	Advantages and disadvantages of Kmedoids	53
4.1	Silhouette validity index values and their interpretation [24] .	62
4.2	Confusion matrix notation	63
5.1	Average runtime: Kmeans versus Kmedoids	72
5.2	Averages and Standard deviations of Intra-cluster distance . .	76
5.3	Averages and Standard deviations of Inter-cluster distance . .	79
5.4	Averages and Standard deviations of Silhouette validity index	82
5.5	Average Precision and recall values: Grabcuts: GMM-Kmeans versus Grabcuts: GMM-Kmedoids	103
5.6	Average runtime: Kmeans versus Kmedoids	109
5.7	Averages and Standard deviations of Intra-cluster distance . .	113
5.8	Averages and Standard deviations of Inter-cluster distance . .	116
5.9	Averages and Standard deviations of Silhouette validity index	119
5.10	Average Precision and recall: Grabcuts: GMM-Kmeans versus Grabcuts: GMM-Kmedoids	140



5.11 Average runtime required by Grabcuts: GMM-Kmeans Scenario 1 versus Scenario 2	153
5.12 Average runtime required by Grabcuts: GMM-Kmedoids Scenario 1 versus Scenario 2	153
A.1 Types of registers and their functions	171
A.2 CPU versus GPU.	176



University of Fort Hare
Together in Excellence

1. INTRODUCTION

1.1 *Background*

Image segmentation is a solution for many computer vision problems. It is the process of partitioning a digital image into small segments such as pixels or sets of pixels. Image segmentation allows for the analysis of images by enabling the visualization of structures of interest and the removal of unnecessary information. In addition, it leads to the extraction of different regions with similar attributes. Image segmentation is applied in many fields which include healthcare for image surgery and construction amongst other fields. It also finds practical applications ranging from filtering of noisy images, this study of anatomical structures, locating objects in satellite images, face recognition, finger print recognition. Thus, image segmentation is considered as the first and significant step in image analysis and it directly influences the overall success in understanding an image. Segmentation of images may be computationally expensive especially when large datasets are used, thus the importance of a fast and efficient segmentation algorithm is realized. Furthermore, a good segmentation is said to be one in which pixels in the same category are allocated to a similar greyscale multivariate value and form a connected region. Also, pixels or sets of pixels in a different category are



University of Fort Hare

Together in Excellence

allocated different values.

1.2 *Problem statement*

We live in the digital era where more often than not information can be found in images or videos. These images contain important information that may be seen to be lost amongst the mist of insignificant aspects of these images. Thus the problem is the extraction of useful information from images. Many users such as graphics artists as well as everyday users find themselves having the need to cut certain objects from other images to compose new ones. Thus, the need to find an algorithm that will assist users is realised. Moreover, many researchers have produced image segmentation algorithms that have catered for different image types. In particular, there has been several graph based image segmentation algorithms developed by researchers such as Boykov et al. [9], etc. According to Han et al. [25], the immense amount of information and the unpredictable complexity in images requires that one develops an efficient and inexpensive segmentation algorithm.

1.3 *Research aims and objectives*

The aim of this study is to provide a comparison of clustering techniques that can be used to generate the Gaussian Mixture Models (GMMs) for the Grabcut-based image segmentation of colour images. Moreover, this study aims at suggesting a clustering technique that will enable the grabcut-

based image segmentation algorithm developed in [48] to be efficient and inexpensive. To do this, the researcher will:

1. Review the literature on graph-based methods used for image segmentation.
2. Provide a summary and survey of the graph-cut algorithms that have been developed thus far.
3. Review the literature and provide summaries of the grabcut algorithms described in [48] and [54].
4. Describe and explain the use of the GPU instead of CPU to run graph-based image segmentation algorithms.
5. Conduct a review of clustering techniques.
6. Identify measures used to determine the quality of clustering solutions.
7. Analyse the performance of the Grabcuts for image segmentation algorithm with a GMM based on Kmeans when the number of clusters is varied from 2 to 5 under scenario 1 and 2.
8. Analyse the performance of the Grabcuts for image segmentation algorithm with a GMM based on Kmedoids when the number of clusters is varied from 2 to 5 under scenario 1 and 2.
9. Compare the performance of the algorithms under scenario 1 and 2.



University of Fort Hare
Together in Excellence

1.4 Contributions of this study

The novel contributions of this study are:

- An adapted Grabcut for image segmentation algorithm with a GMM based on Kmeans and Kmedoids.
- The analysis of Kmeans and Kmedoids clustering techniques under changing conditions.
- A comparison of the performance of the Kmeans and Kmedoids clustering techniques.
- The analysis of the Grabcuts for image segmentation algorithm with GMM based on two clustering techniques under changing conditions.
- A comparison of the performance of the Grabcuts for image segmentation algorithm with a GMM based on the Kmeans and Kmedoids clustering techniques.

1.5 Image segmentation techniques

According to Dass, Priyanka & Devi [18], there is no single method which can be considered good for all types of images. Thus, in this section we will discuss the different image segmentation techniques which are categorized according to the image properties under two headings viz, detecting discontinuities and similarities see Figure 1.1.

Edge based segmentation technique belongs to the image segmentation techniques that detect discontinuities. On the other hand threshold, region,

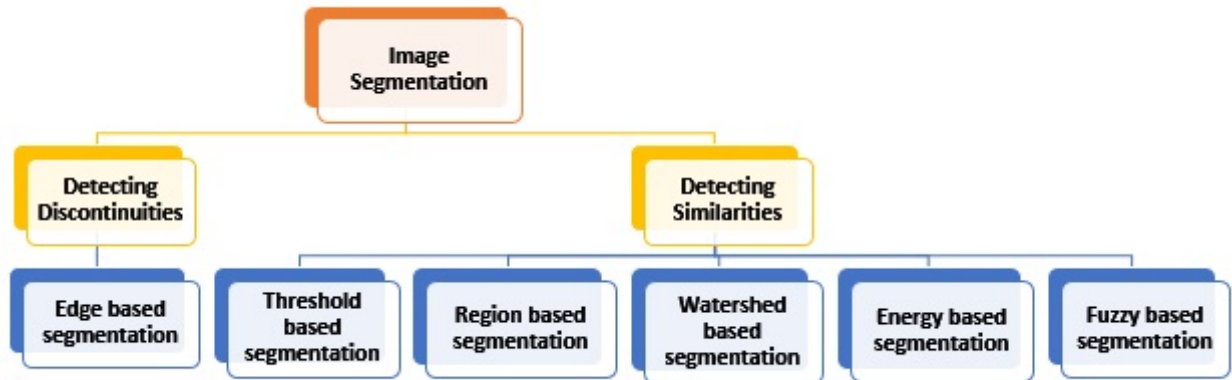


Fig. 1.1: Segmentation techniques

watershed and energy based image segmentation techniques are grouped under techniques that detect similarities to segment images [18].



Detecting discontinuities

University of Fort Hare

In this type of segmentation the partitioning of an image is done based on the abrupt changes in intensity [18]. The following image segmentation techniques are categorised under this category:

1.5.1 Edge based segmentation

This segmentation technique is carried out by detecting the edges or pixels between different regions that have a rapid transition in intensity. These regions are extracted and linked to form closed object boundaries. This property is considered to be the most significant attribute of the edge detection method. In this type of segmentation the values of the pixels connecting foreground and background are assumed to be distinct. According

to Muthukrishnan et al. [39], this technique considers three different types of discontinuities in gray level images namely; points, lines and edges. In addition, edge based segmentation results in a binary image. The following are some of the well known edge based segmentation algorithms; Roberts edge detection, Sobel edge detection, Prewitt edge detection, Robinson edge detection, etc [41,45,50]. Edge based segmentation has two main edge based segmentation methods.

1. Gray histogram method
2. Gradient based method

Detecting similarities

The segmentation of an image under this method is done by grouping similar characteristics of the image according to a set of predefined criterion [18]. There are several image segmentation techniques categorised under this method; including thresholding, region growing, inter alia.

1.5.2 Threshold based segmentation

According to Al-amri et al.[2], threshold based segmentation is a method that is mostly used to discriminate foreground from the background. The method is done by selecting an adequate threshold value T , then a gray level image is converted to a binary image which contains all the significant information of the image. The threshold value is chosen based on the analysis done on the histogram generated by the image. In addition, the gray level values below the value of T will be classified as black and those above T

as white. The following are the five threshold based segmentation techniques:

1. Mean technique

The mean technique uses the value of the mean of the pixels that constitutes the image as the threshold value. This technique is used when approximately half of the pixels of the image belong to the foreground and the remainder of the pixels belong to the background. Although this is the case, it is imprudent to note that this technique is rarely used due to the fact that images in which half of the pixels belong to the foreground with the remaining half belonging to the background are rare.



2. P-Tile technique

According to Al-amri et al. [2], this technique is one of the earliest threshold methods based on gray level histogram. It utilises the size of the area occupied by the object of interest. In addition, this method assumes that the image occupies a fixed percentage of the picture area. This fixed percentage of the picture area is also known as P%. The P-tile technique has a limitation as it assumes that the object is always brighter than the background, which is not always the case.

3. Histogram Dependent technique

4. Edge Maximization technique

5. Visual technique

1.5.3 Automatic threshold

Al-amri et al.[2], maintains that the selection of a threshold value by the system without human intervention is known as automatic thresholding scheme. In this type of threshold scheme, the system needs to have information about the object such as the intensity characteristics, their size, fractions of the image that are occupied by the object and the number of different objects that appear on the image.

1.5.4 Region based segmentation

According to Saini et al.[50], region based segmentation can be further broken down into region growing and region splitting-merging segmentation.



1. Region growing segmentation
2. Region splitting-merging segmentation

1.5.5 Watershed based segmentation

Watershed based segmentation approaches transform an image into a gradient image. This is due to the fact that this approach is a mathematical morphological approach that creates its analogy from a real life flood situations [50]. The approach can be viewed as an edge based method and is said to be susceptible to over segmentation [50].

1.5.6 Energy based segmentation

Energy based segmentation consists of live wire segmentation, active contour, level sets and graph cut based segmentation.

1.5.7 Fuzzy based method

A Fuzzy based method utilise some similarity criterion which includes distance, connectivity or intensity. These approaches include the FCM (Fuzzy C-Means), GK (Gustafson Kessel) and GMD (Gaussian Mixture Decomposition) techniques amongst others [18, 59].

An overview of the remainder of the thesis



This thesis consists of six chapters, in Chapter 1, we give some background and history of image segmentation. We also give definitions of some preliminary concepts and detailed descriptions of some of the prominent image segmentation techniques. In chapter 2, we look at the descriptions of images (i.e. colour images, gray scale images, etc.). In addition, we present some basic terminology related to graph theory. Furthermore, the research visits some of the first graph based algorithms for image segmentation. A detailed description of the Grabcut algorithm proposed by [48, 54] is given in Chapter 3. In addition, the proposed adaptations on the Grabcut algorithm are outlined in this chapter. A detailed discussion of the performance measures used in this study to determine the quality of the clustering solution and segmentation results is given in Chapter 4. Chapter 5 gives the experimental set-up and results as well as discussions obtained from testing the algorithm

proposed. A suitable conclusion for this study and possible future work are alluded to in Chapter 6.



University of Fort Hare
Together in Excellence

2. LITERATURE REVIEW

The graph cuts method has become a popular alternative for solving computer vision problems such as object recognition, tracking and segmentation. Many researchers maintain that the use of graph cuts in image segmentation is motivated by the fact that they allow for geometric interpretation of an image. In addition, graph cuts works as powerful tool for energy minimization for a wide range of binary and non-binary energies that frequently occur in early vision [10]. They also maintain that graph cuts are sometimes used as optimization techniques and that they are closely related to combinatorial algorithms. Graph cuts have been given praises for their ability to convert image segmentation problems into energy minimisation problems, and then finding the global optimal solution with the aid of min cut/ max flow algorithms. In addition, Khokher et al. [30]. have alluded to the fact that graph cuts for image segmentation is the most promising amongst the methods mentioned in Section 1.5, as they promote perceptual grouping and organisation using the image features and spatial information. This is due to some of these methods being effective for some images such as the histogram thresholding that works best for monochrome images and not colour images. In addition some of these methods are biased to ellipsoidal structures and do not necessarily produce closed regions, inter alia edge based methods.

2.1 Digital images and their importance

Digital images have become very dear to our hearts since the advancement of technology. Hence we will examine their basics as well as understand their different types, viz. colour, black and white, and grayscale images. In our attempt to unveil these different types of digital images, we will first examine some basic concepts.

Definition 2.1.1. [49] *A digital image is a rectangular array of pixels sometimes called a bitmap.*

Binary image

As the prefix “bi” may loosely indicate, a binary image is a digital image whose pixels have only two possible values. According to [49], binary images are also called bi-level or two level images. In addition, the names black-and-white, monochrome or monochromatic can be used synonymously with binary image. Although this is the case, these names can be used interchangeably with gray scale images. Conway & Sloane [15], maintains that the binary images can originate from the processing of digital images or as a product of operations such as image segmentation, thresholding and dithering.

2.1.1 Interpretation of binary images

Definition 2.1.2 (n-Dimensional integer lattice [11]). *An n-dimensional integer lattice $L \subseteq \mathbb{R}^n$, is a lattice in the Euclidean space \mathbb{R}^n whose lattice*

points are n -tuples of integers.

According to [49], binary images can be interpreted as subsets of the two-dimensional integer lattice \mathbb{Z}^2 . This lattice is also known as the square or grid lattice and is one of five types of two-dimensional lattices as classified by their symmetry groups. This interpretation leads to the inspiration of a new field called morphological image processing (i.e. theory and technique for analysing and processing geometrical structures, based on set theory, lattice theory, topology, etc.).

Binary images require less memory allocation due to their ability to have small file sizes when stored in packed arrays of bits and bitmaps. For this reason, they are an ideal image format for document management solutions.

Colour image



University of Fort Hare

To understand what a colour image is, we first need to give a description of a colour space. There are several definitions and descriptions given by researchers for a colour space. According to [52], a colour space is a mathematical model that describes the way in which colours can be represented. Sangwine & Horne [52], maintains that a colour space sometimes called colour coordinate systems are three-dimensional groupings of colour sensations. In addition, colour spaces are named by their primary colours. There are several colour spaces that are of interest when it comes to image processing , viz. RGB, CMY(K), HSL, YIQ, YUV, YCbCr, YCC and CIE. We shall now give a brief description of the RGB and CMY(K) colour spaces.

RGB (Red, Green and Blue)

Let us begin by providing a description of the RGB colour space. This colour

space is based on the additive RGB colour model in which the colours red, green and blue are added to produce a broad array of colours. According to Sangwine & Horne [52], this colour space is used more frequently for image processing as many colour cameras, printers, scanner are provided with a direct RGB input or output signals. Moreover, this colour space may be transformed into the other colour spaces. The RGB colour space may be viewed pictorially as a cube since it consists of a three-dimensional coordinate system. Figure 2.1 shows the RGB colour space gamut which represents the RGB colour model where each colour is described by its RGB components.

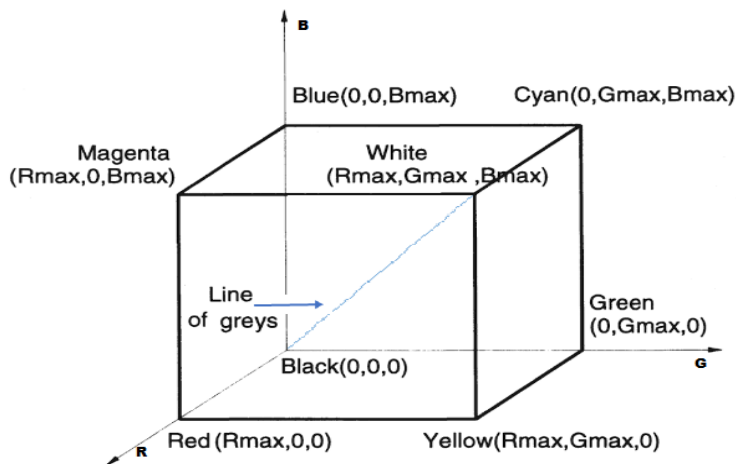


Fig. 2.1: RGB colour space as shown by [52]

The R, G, B axes which are on the edges of the cube represent the three primary colours, viz. red, green and blue. All grey colours are placed on the main diagonal of the cube from black, which is given by $(R = G = B = 0)$, to white $(R, G, B = max)$ where $max = 255$ is the maximum value which any colour can have. Each colour point within the cube is given by weighted

vector sum of the primary colours using the vectors $R, G,$ and B : $C(\lambda) = (r, g, b) = rR + gG + bB$.

CMY(K)

The Cyan, Magenta, Yellow and Key (Black) colour space is said to be the direct opposite of the RGB colour space. This is due to the fact that the color space uses a subtractive colour model where the colours cyan, magenta and yellow subtract brightness from white. According to [52], this colour space is mainly used for colour printing.

Grayscale image

Grayscale images contain brightness information (i.e. intensity) only and each pixel value in such an image corresponds to an amount of light [27]. These images are composed of exclusively shades of gray with black having the weakest intensity to white having the strongest. In addition, grayscale images are different from bi-level images as they contain many shades of gray as opposed to bi-level images that are composed of black and white colours only [49].

2.2 Basic Graph Theory

Definition 2.2.1. [58] A simple graph G is a pair $(V(G), E(G))$, where $V(G)$ is a non-empty finite set of elements called vertices (or vertices), and $E(G)$ is a finite set of unordered pairs of distinct elements of $V(G)$ called edges. A graph H is called a subgraph of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. H is a proper subgraph of G if $E(H) \neq E(G)$.

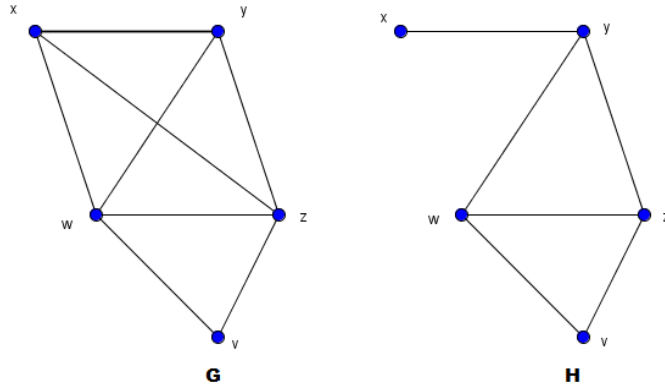


Fig. 2.2: A graph G with its subgraph H

Definition 2.2.2. [16] Let H be a subgraph of G such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. Then $G[V] = (V(H), E(H))$ is the subgraph of G induced by the vertex set such that $E(H) = \{(u, v) \in E(G) : u, v \in V(H)\}$.

Definition 2.2.3. [20, 58] In a graph G , let $v_i \in V(G)$ for $i \in 1, 2, \dots, k + 1$. The sequence $W = v_1, v_2, \dots, v_{k+1}$ is a walk of length k from v_1 to v_{k+1} if v_i and v_{i+1} are adjacent for all $1 \leq i \leq k$. We say that W is closed if $v_1 = v_{k+1}$.

Definition 2.2.4. [20, 58] A path is a non-empty graph $P = (V, E)$ where $V = \{x_0, x_1, \dots, x_k\}$ and $E = \{x_0x_1, x_1x_2, \dots, x_{k-1}x_k\}$ such that the x_i s are distinct. The number of edges in a path is its length. A cycle is a path with an extra edge joining the two end vertices.

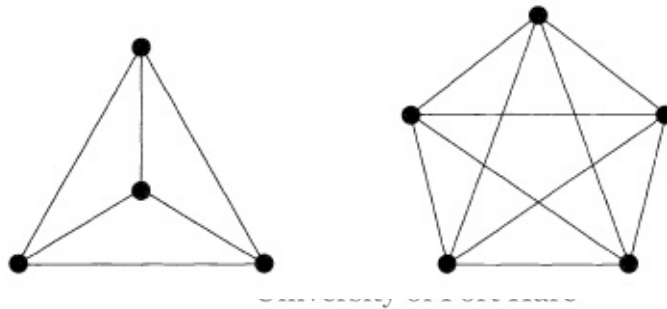
Definition 2.2.5. [20] Let G be a graph. The distance $d(u, v)$ for all $u, v \in V$ is the length of the shortest $u - v$ path in G . If there is no such path, then $d(u, v) := \infty$.

Definition 2.2.6. [20] Let G be a graph. The diameter of G , $\text{diam}(G)$ is the greatest distance between any $u, v \in G$.

Definition 2.2.7. [10] *Image segmentation is the process of partitioning an image into a set of regions that cover it.*

Definition 2.2.8. [16] *An algorithm is a set of steps or instructions followed to achieve a certain goal.*

Definition 2.2.9. [16, 58] *A complete graph is a simple graph in which any two distinct vertices are adjacent. A complete graph is denoted by K_n where n is the number of vertices. The graph K_n has $n(n - 1)/2$ edges.*



Together in Excellence
Fig. 2.3: The graphs K_4 and K_5

Definition 2.2.10. [16, 58] *Let $G = (V, E)$ be a graph. Then G is said to be regular if every $v \in V$ has a degree r , where the degree of a vertex is given by the cardinality of the set of edges incident to v .*

Definition 2.2.11. [51] *A bipartite graph is a graph $G = (V, E)$ in which there exists two disjoint sets $L \cup R = V$ such that each $e \in E$ has one end in L and another end in R .*

Definition 2.2.12. [20] *A non-empty graph G is called connected if any $u, v \in V$ are linked by a path in G . Let $G[U]$ be a graph induced by vertices in $U \subseteq V$. If $G[U]$ is connected, then U is said to be connected in G .*

Proposition 2.2.13. [20] *The vertices of a finite connected graph G can always be enumerated, say as v_1, \dots, v_n , so that $G_i := G[v_1, \dots, v_i]$ is connected for every i .*

Proof. Pick any vertex as v_1 , and assume inductively that v_1, \dots, v_i has been chosen for some $i \leq |G|$. Now pick a vertex $v \in G - G_i$. As G is connected it contains a $v - v_1$ path P . Choose the last vertex v_{i+1} of P in $G - G_i$, then v_{i+1} has a neighbour in $G - G_i$. The connectedness of every G_i follows by induction on i . \square

Definition 2.2.14. [20] *Let $G = (V, E)$ be a graph. A maximal non-empty connected subgraph of G is called a component of G .*

Definition 2.2.15. [58] *Let $G = (V, E)$ be a graph. G is called a null graph if the set of edges is empty. A null graph on n vertices is denoted by N_n .*

Definition 2.2.16. [58] *A finite directed graph or digraph is a pair $D = (V, E)$ where V is a non-empty finite set of vertices and E is a collection of ordered pairs of elements of V called edges.*

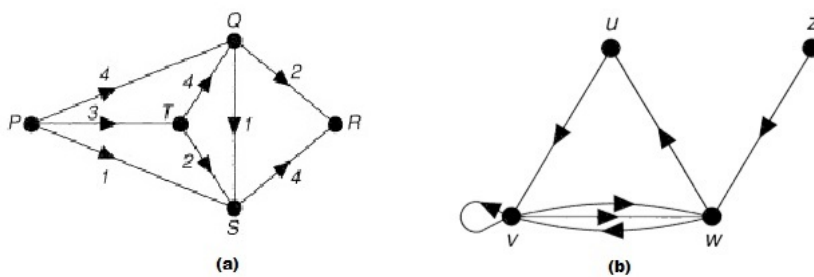


Fig. 2.4: Examples of digraphs

Thus, a digraph is a graph whose edges have a direction which is usually shown by an arrow. In addition, a digraph is said to be simple if all its

edges are distinct and it has no loops [58]. Figure 2.4(a) illustrates a simple directed graph, while Figure 2.4(b) shows a directed graph that is not simple since it has a loop given by (v, v) .

Definition 2.2.17. [58] Let $D = (V, E)$ be a digraph. D is connected if there are no digraphs F and G such that $F \cup G = D$ are digraphs. D is strongly connected if for any $u, v \in V$ there is a path from u to v .

Definition 2.2.18. [58] Let $G = (V, E)$ be a graph. Then G is called a weighted or edge weighted graph, if there is a weight function $\alpha : E \rightarrow \mathbb{R}^+$ on its edges.

Therefore a weighted graph, is a graph G with non-negative real numbers called weights assigned to each edge. Hence a weighted directed graph may be seen as a graph with both direction and weights on its edges.



University of Fort Hare
Together in Excellence

Definition 2.2.19. [20] A graph $G = (V, E)$ is said to be dense if for every $v \in V$, $\text{degree}(v) \geq n/2$, where $n = |V|$. A graph G is said to be sparse if it is not dense.

Definition 2.2.20. [42] A digraph $G = (V, E)$ is said to be cyclic if every non-empty path in G belongs to a cycle. Otherwise, a digraph that does not contain cycles is called acyclic.

Definition 2.2.21. [58] A forest is a graph that contains no cycles, and a connected forest is called a tree.

2.3 Network Flow

Definition 2.3.1. [20] A network N consists of an underlying digraph $D(V, E)$ and two distinct vertices s and t which are called the source and sink of N .

Definition 2.3.2. [20, 53, 58] Let $G = (V, E)$ be a graph. A flow is a real-valued function $f : V \times V \rightarrow \mathbb{R}$ that satisfies the following three properties:
 Capacity constraint: for all $u, v \in V$, $f(u, v) \leq c(u, v)$ where $c(u, v)$ is a non-negative real number assigned to each $(u, v) \in E$;

Skew Symmetry: for all $u, v \in V$, $f(u, v) = -f(v, u)$;

Flow conservation: for all $u \in (V - \{s, t\})$, $\sum_{v \in V} f(u, v) = 0$.



Definition 2.3.3. [20, 58]

University of Fort Hare

Let f be a flow in a Network N . The value of a flow f is defined as $|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$ (the value of flow is the total flow out of the source in the flow network, since there is no flow into the sink s).

Definition 2.3.4. [13, 20] Let $G = (V, E)$ be a graph. An edge $(u, v) \in E$ is saturated if $f(u, v) = c(u, v)$, otherwise (u, v) is unsaturated.

Figure 2.5 shows a network and a possible flow for the network. In Figure 2.5(b) the edges yx and zx are called zero-flows and edges vz , xz , yz , xw and zw are saturated.

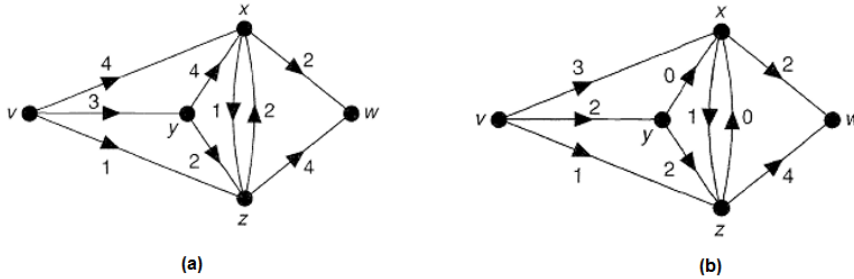


Fig. 2.5: Network and possible flow [58]

2.3.1 Residual Network

Definition 2.3.5. [16] Let f be a flow in G , and consider the pair of vertices $u, v \in V$. The residual capacity

$$c_f(u, v) = \begin{cases} c(u, v) - f(u, v), & \text{if } (u, v) \in E; \\ f(v, u), & \text{if } (v, u) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

If $c_f(u, v) > 0$, then (u, v) is a residual edge.

Definition 2.3.6. [16] Given a flow network $G = (V, E)$ and a flow f , the residual network of G induced by f is $G_f = (V, E_f)$ where

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$$

2.4 Graph cuts

Definition 2.4.1. [32, 45] Let $G = (V, E)$ be a digraph with two terminal vertices s and t . An $s - t$ cut $C = (S, T)$ is a partition of V into two disjoint sets S and T such that $s \in S$ and $t \in T$. If $c(S, T)$ is the cost of the cut, then $c(S, T)$ is the sum of all capacities of the edges that go from S to T given by

$$|c(S, T)| = \sum_{u \in S, v \in T, (u, v) \in E} c(u, v)$$

Definition 2.4.2. [16] Let $G = (v, E)$ be a flow network with source s and sink t , and let f be a preflow in G . A function $d : V \rightarrow \mathbb{N}$ is a height function, also known as the distance function, if;

1. $d(s) = n$, where s is the source vertex and n is the number of vertices in the graph,
2. $d(t) = 0$, where t is the sink vertex,
3. $d(u) \leq d(v) + 1$ for every residual edge (u, v) .

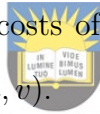


University of Fort Hare
Together in Excellence

Graph cuts have been instrumental in solving many computer vision problems such as image restoration, image segmentation, object recognition, tracking and analysis. This method was first proposed by Boykov and Jolly in their work entitled “*Interactive graph cuts for optimal boundary and region segmentation of objects in N-D Images*” in 2001 [9].

In this method an image is seen as a graph $G = (V, E)$ where labels are assigned to the different pixels or sets of pixels. Pixels that share similar characteristics are assigned to the same label. These characteristics include

colour, intensity and texture of the image. The pixels of the image are seen as the set of vertices V and E is the set of edges which are the distances between the vertices $u, v \in V$. The vertices are grouped under two categories namely; the neighbouring vertices which correspond to the pixels and the terminal vertices which correspond to the source and sink vertices. The source vertex s usually represents the object and the sink vertex t represents the background. The neighbouring vertices are connected by n-links, while the terminal vertices are connected to the neighbouring vertices by t-links. Therefore the edges connecting the neighbouring vertices are called n-links, whereas t-links are edges that connect the neighbouring vertices to the terminal vertices. The edges are assigned with non-negative real numbers called weights, these are also referred to as costs of the edges. The weight of an edge (u, v) is denoted by $w(u, v) = c(u, v)$.



University of Fort Hare
Together in Excellence

The graph cut method seeks to find the minimal cut from the source vertex s to the sink vertex t , which is a cut with the minimum cost value. The minimum cut known as min-cut can be found by finding the maximum flow, since min-cut \equiv max-flow as verified in [9, 20, 58]. The label of the object of interest is set to 1 while that of the background is set to 0. Thus, image segmentation is achieved by minimizing the energy function through graph cuts.

Definition 2.4.3. [9, 32] Let $L = \{l_1, l_2, \dots, l_p\}$ be a binary vector whose components l_i specify assignments to pixels $i \in P$. Each l_i can either belong to the background or foreground. Then, the energy function is given by;

$$E(L) = \alpha R(L) + B(L)$$

where $R(L) = \sum_{i \in P} R_i(l_i)$ is the regional term which incorporates the regional information into the segmentation. $B(L)$ is the boundary term which incorporates the boundary constraint into the segmentation and α is the relative importance factor between the regional and boundary term.

Thus L consists of two parts: the object labelled as 1 and the background labelled as 0. In addition, $R_i(L_i)$ is the penalty for assigning the label l_i to the pixel i , whose weight can be obtained by comparing the intensity of the pixel p with the given histogram (intensity model) of the object and background. Furthermore, when $\alpha = 0$ the regional information is ignored and only the boundary information is considered .

2.5 Graph cut based algorithms



University of Fort Hare
Together in Excellence

According to Collins [14], graph cut algorithms can be classified under two categories namely; Ford-Fulkerson and Push-relabel algorithm.

Theorem 2.5.1 (Integrity theorem [58]). *The maximum number of edge-disjoint paths from a vertex v to a vertex w in a digraph D is equal to the minimum number of edges in a vw -disconnecting set.*

Theorem 2.5.2 (Max-flow min-cut theorem [14,20,58]). *In any network N , the value of any maximum flow is equal to the capacity of any minimum cut.*

Proof. Suppose the capacity of every edge is an integer. In this case, the network can be regarded as a digraph \tilde{D} in which the capacities represent the number of edges connecting to the various vertices. The value of a maximum

flow then corresponds to the total number of edge-disjoint paths from v to w in \tilde{D} , and the capacity of a minimum cut refers to the minimum number of edges in a vw -disconnecting set of \tilde{D} . The result follows from the integrality theorem. The extension of this proof is the consideration of networks with rational numbers as capacities. These capacities are multiplied by a suitable integer d to make them integers. Then have the case described above and the result follows on by dividing by d . Finally, if some capacities are irrational, then we approximate them as closely as we please by rationals and use the above result. By choosing these rationals carefully, we can ensure that the value of any maximum flow and the capacity of any minimum cut are altered by an amount that is as small as we wish. Note that, in practical examples, irrational capacities rarely occur since the capacities are usually given in decimal form. □



University of Fort Hare
Together in Excellence

2.5.1 Minimum cut/ Maximum flow algorithms

The minimum cut problem is described as finding a cut that has the minimum cost amongst all possible cuts that a network can have. This problem has been solved by finding the maximum flow from the source vertex s to the sink vertex t . This is due to the theorem developed by Ford and Fulkerson (Theorem 2.5.2). According to Boykov and Veksler [10], there are many standard polynomial time algorithms that are developed to solve the aforementioned problem. These algorithms are categorised under two significant groups, viz. Push-relabel and augmenting paths methods [10, 14]. We now discuss the Push-relabel and Ford-Fulkerson algorithms.

2.5.2 Push-relabel algorithm

The Push-relabel algorithm, also known as the preflow-push algorithm, was primitively designed by Alexander V. Karzanov who published it in the Soviet Mathematical Dokladi 15 in the year 1974. According to Cormen et al. [16], this algorithm had three basic operations the preflow, push as well as the relabel operations. Although this was the case, this method utilises distances instead of a labelling system in the auxiliary network to determine where to push the flow. Chandran and Hochbaum [13], maintains that this algorithm was later developed by Andrew Goldberg and Robert Tarjan. It was then published as an article in the Journal of the ACM in 1988.

Definition 2.5.3 (Preflow [16]). *A preflow in a network $G = (V, E, s, t, c, e)$ is a function $f : E \rightarrow \mathbb{Z}$ satisfying the following constraints*

- For each edge (u, v) , $f(u, v) \leq c(u, v)$.
- For each vertex $v \in V - t$,

$$\sum_u f(u, v) - \sum_w f(v, w) \geq 0.$$

Definition 2.5.4. [16] *The excess flow $e(v) = \sum_{(u,v) \in E} f(u, v) - \sum_{(v,w) \in E} f(v, w)$ is the net flow into v . The vertex $u \in V - \{s, t\}$ is overflowing or active if $e(u) > 0$.*

2.5.3 Operations of the algorithm

The push-relabel algorithm has three basic steps. These include the initialisation of preflow, the push and relabel operations.

1. Initilise Preflow() operation

This operation initialises the height and flows of all vertices. The following are the initialization steps;

- (a) Initialise height and flow of every vertex as 0
- (b) Initialise height of source vertex equal to the total number of vertices in the graph
- (c) Initialise flow of every edge as 0
- (d) For all vertices adjacent to the source s , flow and excess flow is equal to initial capacity

2. Push() operation

Definition 2.5.5. [13, 16] An edge $(u, v) \in E_f$ is admissible if $d(u) = d(v) + 1$ where $d(u)$ and $d(v)$ are the distances from u and v to t in G . E_f is the set of edges in the residual graph.

This operation is used to push flow from an overflowing vertex to a vertex that can take more flow. Thus if the vertex u has excess flow $c_f(u, v) > 0$ and $d(u) > d(v)$, then flow can be pushed into the residual edge (u, v) . The amount of flow pushed through a residual edge is given by $\min(e(u), c_f(u, v))$.

3. Relabel() operation

The relabel operation raises the height of an overflowing vertex that has no adjacent vertices which are at a lower height. That is, the $\text{relabel}(u)$ applies if $c_f > 0 \forall w$ such that $(u, w) \in E_f, d(u) \leq d(w)$. The height $d(u)$ is increased by 1 so that the $\text{push}()$ operation is possible.


Lemma 2.5.6. [16]

At every step, if a vertex v has positive excess flow, then there is a path from v to s in the residual network.

Proof. [16] For an overflowing vertex x of G , let $U = \{v : \text{there exists a simple path from } x \text{ to } v \text{ in } G_f\}$, and suppose for the sake of contradiction that $s \notin U$. Let $W = V - U$. We use the definition of the excess flow, sum over all vertices in U , and note that $V = U \cup W$, to obtain

$$\sum_{u \in U} e(u) = \sum_{u \in U} \left(\sum_{v \in V} f(v, u) - \sum_{v \in V} f(u, v) \right)$$

Because $V = U \cup W$ and $U \cap W = \emptyset$, then the summation over V can be split into summation over U and W to obtain;



University of Fort Hare
Together in Excellence

$$\begin{aligned} \sum_{u \in U} e(u) &= \sum_{u \in U} \left(\left(\sum_{v \in U} f(v, u) + \sum_{v \in W} f(v, u) \right) - \left(\sum_{v \in U} f(u, v) + \sum_{v \in W} f(u, v) \right) \right) \\ &= \sum_{u \in U} \sum_{v \in U} f(v, u) + \sum_{u \in U} \sum_{v \in W} f(v, u) - \sum_{u \in U} \sum_{v \in U} f(u, v) - \sum_{u \in U} \sum_{v \in W} f(u, v) \\ &= \sum_{u \in U} \sum_{v \in W} f(v, u) - \sum_{u \in U} \sum_{v \in W} f(u, v) + \left(\sum_{u \in U} \sum_{v \in U} f(v, u) - \sum_{u \in U} \sum_{v \in U} f(u, v) \right) \end{aligned}$$

The two summations within the parentheses are the same, since for all $x, y \in V$ the term $f(x, y)$ appears once in each summation. Hence, these summations cancel, we have;

$$\sum_{u \in U} e(u) = \sum_{u \in U} \sum_{v \in W} f(v, u) - \sum_{u \in U} \sum_{v \in W} f(u, v) \quad (2.1)$$

We know that the quantity $\sum_{u \in U} e(u)$ must be positive because $e(x) > 0, x \in U$, all vertices other than s have a non-negative excess, and, by assumption, $s \notin U$. Thus, we have

$$\sum_{u \in U} \sum_{v \in W} f(v, u) - \sum_{u \in U} \sum_{v \in W} f(u, v) > 0. \quad (2.2)$$

By skew symmetry we have $f(u, v) = -f(v, u)$, thus (2.2) gives

$$2 \sum_{u \in U} \sum_{v \in W} f(v, u) > 0 \Rightarrow \sum_{u \in U} \sum_{v \in W} f(v, u) > 0.$$

Hence, there must exist a pair of vertices $u' \in U$ and $v' \in W$ with $f(v', u') > 0$. The edge $(u', v') \in E_f$, which means that there is a simple path from x to v' (the path $x \rightsquigarrow u' \rightarrow v'$), thus contradicting the definition of U . \square

Lemma 2.5.7. [16] *At every step, if there is an edge (u, v) that has positive capacity $c(u, v) > 0$ in the residual network, then $h(u) \leq h(v) + 1$.*

2.5.4 Ford-Fulkerson algorithm

The Ford-Fulkerson algorithm, which in this study is called the Ford-Fulkerson method, depends on three significant ideas, namely; residual networks, augmenting paths and cuts[16]. It is referred to as a method rather than an algorithm because the approach of finding the augmenting paths in the residual network is not specified, and hence there are several implementations with different running times [16]. Since the concept of residual networks and cuts are already defined previously, we shall now define an augmenting path.

Definition 2.5.8 (Augmenting path [16,58]). *Let $G = (V, E)$ be a graph and*

f a flow. An augmenting path p is a simple path from s to t in the residual network. Augmenting paths consist of entirely unsaturated edges that carry non-zero flow.

Thus any path p with available capacity may be considered an augmenting path.

Idea behind the Ford-Fulkerson method:

1. There is path from the source s to the sink t , with available capacity on all edges in the path.
2. We send flow along the path.
3. Then we find another path and send flow there.



The Ford-Fulkerson algorithm is a greedy algorithm that was first published in 1956. The algorithm was further improved to be the Edmonds-Karp algorithm. The analysis of this algorithm varies depending on the technique used to find the augmenting paths. Typically this algorithm runs in polynomial time if the Breath-first search method is used[16]. The runtime for this algorithm is said to be $O(E \cdot |f^*|)$ where f^* is the maximum flow [16].

Theorem 2.5.9. [16] Let $G = (V, E)$ be a flow network, f be a flow in G , and P be an augmenting path in G_f . Let

$$f_P(u, v) = \begin{cases} c_f(P), & \text{if } (u, v) \text{ is on } P; \\ 0, & \text{otherwise.} \end{cases} \quad (2.3)$$

and suppose that we augment f by f_P . Then the function $f \uparrow f_P$ is a flow

in G with value $|f \uparrow f_P| = |f| + |f_P| > |f|$, where $f \uparrow f_P$ implies the augmentation of the flow f by f_P .

Lemma 2.5.10. [16] *Let f be a flow in a flow network G with source s and sink t , and let (S, T) be any cut of G . Then the net flow across (S, T) is $f(S, T) = |f|$.*

Corollary 2.5.11. [16] *The value of any flow f in a flow network G is bounded above by the capacity of any cut of G .*

Proof. [16] Let $c(S, T)$ be any cut of G and let f be any flow. By Lemma 2.5.11 and the capacity constraint,

$$\begin{aligned}
 |f| &= f(S, T) \\
 &= \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u) \\
 &\leq \sum_{u \in S} \sum_{v \in T} f(u, v) \\
 &\leq \sum_{u \in S} \sum_{v \in T} c(u, v) \\
 &= c(S, T).
 \end{aligned}$$

□

Definition 2.5.12. [16] *Let f be a flow network in G with source s and sink t , and let (S, T) be any cut of G . Then the net flow across (S, T) is $f(S, T) = |f|$.*

Theorem 2.5.13. (Optimality conditions for maximum flow [16])

Let f be a flow in a graph G . The following are equivalent.

1. f is a maximum flow of G ;
2. There is no $s - t$ path (with positive residual capacity) in the residual network G_f .
3. There is an (s, t) -cut (A, B) such that the value of f equals the capacity of (A, B) ;

Proof. [16]

(1) \implies (2): suppose for the sake of contradiction that f is a maximum flow in G , but that G_f has an augmenting path p . Then by Theorem 2.5.10, the flow found by augmenting f by f_p , where f_p is given by equation(2.2), is a flow in G with value strictly greater than $|f|$, contradicting the assumption that f is a maximum flow in G .



(2) \implies (3) : Suppose that G_f has no augmenting path, that is, G_f contains no path from s to t . Define $S = \{v \in V : \text{there exist a path from } s \text{ to } v \text{ in } G_f\}$ and $T = V - S$. The partition (S, T) is a cut: we have $s \in S$ trivially and $t \notin S$ because there is no path from s to t in G_f . Now consider a pair of vertices, $u \in S$ and $v \in T$. If $(u, v) \in E$, we must have $f(u, v) = c(u, v)$, since otherwise $(u, v) \in E_f$ which would place v in the set S . If $(v, u) \in E$ we must have $f(v, u) = 0$, because otherwise $c_f(u, v) = f(v, u)$ would be positive and we would have $(u, v) \in E_f$, which would place v in S . Of course, if neither

(u, v) nor (v, u) is in E , then $f(u, v) = f(v, u) = 0$. We thus have

$$\begin{aligned}
 f(S, T) &= \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{v \in T} \sum_{u \in S} f(v, u) \\
 &= \sum_{u \in S} \sum_{v \in T} c(u, v) - \sum_{v \in T} \sum_{u \in S} 0 \\
 &= c(S, T).
 \end{aligned} \tag{2.4}$$

Therefore, $|f| = f(S, T) = c(S, T)$.

(3) \implies (1) : By the definition of net flow across a cut (S, T) , $|f| \leq c(S, T)$ for all cuts (S, T) . The condition $|f| = c(S, T)$ thus implies that f is a maximum flow.



□

University of Fort Hare
Together in Excellence

2.5.5 Push-Relabel versus Ford-Fulkerson algorithm: Similarities and differences

The Push-relabel algorithm has seen many applications as it is proven to be the more efficient algorithm in terms of time complexity as compared to its counterpart the Ford-Fulkerson algorithm [13, 34]. However, these two graph based algorithms share a similar underlying concept. The push-relabel and Ford-Fulkerson algorithms both work on residual graphs. Although this is the case, there are certain differences between the two algorithms. According to [40], the runtime of the Push-relabel algorithm depends on the number of possible relabel operations, saturating paths and non-saturating pushes. On

the other hand the runtime of the Ford-Fulkerson algorithm depends on the order of choosing the augmenting paths. Figure 2.6 shows the differences of the push-relabel algorithm as compared to the Ford-Fulkerson algorithm.

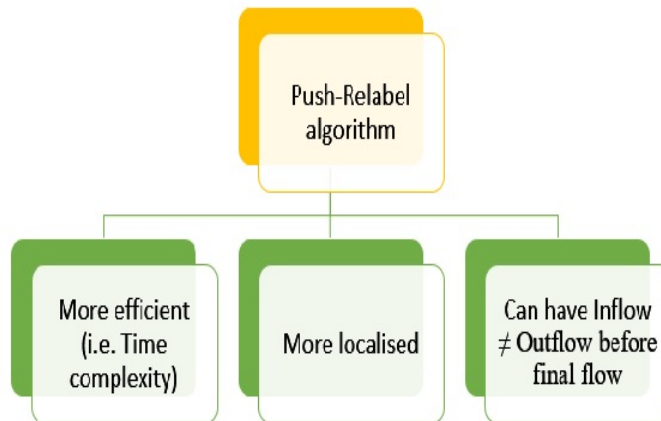


Fig. 2.6: The Push-relabel algorithm as opposed to the Ford-Fulkerson algorithm [34].



University of Fort Hare
Together in Excellence

According to Lalwani [34], the push-relabel algorithm works in a more locally manner as it considers one vertex at a time rather than looking at the whole residual network like the Ford-Fulkerson algorithm. In addition, the push-relabel algorithm permits inflow to exceed outflow before the final flow is reached. When the final flow is reached the net difference, outflow - inflow = 0, except for the source and sink vertex. While this is the case, the Ford-Fulkerson algorithm maintains a net difference of 0 for all vertices except for the source and sink vertex at all times. Furthermore, the push-relabel algorithm is more efficient than the Ford-Fulkerson algorithm [34, 47].

2.6 *Advantages of Graph based segmentation*

Graph based segmentation methods have been said to be one of the most successful techniques in performing image segmentation [45]. This is due to several benefits that these methods possess. The benefits include;

- No discretization by virtue of purely combinatorial operators being required. Thus discretization errors are less likely, etc.

2.7 *Delimitation of graph cut based segmentation*

Although this is the case, graph cuts for image segmentation have had some criticism. They include metrication artifacts, shrinking bias, multiple labels and memory.



University of Fort Hare
Together in Excellence

3. GRABCUTS: THEORY

3.1 Energy minimisation framework

As we have seen in the previous chapter, graph based segmentation algorithms aim to minimise a certain energy function, E , thus we need to look at the different energy functions. According to [14, 53], there are two main models used to define energy function and one is chosen depending on the characteristics of the labelled regions. The models are described below;



University of Fort Hare

Together in Excellence

- Potts Interaction Energy Model

This energy model is given by the following equation

$$E(I) = \sum_{p \in P} |I_p - I_p^0| + \sum_{(p,q) \in N} K_{(p,q)} T(I_p \neq I_q).$$

where the unknown true labels over the set of pixels P and the observed labels corrupted by noise are given by $I = \{I_p | p \in P\}$ and $I^0 = \{I_p^0 | p \in P\}$, respectively [53]. The operator $|I_p - I_p^0|$ gives the absolute value of the difference between the unknown true labels and the labels that are observed to be corrupted by noise over a set of pixels P . Hence it is used to determine the number of correct classifications of pixels. The penalties for label discontinuities between adjacent vertices or pixels

also referred to as Potts interactions are given by $K(p, q)$. T is an indicator function. The Potts interaction energy model is particularly useful when the labels are piecewise constant with discontinuities at the boundaries. This energy can be optimally solved for binary labelling using maximum flow, nevertheless the multiple label case is NP-Hard [14, 53].

- Linear Interaction Energy Model

$$E(I) = \sum_{p \in P} |I_p - I_p^0| + \sum_{(p,q) \in N} A_{(p,q)} T(I_p \neq I_q).$$

The fundamental distinction between the Potts and the Linear interaction energy model is the introduction of constants $A(p, q)$, which stores the importance of interactions between neighbouring pixels p and q [53]. The linear interaction energy produces labellings which are piecewise smooth, but with discontinuities at the boundaries [14].

There are many methods that have been used to segment images as mentioned in Chapter 2. In this study the grabcut method was used.

3.2 Grabcuts for image segmentation

Definition 3.2.1. A Gaussian mixture model is a weighted sum of K component Gaussian densities as given by the equation:

$$p(\vec{x}) = \sum_{i=1}^K w_i N(\vec{x} | \vec{\mu}_i, \Sigma_i)$$

,

where

$$N(\vec{x}|\vec{\mu}_i, \sum_i) = \frac{1}{\sqrt{(2\pi)^K |\sum_i|}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu}_i)^T \sum_i^{-1} (\vec{x} - \vec{\mu}_i)\right) \text{ and } \sum_{i=1}^K w_i = 1$$

\vec{x} is a D -dimensional continuous valued data vector (i.e. measurement or features), w_i are the component weights for $i = 1, \dots, K$, μ_i are the means and \sum_i are the covariance matrices.

Thus, the Gaussian mixture model(GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities. In this segmentation technique the GMM is used to give the statistics of the image. The algorithm then uses the min cut/max flow to arrive at a segmentation of the image. Border matting and some user defined editing is then applied in order to ensure that the segmented image is accurate [48].

Description 3.2.2. [52] A trimap is a pre-segmented image consisting of three regions namely; the foreground, background and unknown regions.

The Grabcuts approach was introduced by Rother, Kolmogorov and Blake in their work entitled “Grab-cut: Foreground extraction using iterated graph cuts.” published in 2004 [48]. According to Ramirez et al [47], the grabcut approach is an iterative and minimal user interaction algorithm as it chooses a segmentation by iteratively revising the foreground and background pixels assignments. The method uses min-cut/max-flow algorithm to segment digital images[47]. The input of this approach is a digital image with a selected region of interest(ROI). The ROI is selected using a rectangular bounding



Fig. 3.1: An image with selected ROI

box. A trimap $T = \{T_B, T_U, T_F\}$ is used to help with the segmentation. The sets T_B , T_U and T_F store all background, unknown and foreground pixels, respectively. The pixels outside the bounding box are confidently assigned to the background, while the ones inside the bounding box are seen as unknown.

The grabcut algorithm then uses these assignments to generate the Gaussian mixture model (GMM) for the foreground as well as the background. The background GMM is created by T_B which has the value $\alpha_n = 0$, where α_n is the opacity value at each pixel and n is the index of that pixel. In addition, a distance measure between each pixel and the foreground and background models is defined based on the component more similar to the pixel in each model [3]. Furthermore, the colour distances between a pixel and its neighbours is calculated. The two distance measures are then used to model the image as a directed weighted graph.

Grabcuts algorithm summary

The grabcut algorithm works with images in the RGB colour space and uses GMMs instead of histograms of pixels as in [9]. This is due to the practically inadequate construction of colour space histograms [48]. We will now give a summary of the grabcut algorithm below as described by [48, 54].

1. The user draws a rectangular bounding box enclosing the region of interest. Thus creating an initial trimap by specifying the background pixels which are situated outside the bounding box. The pixels inside the box are marked as unknown, while those for the foreground are set to the empty set.
2. Hard segmentation is performed by placing the set of unknown pixels into the foreground set and the background pixels are placed into the background set.
3. GMMs with five Gaussian components are created based on the background and foreground sets.
4. Each pixel in the foreground set is assigned to the most likely Gaussian component in the foreground GMM. Similarly, each pixel in the background is assigned to a Gaussian component in the background GMM.
5. A foreground and background GMM cluster is assigned to every pixel based on the minimum distance to the respective clusters.
6. The learning of GMM parameters is conducted based on the pixel data with the newly assigned foreground and background clusters. Thus the



University of Fort Hare
Together in Excellence

old GMMs are discarded and new GMMs are created with the newly assigned foreground and background clusters from every pixel.

7. A graph is constructed and Graph cuts are run to estimate the new foreground and background pixels.
8. The steps 4-7 are repeated until the label of foreground and background pixels converges.
9. Border matting is applied.
10. Editing is performed which entails fixing pixels to be part of the correct set of pixels and updating trimap T accordingly. Step 7 is allowed to run again only once.



According to Rother et al.[48], the grabcut algorithm has three main stages. These are the initialisation, iterative minimisation and user editing stages. We will now give a description of each stage.

3.2.1 Initialisation

The initialisation stage includes steps 1-3 in the grabcut algorithm summary above. During this stage, a trimap $T = \{T_B, T_U, T_F\}$ is used to help with the segmentation. The sets T_B , T_U and T_F store all background, unknown and foreground pixels, respectively. The user marks the region with the object of interest with a bounding box. Then T_B is initialised with all the pixels outside the bounding box. The unknown pixels (pixels inside the bounding box) are set to be $T_U = \overline{T_B}$ where $\overline{T_B}$ is the complement of T_B and $T_F = \emptyset$. The mattes for T_B and T_U are set to matte-background and

matte-foreground, respectively. According to Rother et al.[48], this distinction between the trimap and matte formalises the separation between the user input and the segmentation derived by the grabcut algorithm. Given the mattes the K components of the GMM are created for matte-foreground and matte-background regions. Thus a total of $2K$ GMM components is created. The background and foreground regions are divided into K pixel clusters. The Gaussian components are then initialised from the colours in each cluster. For a given GMM component k , in the foreground for instance, the subset of pixels $F(k) = \{z_n : k_n = k \text{ and } \alpha_n = 1\}$ where z_n are the pixels in the image, α_n is the opacity value at each pixel and n is the index. The mean $\mu(\alpha, k)$ and the covariance $\Sigma(\alpha, k)$ of pixel values in $F(k)$ are estimated as the sample mean and covariance of pixel values and weights are given by $\pi(\alpha, k) = |F(k)| / \sum_k |F(k)|$ [48]. According to Talbot et al. [54], a good separation between foreground and background occurs when a low variance Gaussian component is generated. Thus, the need for tight, well-separated clusters is realised by the grabcut algorithm.

3.2.2 Iterative minimisation

This stage consists of the steps 4-9 in the grabcut algorithm summary. The steps 4-7 in the grabcut algorithm summary can be shown to be a minimisation of the Gibbs energy [48] given by

$$E(\alpha, k, \theta, z) = U(\alpha, k, \theta, z) + V(\alpha, z)$$


with respect to the three sets of variables k, θ, α in turn. U is the data term defined by

$$U(\alpha, k, \theta, z) = \sum D(\alpha_n, k_n, \theta, z_n)$$

where

$$D(\alpha_n, k_n, \theta, z_n) = -\log \pi(\alpha_n, k_n) + \frac{1}{2} \log \det \Sigma(\alpha_n, k_n) + \frac{1}{2} [z_n - \mu(\alpha_n, k_n)]^T \Sigma(\alpha_n, k_n)^{-1} [z_n - \mu(\alpha_n, k_n)]$$

is a weighting function and $\theta = \{\pi(\alpha, k), \mu(\alpha, k), \Sigma(\alpha, k), \alpha = 0, 1$ and $k = 1, \dots, K\}$ are the GMM parameters[48]. In the grabcut implementation of Talbot et al. [54], the weighting function was corrected to be computed as

$$D(m) = -\log \sum_{i=1}^K \pi_i \frac{1}{\sqrt{\det \Sigma_i}} e^{-\frac{1}{2} [z_m - \mu_i]^T \Sigma_i^{-1} [z_m - \mu_i]} \quad (3.1)$$


University of Fort Hare
Together in Excellence

The smoothness term $V(\alpha, z)$ is given by;

$$V(\alpha, z) = \gamma \sum_{(m,n) \in C} d(m, n)^{-1} [\alpha_n \neq \alpha_m] - \beta (z_n - z_m)^2$$

where $d(\cdot)$ is the Euclidean distance, $\gamma = 50$ and $\beta = (2(z_n - z_m)^2)$ are constants. During this stage some pixels are moved from the matte-foreground set to the matte-background set, and vice versa. All pixels in T_U are assigned to a cluster based on the minimum unary weighting function $D(m)$. Thus pixels in the matte-foreground are assigned to the foreground GMM component with the highest likelihood of producing the pixels colour. A similar approach is taken for pixels in the matte-background set, where pix-

els are assigned to the highest likely background GMM component. This process refines the GMM distribution to better define the foreground and background pixels. The learning step takes the newly assigned k values and recalculates both GMMs. The old GMMs are discarded and the new GMMs are used to calculate unary weights. A graph is built for the Graph cut step of the grabcut algorithm. The graph consists of vertices that represent all the pixels in the image and two terminal vertices, viz., the source and sink vertex. The source vertex represents the foreground vertex, while the sink vertex represents the background vertex. Each pixel is connected to eight neighbouring pixels as well as to the source and sink vertex by two types of edges also called links. The N-link describe the penalty for placing a segmentation boundary between the neighbouring pixels and they connect pixels in the 8-neighbourhood. According to Talbot et al. [54], N-links can be reused since the N-links weights are always constant throughout the execution of the grabcut algorithm. The T-links connect the pixels to the terminal vertices(i.e the source and sink vertices) and they describe the probability that each pixel belongs to the foreground or to the background. These probabilities change as the GMMs are updated due to the fact that the updated GMMs change the trimaps thus leading to a change in the T-links. Hence the T-link weights are updated during each iteration. The N-link weight between pixel m and a pixel in its 8-neighbourhood, p , is:

$$N(m, p) = \frac{\gamma}{d(m, p)} e^{-\beta \|z_m - z_p\|^2}, \quad (3.2)$$

where z_m is the colour of pixel m . Rother et al. [48], set $\gamma = 50$ and $\beta = \frac{1}{2(\|z_m - z_p\|)}$ according to [9]. Talbot et al. [54] maintains that there are two T-links for each pixel, the background and foreground T-link that connect a pixel to the background and foreground vertices, respectively. The weights of the T-links depend on whether the user marked a particular pixel as a definite background or foreground pixel. This is reflected by weighting the links such that the pixel is forced into the appropriate set. The weights of these links are given by the labeling costs, $L(m)$, for labeling a pixel m as foreground or background. The probabilities acquired from the GMMs are used to set the weights for the unknown pixels.

Tab. 3.1: T-link weights for pixel m [54].

Pixel type	Background T-link	Foreground T-link
$m \in T_B$	$L(m)$	0
$m \in T_F$	0	$L(m)$
$m \in T_U$	$D_{fore}(m)$	$D_{back}(m)$

Table 3.1 shows the T-link weights for pixel m , where $L(m) > \sum_p N(m, p)$ is chosen to force m to be a member of either the foreground or background. The sum is over all pixels neighbouring m . When Equation (3.2) is maximised $\gamma \geq N(m, p)$ is observed [54]. Thus for an 8-neighbourhood $L(m) = 8\gamma + 1$. $D_{fore}(m)$ and $D_{back}(m)$ are computed using the Equation (3.1), where the summation is over the foreground Gaussian components for D_{fore} and background Gaussian components for D_{back} . Once the graph structure has been initialised with all the weights, the min-cut algorithm by Boykov and Jolly [9] is performed on the graph to estimate segmentation.

3.2.3 User editing

According to Rother et al. [48], the initial incomplete user labelling is often sufficient to allow the entire segmentation to be completed automatically, but this is not always the case. In cases where additional user input is required to complete the segmentation, foreground and background pixel brushes are used to constrain pixels as firm foreground or background. Then the minimisation step 6 in the grabcut algorithm summary above is run once more. In this study we shall limit the analysis and comparison of the Grabcuts for image segmentation algorithm to initial segmentation, even though the function of the user editing will be included in the algorithm.

3.3 Grabcuts based algorithms



The grabcut technique for image segmentation has become popular over the years to many researchers in various fields. Since its introduction by Rother et al. in 2004, there are several algorithms developed. In this section we will review some of these grabcuts based algorithms. The algorithm introduced a technique that made it possible for the segmentation of colour images as opposed to the Graph cuts algorithms that segmented grayscale images (i.e. graph cuts based algorithm developed by BoyKov & Jolly)[48]. Although this was the case, the algorithm had some challenges which include difficulty segmenting noisy images and camouflaged images. Moreover, the original grabcut algorithm had a difficulty in understanding the intentions of the user in different scenarios [59]. This inspired the development of several algorithms that used the grabcut based image segmentation method. These

algorithms solved problems ranging from, but not limited to image segmentation for face & clothing recognition to the reduction of noise in images [28,35]. Ramirez et al. [47], introduced some changes to the original grabcut algorithm. The changes enabled the new algorithm to segment 3D images and it was allowed to run on both the CPU and GPU. The results of this algorithm were impressive, but the algorithm faced similar challenges when it came to the colour distribution of the foreground and background of an image as the original grabcut algorithm [47]. This algorithm led to the deduction that the GPU is superior to the CPU in terms of execution times of the algorithm. In addition to the observations made by the researchers mentioned above, it has also been observed that the grabcut algorithm outperforms the graph cut algorithm in terms of runtime speed and segmentation accuracy [48]. In 2005, a graph cut based algorithm was introduced by Lombaert et al[36]. This algorithm adopted the use of multilevel banded graph cuts to segment images. This algorithm was inspired by the multilevel graph partition method in the work of Karypis and Kumar [29] and the narrow banded algorithm in the level sets method in the work of Adalsteinsson & Sethian [1]. This algorithm was proven to be fast, memory and time efficient as compared with the conventional graph cuts algorithm of Boykov and Jolly [36].

3.4 Advantages of using Grabcuts method

The grabcut approach extends graph cuts to colour images and incomplete trimaps [25]. Due to the iterative minimisation characteristics of the grabcut method, it can substantially reduce the amount of user interaction to

complete the segmentation task [48].

3.5 *Disadvantages of using Grabcuts method*

The grabcut method has many advantages as mentioned above. Although this is the case, [46] maintains that this method lacks the ability to completely segment an image by itself and that it further requires the user to select the foreground and background seeds. In addition, the grabcut method is only meant for colour images [3].

3.6 *Clustering Techniques*

According to Gandhi and Srivastava (2014), clustering is the process of finding meaningful patterns in a set of data that is of interest in order to partition that data into sub classes called clusters. Clustering is known as a form of unsupervised learning as it has no predefined classes [23]. There are a variety of clustering techniques categorised under the following sub headings;

- Partitioning Techniques
- Hierarchical Techniques
- Grid-based Techniques
- Density-based Techniques
- Model-Based Techniques

Partitioning techniques

Partitioning techniques divide the dataset of interest into multiple partitions,

where a single partition constitutes a cluster [6, 23]. The data within a single cluster have similar characteristics, whilst data in different clusters have dissimilar characteristics.

Hierarchical techniques

In this method a hierarchical structure is imposed on the dataset and their step-wise clusters, for instance, one extreme of the clustering structure is only one cluster containing all the data in the dataset [26]. This technique is said to be rigid as it prohibits re-organisation of clusters established in the previous steps. Hierarchical techniques can be further sub divided into those that use a top-bottom or bottom-up approaches.

Grid-based techniques

This technique utilises a multiresolution grid data structure for partitioning the dataset. It quantises the space occupied by the dataset into a finite number of cells that form a grid structure where all the clustering operations are performed.



Density-based techniques

Density-based techniques discover arbitrary- shaped clusters. These clusters are regarded as regions in which the density of data in the dataset exceeds a certain threshold. The DBSCAN and SSIV algorithms are some examples of density-based techniques.

Model-based techniques

This approach uses a mathematical model as a benchmark for clustering and are based on the assumption that the dataset is generated by a mixture

of underlying probability distributions. They attempt to optimise the fit between the dataset and the model. These techniques can be further subdivided into Statistical approaches and Neural network approaches.

In this study we will make use of a combination of clustering techniques, a GMM with Kmeans as well as a GMM with Kmedoids. The GMM is categorised under Model-based techniques while Kmeans and Kmedoids are classified under partitioning techniques.

Kmeans algorithm

The Kmeans algorithm is one of the simplest unsupervised learning algorithms that are known to solve clustering problems [26]. It is a centroid based technique, i.e. finds clusters in a dataset such that the distance measure is minimised [5, 23]. The distance measure chosen in most cases is the Euclidean distance, given by;

$$d(x_i, v_j) = \sqrt{\sum_{k=1}^n (x_k^i - v_k^j)^2}$$


where n is the number of dimensions of each data point, x_k^i is the value of the k -th dimension of x_i and v_k^j is the value of the k -th dimension of v_j . The algorithm partitions the dataset into K clusters (C_1, C_2, \dots, C_K), represented by their means or centers. The center of each cluster is calculated as the mean of all the data points that belong to that cluster and it is a point to which the sum of distances from all data points in that cluster is minimised [23]. The Kmeans algorithm takes the number of clusters to be formed, K , and X the dataset on which the clustering is to be performed on as inputs. The

algorithm returns a set of K clusters.

Kmeans algorithm uses the following steps:

1. Initialise the cluster centers $C_i = \{C_1, C_2, \dots, C_K\}$ by randomly selecting K points from the dataset X .
2. Assign each data point in X to the cluster with the closest centroid C_i . The partitioned groups are defined by a $c \times n$ binary membership matrix U .
3. When all the data points have been assigned, recalculate the positions of the K centroids.
4. Repeat steps 2-3 until no new centroids can be found.

Tab. 3.2: Advantages and disadvantages of Kmeans

 University of Fort Hare <i>Together in Excellence</i>	
Advantages	Drawbacks
Kmeans works well with ill-separated clusters	Suffers when dealing with large datasets due to the predefinition of number of clusters
Works well with natural images	Has difficulties with clusters of different sizes, densities and shapes
Is easy to implement	Outliers affect the Kmeans algorithm
Time complexity is $O(n)$, where n is the number of data points	

Kmedoids algorithm

The Kmedoids algorithm is another techniques that is classified under the partitioning clustering techniques. It attempts to minimise the Sum of Squared Error(SSE). This algorithm finds a medoid in a cluster, which is

a centrally located point in a cluster. According to Arora et. al.(2016), Kmedoids is more robust as compared to Kmeans as the Kmeans algorithm is sensitive to outliers. This algorithm reduces the disadvantages experienced by the Kmeans algorithm as it is based on data point representative methods [6, 23]. In the Kmedoids algorithm, each cluster is represented by the most central data point in the cluster, rather than the implicit mean that may not belong to the cluster. The algorithm partitions the dataset into K clusters by first finding a representative data point for each cluster. It then assigns the remaining data points to the cluster with the medoid to which they are most similar to. The Kmedoids algorithm takes the number of clusters to be formed, K, and X the dataset. The output of the algorithm is a set of K cluster.



Kmedoids follows the steps described below:

University of Fort Hare

1. Initialise the cluster representatives by randomly selecting K data points as medoids.
2. Associate the remaining data points to the nearest medoid by calculating the distance between data points and the medoid of each cluster.

This is done by;

- Performing element by element binary operations to compute $A = X - \text{mean}(X, 2)$.
- Computing $S = A^2$, then convert the sparse matrix S to a full matrix (S).
- Computing $D = -2 \times (A^T)A$, where A^T is the transpose of matrix A.

- Computing $D = D + S^T$ where S^T is the transpose of matrix S .
 - Selecting the minimum from D .
3. For each data point p associated with the medoid K , swap p and K then compute the total cost.
 4. Compute minimum cost c from the total cost.
 5. Add c to final cost.
 6. Repeat steps 2-5 until there are no new medoids found.

Tab. 3.3: Advantages and disadvantages of Kmedoids

Advantages	Drawbacks
Kmedoids is immune to noise and outliers	Kmedoids is computationally costlier than Kmeans
It is computationally more intensive	Applied only when dealing with categorical data
Can work with any distance measure	Does not scale well to for large datasets

3.7 Distance measures and metrics

According to Pandit et al.(2014), distance measures are used to calculate similarity or dissimilarity between two data objects within a given dataset. They reflect the degree of separation among data points within a dataset and distinguish the clusters embedded in the dataset [44]. Distance measures include Manhattan, Minkowski, Hamming, Jaccard, Cosine, Euclidean and Squared Euclidean distance amongst others. Some of the distance measures are also referred to as distance metrics. For a distance measure to be considered a metric, it has to satisfy the following conditions [44];

1. The distance between any two data points must be non- negative, i.e., $d(x_i, x_j) \geq 0$ for all x_i and x_j .
2. The distance between two data points must be equal to zero if and only if the data points are the same, i.e., $d(x_i, x_j) = 0$ if and only if $x_i = x_j$.
3. The distance from x_i to x_j is the same as the distance from x_j to x_i , i.e., $d(x_i, x_j) = d(x_j, x_i)$.
4. The triangle inequality is satisfied, i.e., $d(x_i, x_j) + d(x_j, x_k) \geq d(x_i, x_k)$ for all x_i, x_j and x_k .

The City block, Minkowski, Hamming and Euclidean distance satisfy the conditions of a distance metric and thus are distance metrics. Kumar et al. [33], maintains that there is no single distance measure that best suits all clustering problems. In this study a comparison of the Grabcuts for image segmentation algorithm with a GMM based on Kmeans and Kmedoids is performed with three distance measures. These are the Squared Euclidean, Cosine and Manhattan distance measures. We shall now give the descriptions of these distance measures.

Euclidean distance

The Euclidean distance measure refers to the straight line distance between two data points [33]. It is given by the formula, [33, 44]

$$d_e(x_i, x_j) = \sqrt{\sum_{k=1}^n |x_{ik} - x_{jk}|^2}$$

where x_{ik} and x_{jk} represent the k -th dimension of x_i and x_j , respectively.

The strength of this distance measure is in its ability to form clusters that are invariant to translation and rotation in the feature space [33].

Squared Euclidean distance

The squared Euclidean distance is given by the sum of squared differences between corresponding data point values. It is defined as,

$$d_{Se}(x_i, x_j) = \sum_{k=1}^n |x_{ik} - x_{jk}|^2$$

City block distance

This distance measure is also known as the Manhattan distance. The City block distance between two data points is defined as the absolute difference of the coordinates of the data points [33]. It is Known as the Manhattan distance as it is similar to the walking distance between two points in a city like Manhattan, where one has to walk around the buildings instead of going through them. That is, the distance between two data points is measured along the axes at right angles. The formula for the Manhattan distance is given by,

$$d_M(x_i, x_j) = \sum_{k=1}^n |x_{ik} - x_{jk}|$$

The City block distance is said to have an advantage over the Euclidean distance as it has reduced computation time [8, 33, 60]. Although this is the case, this distance measure depends upon the rotation of the coordinate system which is seen as its limitation [33].

Cosine distance

The Cosine distance is a measure of similarity or dissimilarity between two vectors of n dimensions. This measure is determined by the cosine of the

angle between the two vectors of interest [8]. the Cosine distance is given by,

$$d_{cos}(x_i, x_j) = 1 - \frac{x_i^T x_j}{\|x_i\| \|x_j\|}$$

where x_i^T is the transpose of x_i . The Cosine distance is used to measure cohesion within clusters and lies between 0 and 1. This distance measure violates the triangle inequality and thus is not a distance metric [33]. Furthermore, it is invariant to scaling and not invariant to shifts.



University of Fort Hare
Together in Excellence

4. PERFORMANCE MEASURES

4.1 *Evaluation metrics*

Many researchers [12, 56, 60], in the field of image processing have been frequently pointing out the need for a standard quality measure for both the evaluation and comparison of image segmentation algorithms. These researchers maintain that few efforts have been made on the evaluation of segmentation algorithms, regardless of the realised significance of performance evaluation. According to Zhang [60], segmentation algorithms can be evaluated using either analytical or empirical methods. The analytical method directly investigates and assesses the segmentation algorithms themselves by analysing their principles and properties. Udupa et al. [56], maintains that the analytical method has major drawbacks as some of the characteristics of a segmentation cannot be obtained and described analytically. In addition, the analytical method avoids the implementation of the algorithms, thus making it difficult to compare algorithms entirely by analytical studies [60].

On the other hand, empirical methods indirectly criticise the image segmentation algorithms by measuring the quality of segmentation results after the algorithm has been applied to the test images. The empirical method can further be classified under two categories, viz. the goodness method and

discrepancy method [60]. The empirical goodness method measures some desirable traits of segmented images that are often established according to human intuition using goodness parameters. The value of goodness measure is used to judge the performance of the segmentation algorithms. According to Cardoso et al. [12], the empirical goodness method does not seek a prior knowledge of the reference segmentation as it evaluates and rates the different algorithms by computing goodness measures based on the segmented image. There are different types of goodness measures, including Colour uniformity [57], Entropy [43], intraregion uniformity [60], inter-region uniformity[60] and the region shape [60] amongst others.

The empirical discrepancy method is based on the availability of a reference segmentation, sometimes called a ground truth, which is an ideal or expected segmentation result. The image segmentation algorithm's performance is assessed using the disparity between the actual segmented image and the ground truth. Both the actual segmented image and the ground truth are obtained from the same input image [12]. The method aims to determine how far the actually segmented image is from the reference image. In this method, the performance of the applied image segmentation algorithm is assessed using discrepancy measures, where a high value of discrepancy measure implies a bigger error in the segmented image in relation to the ground truth [60]. This leads to a low performance in the image segmentation algorithm. The different types of discrepancy measures include those based on; the number of mis-segmented pixels, position of mis-segmented pixels, number of objects in the image, feature values of segmented objects



University of Port Hare
Together, in Excellence

and miscellaneous quantities, amongst others [60].

4.2 Intra-cluster distance

The Intra-cluster distance evaluates the compactness of a clustering solution. This is achieved by measuring the average distance between the centroid and data points of a cluster [5, 24]. The intra-cluster distance is given by;

$$I_{intra} = \frac{\sum_{k=1}^K \sum_{\forall x \in C_k} d(x_p, c_k)}{|P|}, \quad (4.1)$$

where c_k is a centroid of cluster C_k , x_p is a data point and $|P|$ is the total number of data points in a dataset.



University of Fort Hare
Together in Excellence

4.3 Inter-cluster distance

The inter-cluster distance assesses the separation of the clusters formed by a clustering technique. This is done by determining the average distance between different clusters which is calculated as the difference between the centers of different clusters [5, 24]. The inter-cluster distance is defined as;

$$I_{inter} = \frac{2}{K(K-1)} \sum_{k=1}^{K-1} \sum_{k_2=k+1}^K d(c_k, c_{k_2}), \quad (4.2)$$


where $d(c_k, c_{k_2})$ is the Euclidean distance between the centroids c_k and c_{k_2} and K is the number of clusters formed.

4.4 Validity Indices

According to Ansai et al., Validity indices are quantitative measures used to assess the solutions of clustering techniques. Due to the fact that clustering techniques can form different clusters, the need of quality measures to evaluate these clusters is realised. There are various validity indices that exist, we shall present a few examples below.

4.4.1 Dunn's validity index

The Dunn's validity index is used to describe how compact and separated clusters formed by a clustering technique are [4, 24]. The larger the value of the index, the better the clustering solution. The following formula is used to give the Dunn's validity index;



University of Fort Hare
Together in Learning

$$D = \min_{1 \leq i \leq k} \left(\min_{i+1 \leq j \leq k} \left(\frac{d(c_i, c_j)}{\max_{i+1 \leq j \leq k} \text{diam}(c_j)} \right) \right), \quad (4.3)$$

Where $d(c_i, c_j)$ is the distance between cluster c_i and c_j ; $d(c_i, c_j) = \min_{\substack{x_i \in c_i \\ x_j \in c_j}} d(x_i, x_j)$ and $d(x_i, x_j)$ is the distance between data points $x_i \in c_i$ and $x_j \in c_j$. The diameter of cluster c_l is given by $\text{diam}(c_l) = \max_{x_{l_1}, x_{l_2} \in c_l} d(x_{l_1}, x_{l_2})$.

4.4.2 Davies-Bouldin validity index

The Davies-Bouldin validity index gives the average similarity between each cluster and the one that is most similar to it [24]. The best clustering solution under this index is one in which the value of the index is minimised. The

Davies-Bouldin validity index is defined as;

$$DB = \frac{1}{k} \sum_{i=1}^k \max_{1 \leq j \leq k, j \neq i} \left(\frac{\text{diam}(c_i) + \text{diam}(c_j)}{d(c_i, c_j)} \right). \quad (4.4)$$

4.4.3 Silhouette validity index

The silhouette validity index is the silhouette width for each cluster and overall average silhouette width for the total dataset [4]. The index is computed using the formula;

$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

,

where a_i is the average dissimilarity of the i -th data point to all other points within the same cluster and b_i is the minimum average dissimilarity from the i -th data point to a data point in a different cluster. The dissimilarity can be measured using any of the distance measures mentioned in Section 3.7. The index values range from -1 to 1, where 1 is said to be the highest value which implies that a data point i is well-matched to its own cluster. On the other hand, a low or negative silhouette value indicates that the clustering solution may either have too few or too many clusters. Table 4.1 shows the silhouette values and their interpretation.

4.4.4 Rand validity index

This validity index measures the number of pairwise similarity between the set of discovered clusters and set of class labels. The Rand index is defined

Tab. 4.1: Silhouette validity index values and their interpretation [24]

Silhouette Value	Interpretation
0.71-1.00	A strong clustering structure has been found
0.51-0.70	A reasonable clustering structure has been found
0.26-0.50	A weak clustering structure has been found
≤ 0.25	No adequate clustering structure has been found

as;

$$R = \frac{a + d}{a + b + c + d} \quad (4.5)$$

where a is the number of pairs of data points that have the same label C and are assigned to the same cluster, b denotes the number of pairs with the same label, but are assigned to different clusters. The number of pairs of data points in the same cluster, but have different class labels are denoted by c , and d denotes the number of pairs of data points that have different class labels that are assigned to different clusters. The Rand index values range from 0 to 1, where a high Rand index value indicates a high level of agreement between the class and natural labels.

4.4.5 Jaccard validity index

This index assesses the agreement between different clusters formed for a given dataset. The similarity between the class labels and clustering solution is determined by the number of pairs of data points assigned to the same cluster in both the class labels as well as in the clustering solution. The Jaccard validity index is given by;

$$J = \frac{a}{a + b + c} \quad (4.6)$$

where $a, b,$ and c denote the same number of pairs of data points as in the Rand validity index. The index value ranges from 0 to 1, where an index value of 1 indicates that the class labels and clusters discovered are identical.

4.5 Precision

The Precision, or Positive Predictive Value (PPV) is used to measure how accurate of the boundaries of the segmentation results are [19]. This is done by calculating the rate of true positives among the segmentation result and the ground-truth. Precision is calculated as follows;

$$P = \frac{TP}{TP + FP}$$

Precision values range from 0 to 1. The true positive, true negatives, false positives and false negatives are described in Table 4.2.

Tab. 4.2: Confusion matrix notation

	Ground-truth	
Segmented image	True positive (TP)	False negative (FN)
	False positive (FP)	True negative (TN)

4.6 Recall

The recall, True Positive Value (TPV) or sensitivity measures how good a test or algorithm is at detecting the true positives between the segmented image and the ground-truth [22]. Recall refers to is the relation between true


positives and all positive elements and is given by the equation;

$$R = \frac{TP}{TP + FN}.$$

Recall vales range from 0 to 1, where a value of 1 indicates a perfect match of between the segmentation result and the ground-truth.

4.7 *BF-score*

The BF-score is the harmonic mean of the precision and recall with a distance error tolerance to decide whether a point on the segmented image's boundary has a match on the ground-truth boundary or not.[22]. It is used to measure efficiency and success of segmentation based on the values of precision and recall. The BF-score is given by the formula,



University of Fort Hare
Together in Excellence

$$BF = \frac{2 * Precision * Recall}{Recall + Precision}.$$

The value of the BF-score ranges from 0 to 1, where a score of 1 means that the contours of objects in the corresponding class in the segmented image and ground-truth are a perfect match.

An evaluation metric is desired to take into consideration over segmentation, under segmentation, inaccurate boundary localization and different number of segments. The precision, recall and BF-score are used to determine whether the segmented image falls under any of the categories mentioned above. In addition, these evaluation measures are controlled by a threshold, distance of tolerance θ which determines whether a boundary point has a

match or not. According to Prabha et al. [19], over segmentation occurs when the value of the precision is low. On the other hand, under segmentation occurs when the recall is low. This study utilises an empirical method to measure the performance of the algorithms. This is due to the fact that the analytical method for evaluation of image segmentation algorithms is not sufficient as it cannot describe all the qualities of the algorithm quantitatively [60]. Thus this study uses the empirical discrepancy method. This method exempts the influence of human factors and provides consistency and non-biased results [60]. In addition, the quality measures used to assess the image segmentation algorithm can be numerically computed [60]. Thus, the empirical discrepancy method is the most suitable evaluation method for the proposed algorithm as it is both objective and quantitative.

The separability and compactness of the clusters generated by the clustering techniques was measured. This was done by computing the intra cluster distance and inter cluster distance. According to Arbin et al. [5], the intra cluster distance refers to the mean distance between data points within a cluster and inter cluster distance is mean distance between centroids of the clusters in a given dataset. In addition, the Silhouette validity index was computed for all algorithms. Furthermore, the segmentation results for the algorithms are also presented. The precision, recall and BF-score with $\theta = 0.5$ for the segmentation results are also provided.

5. EXPERIMENTATION AND RESULTS

Since the development of the grabcut algorithm of Rother, Kolmogorov and Blake [48], many researchers have come up with ways of improving the algorithm, i.e. running time, efficiency, etc [28, 35, 54]. This study aims at providing a comparison between a Grabcuts algorithm with GMM based on Kmeans and that with a GMM based on Kmedoids. The Grabcuts algorithm considered in this study is that which is based on the work in [48] and [9]. The algorithm will be allowed to run with a GMM based on Kmeans and also with a GMM based on Kmedoids. The algorithms mentioned above will be executed under two scenarios. In addition, these algorithms utilised the same colour images obtained from BSDS500 database. Furthermore, the number of clusters have been varied from 2 to 5 for all algorithms. In this chapter we will look at the performance of the Grabcuts for image segmentation algorithm with a GMM based on Kmeans developed in [48] as well as the performance of the Grabcuts algorithm with a GMM based on the Kmedoids clustering technique. We consider two scenarios in the experiment. In the first scenario, GMM's based on Kmeans and Kmedoids clustering techniques that use the Squared Euclidean distance measure are considered. In the second scenario the Kmeans and Kmedoids clustering techniques utilise the City block distance measure. In the works of [6, 48], the number of clusters is

chosen arbitrary to be 5. Although this is the case some researchers [5], have varied K from 2 to 5. In this study the number of clusters, K, is varied from 2 to 5 for all the algorithms so as to observe if the number of clusters chosen has an effect on the segmentation results. The same colour images obtained from the BSDS500 database were used to test all algorithms under the two scenarios. Images and their ground-truth annotations from the Berkeley Segmentation Dataset and Benchmark 500(BSDS500) [37] database are used to evaluate the algorithms.

Figure 5.1 shows the images used to test the algorithms in this study. The images in Figure 5.1 (a)-(l) will be referred to as Butterfly, Insect, Soldier, Kids, Swimmer, Car, Aeroplane, Flower, Swan, Statues, Star-fish, and Horses. These images were deliberately chosen as they are considered difficult images as they contain regions of low contrast at transition areas from background to foreground [48, 54]. In addition, these images have regions which partially overlap in colour space between foreground and background regions. The researchers selected these image to investigate the effect these images have on tested algorithms running time and segmentation results.



University of Fort Hare
Together in Excellence



(a)



(b)



(c)



(d)



(e)



(f)



(g)



(h)



(i)



(j)



(k)



(l)

Fig. 5.1: Images from BSDS500 database

5.1 Experimental setup

The algorithms executed in this study are modified algorithms developed from the algorithms provided in [48], [9] and [17]. All experimental tests of the algorithms were conducted on the Matrix Laboratory (MATLAB) software version R2016a. This software allows the user to perform data manipulation and visualisation, image analysis and calculations amongst others operations. The machine used is a Fujitsu computer with Intel® Core™ i5-4300M CPU @ 2.60 GHz processor and 8.00 GB RAM running on a Windows 7 operating system. When the MATLAB codec is executed for the algorithms, a Graphics User Interface (GUI) will open, Figure 5.2 shows its image;

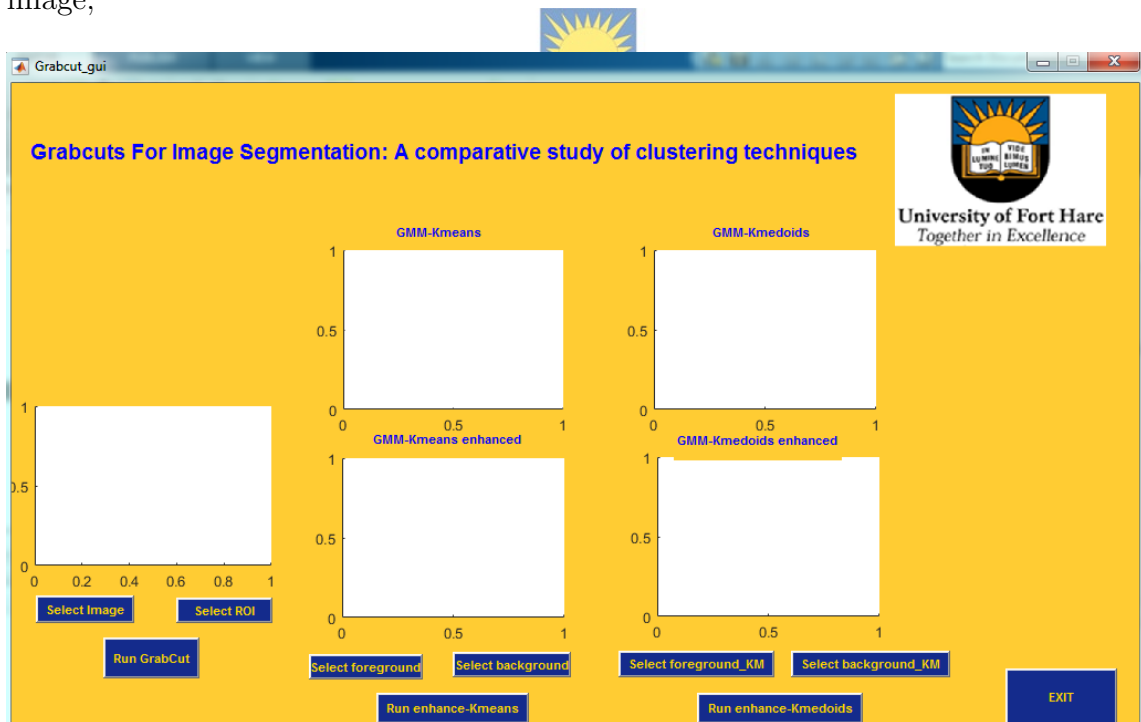


Fig. 5.2: Grabcuts for image segmentation: A comparative study of clustering techniques GUI

The working order of the GUI is from left to right. The following are steps on how the GUI works;

- The user is required to press the “Select image” button to select an image from any folder in the computer.
- Then the “Select ROI ”button is pressed to mark the background pixels on the original image, which is located on the left most axis.

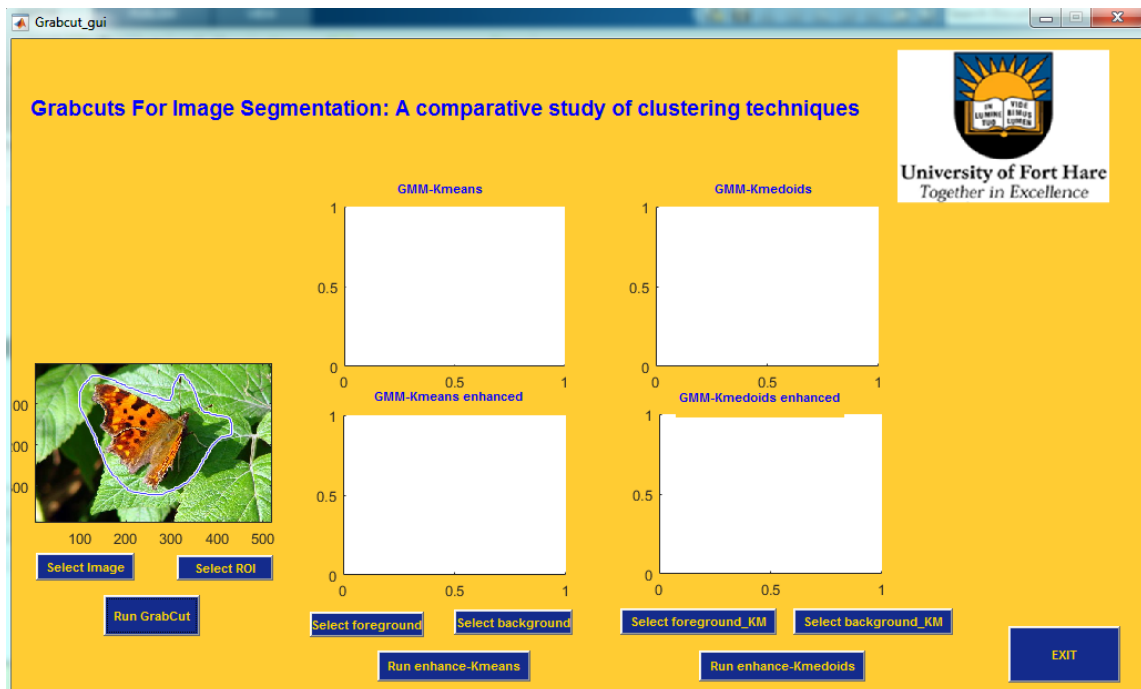


Fig. 5.3: Selection of ROI from image loaded in the GUI

- “Run Grabcut” button is pressed to run the algorithm. The Grabcuts algorithm with GMM based on Kmeans will run first followed by the algorithm with GMM based on Kmedoids. When the algorithms finish running, the segmented images appear on the first axes under the words

Kmeans and Kmedoids, respectively. There is also an option to enhance the result obtained in this step.

- There are two sections for the enhancement of the segmentation, one is for marking the foreground and the other for marking the background. To mark a foreground region for the Kmeans based algorithm, the “Select foreground” button is pressed and “Select background” button to mark a background region. After selecting the preferred region one can proceed to press the “Run enhance Kmeans” button, the enhanced image will appear on the axis below the segmented image. On the other hand, for the Kmedoids the buttons “Select foreground_KM”, “Select background_KM” and “Run enhance-Kmedoids” to mark foreground region, background region and to enhance segmentation result.



University of Fort Hare
Together in Excellence

5.2 *Experimental results*

5.2.1 *Performance of Grabcuts for image segmentation algorithm under Scenario 1*

The results obtained from the running of the Grabcuts image segmentation algorithms with a GMM based on Kmeans and Kmedoids clustering techniques that utilised the Squared Euclidean distance are presented in this section. These algorithms were implemented on the images provided in Figure 5.1 and the number of clusters, K , was varied from 2 to 5.

One of the important aspects that is considered to be a deciding factor in choosing one of the two clustering techniques is their runtime. Table 5.1 presents the average runtime for the Kmeans clustering techniques as com-

pared to the that of the Kmedoids clustering technique.

Tab. 5.1: Average runtime: Kmeans versus Kmedoids

Image	K	Average runtime in seconds (s)	
		Kmeans	Kmedoids
Butterfly	2	0.480	6.27
	3	0.0447	1.65
	4	0.122	4.16
	5	0.349	5.69
Insect	2	0.147	2.09
	3	0.207	3.57
	4	0.524	4.13
	5	0.343	5.26
Soldier	2	0.160	4.25
	3	0.259	5.38
	4	0.398	8.82
	5	0.845	12.19
Kids	2	0.0950	3.49
	3	0.161	4.26
	4	0.328	4.82
	5	0.403	4.78
Swimmer	2	0.0981	2.24
	3	0.247	3.79
	4	0.217	4.61
	5	0.235	4.88

Car	2	0.0761	1.31
	3	0.147	2.22
	4	0.202	1.98
	5	0.192	2.59
Aeroplane	2	0.0654	2.00
	3	0.126	1.91
	4	0.353	3.41
	5	0.316	3.59
Flower	2	0.154	2.26
	3	0.206	3.74
	4	0.213	4.67
	5	0.321	5.06
Swan	2	0.0573	1.15
	3	0.177	3.09
	4	0.177	6.07
	5	0.400	5.99
Statues	2	0.249	2.41
	3	0.321	3.30
	4	0.304	3.95
	5	0.361	4.27
Star-fish	2	0.135	2.51
	3	0.208	5.29
	4	0.265	5.66
	5	0.354	6.39



University of Fort Hare
Together in Excellence

Horses	2	0.231	3.20
	3	0.186	3.36
	4	0.323	4.23
	5	0.498	6.84

It is observed from Table 5.1 that the Kmeans clustering technique outperforms the Kmedoids techniques for all the images. In addition, the Kmeans clustering technique achieved the lowest and highest runtimes of 0.0447 and 0.845 seconds, respectively. On the other hand, the Kmedoids clustering technique achieved the lowest and highest runtimes of 1.15 and 12.19 seconds. The highest runtimes for both clustering techniques were achieved when the Grabcuts for image segmentation algorithm was executed on the Soldier image with K= 5. In addition, most images achieved lowest and highest runtimes when K=2 and K=5 for both the Kmeans and Kmedoids clustering techniques.

The number of iterations taken by the Kmeans and Kmedoids clustering techniques to reach convergence are shown in Figure 5.4. These are the number of iterations required in order for the algorithms to create K Gaussian components which are required for assignment of both foreground and background pixels. A similar trend as that of the runtime achieved by both the Kmeans and Kmedoids clustering techniques is observed from the number of iterations required for convergence by the two techniques, where the number of iterations required by the Kmeans is less than that required by the

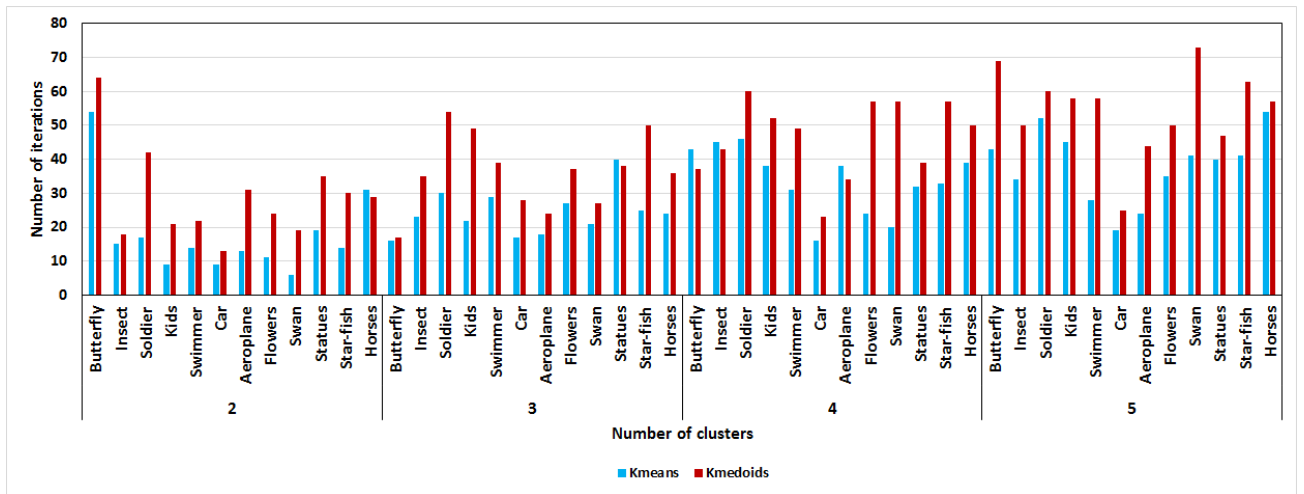


Fig. 5.4: Number of iterations: Kmeans versus Kmedoids clustering technique

Kmedoids clustering technique. The Kmeans clustering technique converged rapidly with the lowest and highest observed number of iterations of 5 and 66, respectively. On the other hand, the Kmedoids technique manages to converge at the lowest and highest number of iterations of 13 and 85, respectively. In addition, the lowest number of iterations is achieved when $K=2$ for both clustering techniques in all images. Furthermore, when $K=5$ most images required the highest number of iterations.

To determine the separability and compactness of the clustering techniques used in this study, two measures were computed. These are the intra and inter-cluster distance. The main aim of any clustering technique is to minimise the intra-cluster distances which is the sum of dissimilarities of data points within a cluster [6]. This is because the desired outcome of the clustering technique is to have groupings of data points such that all the similar points are grouped together and hence have close to no overlapping clusters. Table 5.2 presents the average intra-cluster distances and standard

deviations obtained when the Grabcuts for image segmentation algorithm was executed with GMM-Kmeans and GMM-Kmedoids.

Tab. 5.2: Averages and Standard deviations of Intra-cluster distance

Image	K	Average intra-cluster distance (1×10^7)	
		Kmeans	Kmedoids
Butterfly	2	4.457 \pm 1.877	4.390 \pm 2.027
	3	15.16 \pm 8.446	15.09 \pm 8.623
	4	8.792 \pm 4.554	8.518 \pm 4.885
	5	5.756 \pm 2.663	5.659 \pm 2.924
Insect	2	4.879 \pm 2.429	4.936 \pm 2.557
	3	3.426 \pm 1.794	3.405 \pm 1.890
	4	2.689 \pm 1.283	2.665 \pm 1.241
	5	2.178 \pm 1.222	2.219 \pm 1.240
Soldier	2	18.37 \pm 6.578	18.41 \pm 6.590
	3	10.92 \pm 5.040	10.94 \pm 5.050
	4	7.303 \pm 3.601	7.331 \pm 3.747
	5	5.657 \pm 2.904	5.652 \pm 2.687
Kids	2	10.26 \pm 3.528	10.27 \pm 3.532
	3	5.507 \pm 1.785	5.523 \pm 1.799
	4	3.592 \pm 1.081	3.612 \pm 1.065
	5	2.544 \pm 0.7083	2.567 \pm 0.7235
Swimmer	2	20.23 \pm 11.54	20.41 \pm 11.40
	3	13.89 \pm 7.313	13.96 \pm 7.458

	4	9.400 ± 5.396	9.541 ± 5.570
	5	6.615 ± 3.874	6.637 ± 3.883
Car	2	15.35 ± 6.478	15.58 ± 6.468
	3	8.001 ± 3.763	7.566 ± 2.942
	4	5.865 ± 3.615	5.876 ± 3.724
	5	4.239 ± 3.182	3.946 ± 2.349
Aeroplane	2	5.420 ± 2.853	5.396 ± 2.802
	3	3.529 ± 1.662	3.533 ± 1.641
	4	2.536 ± 1.118	2.538 ± 1.206
	5	1.843 ± 0.7982	1.601 ± 0.9408
Flowers	2	15.74 ± 8.876	15.62 ± 8.787
	3	8.794 ± 4.760	8.392 ± 4.281
	4	6.163 ± 3.981	6.037 ± 3.751
	5	4.494 ± 2.804	4.528 ± 2.918
Swan	2	8.460 ± 3.884	8.412 ± 3.874
	3	4.890 ± 1.260	4.673 ± 1.460
	4	2.697 ± 0.6939	2.729 ± 0.6974
	5	2.009 ± 0.5253	2.077 ± 0.5860
Statues	2	20.31 ± 10.67	20.00 ± 11.07
	3	11.38 ± 4.948	10.34 ± 6.013
	4	6.152 ± 3.298	6.245 ± 3.426
	5	4.481 ± 2.011	4.505 ± 2.010
Star-fish	2	16.83 ± 7.183	16.86 ± 7.189
	3	9.834 ± 4.315	9.845 ± 4.326

	4	6.865 ± 3.120	7.138 ± 3.319
	5	5.356 ± 2.517	5.359 ± 2.371
Horses	2	13.73 ± 4.642	13.58 ± 4.383
	3	7.633 ± 2.856	7.640 ± 2.861
	4	5.478 ± 2.125	5.444 ± 2.012
	5	4.259 ± 1.473	4.281 ± 1.468

The lowest and highest intra-cluster distances observed were obtained when $K=2$ for the Insect and Butterfly images, respectively, for both clustering techniques. The Kmeans clustering technique obtained the lowest and highest intra-cluster distance of 1.843×10^7 and 20.31×10^7 . The Kmedoids technique, on the other hand, obtained the lowest intra-cluster distance of 1.601×10^7 and a highest intra-cluster distance of 20.41×10^7 . In addition, the Kmeans clustering technique obtained the lowest intra-cluster distance for most images than the Kmedoids technique. Although this is the case, the Kmedoids clustering technique obtained the lowest intra-cluster distance across all images for $K=2$ to 5.

The separability of the clustering solution was measured using the inter-cluster distance. The aim of a clustering technique is to ensure that the clusters formed are dissimilar. This implies that the clustering technique aims to maximise the inter-cluster distance. Table 5.3 shows the average inter-cluster distance and standard deviations for the clustering solution obtained by the Kmeans clustering technique as compared to that obtained by the Kmedoids clustering technique.

Tab. 5.3: Averages and Standard deviations of Inter-cluster distance

Image	K	Inter-cluster distance	
		(1×10^9)	
Butterfly	2	9.918 ± 3.464	9.609 ± 4.141
	3	3.648 ± 1.588	3.651 ± 1.607
	4	5.413 ± 2.402	5.335 ± 2.520
	5	7.614 ± 2.837	7.343 ± 3.305
Insect	2	0.4382 ± 0.1634	0.4351 ± 0.1741
	3	0.8126 ± 0.2591	0.7674 ± 0.2513
	4	1.191 ± 0.2710	1.189 ± 0.2194
	5	1.437 ± 0.3323	1.428 ± 0.3375
Soldier	2	1.644 ± 0.2285	1.640 ± 0.2327
	3	2.941 ± 0.6128	2.935 ± 0.6087
	4	5.199 ± 1.996	5.077 ± 1.992
	5	6.780 ± 2.687	5.384 ± 2.223
Kids	2	2.303 ± 1.045	2.308 ± 1.053
	3	3.420 ± 1.474	3.413 ± 1.493
	4	4.411 ± 1.774	4.398 ± 1.711
	5	5.853 ± 2.332	5.963 ± 2.371
Swimmer	2	2.101 ± 1.498	2.122 ± 1.490
	3	3.771 ± 2.618	3.831 ± 2.755
	4	5.701 ± 4.327	5.658 ± 4.288
	5	7.036 ± 5.212	7.069 ± 5.231
Car	2	1.691 ± 0.4661	1.669 ± 0.5040

	3	2.777 ± 0.5362	2.796 ± 0.4966
	4	3.856 ± 0.8225	3.863 ± 0.8944
	5	4.978 ± 1.833	4.858 ± 1.047
Aeroplane	2	0.9817 ± 0.5094	0.9850 ± 0.5258
	3	1.637 ± 0.5464	1.623 ± 0.5405
	4	2.314 ± 0.7165	2.127 ± 0.7409
	5	2.672 ± 0.3327	2.717 ± 0.2996
Flowers	2	2.022 ± 0.9244	1.970 ± 0.8586
	3	3.239 ± 1.370	2.979 ± 0.9976
	4	5.030 ± 2.255	4.737 ± 1.965
	5	7.780 ± 2.603	7.182 ± 2.236
Swan	2	2.474 ± 1.631	2.458 ± 1.643
	3	3.942 ± 2.186	3.335 ± 1.586
	4	5.600 ± 2.384	5.458 ± 2.415
	5	7.411 ± 2.451	7.187 ± 2.496
Statues	2	1.936 ± 1.312	1.845 ± 1.276
	3	5.072 ± 1.456	3.678 ± 2.234
	4	5.027 ± 3.300	5.103 ± 3.246
	5	8.921 ± 2.504	8.972 ± 2.577
Star-fish	2	1.699 ± 0.5582	1.702 ± 0.5493
	3	2.820 ± 0.9481	2.822 ± 0.9478
	4	4.864 ± 1.777	4.892 ± 1.795
	5	6.157 ± 2.229	6.179 ± 2.087
Horses	2	1.035 ± 0.3470	1.034 ± 0.3393

3	1.952 ± 0.5179	1.945 ± 0.5154
4	2.741 ± 0.5835	2.753 ± 0.6039
5	3.738 ± 0.8266	3.722 ± 0.8195

The lowest and highest inter-cluster distances observed was obtained when $K=2$ for the Insect and Butterfly images, respectively, for both clustering techniques. The Kmeans and Kmedoids clustering techniques obtained lowest inter-cluster distances of 0.4382×10^9 and 0.4351×10^9 , respectively. On the other hand, the highest inter-cluster distance obtained by the two techniques were 9.918×10^9 and 9.609×10^9 , respectively. The Kmeans clustering technique obtained a higher inter-cluster distance for most images with $K=2$ to 5 than the Kmedoids clustering technique. In addition, the technique obtained the highest inter- cluster distance across all images for $K=2$ to 5.



University of Fort Hare
Together in Excellence

5.2.2 Optimal number of clusters

The silhouette validity index was calculated for all the clustering solutions found by both the Kmeans and Kmedoids clustering techniques. The silhouette value of the clusters discovered by clustering techniques indicates whether the data points are appropriately clustered or not. The best clustering solution is observed when the silhouette value is very close to 1 and the worst clustering solution is observed when the silhouette value is very close to -1. This means that the data points were misclassified and lie somewhere in between clusters. In addition the number of clusters with the largest average silhouette value is taken to be the optimal number of clusters [4]. The average silhouette values and their standard deviations are given in Table 5.4 and graphical comparisons of these value are provided in Figure 5.5 to 5.16.



Tab. 5.4: Averages and Standard deviations of Silhouette validity index

Image	K	Silhouette value	
		Kmeans	Kmedoids
Butterfly	2	0.702 ± 0.056	0.676 ± 0.064
	3	0.895 ± 0.021	0.896 ± 0.021
	4	0.808 ± 0.044	0.811 ± 0.045
	5	0.755 ± 0.061	0.735 ± 0.045
Insect	2	0.720 ± 0.140	0.719 ± 0.142
	3	0.631 ± 0.123	0.631 ± 0.123
	4	0.590 ± 0.125	0.592 ± 0.120
	5	0.604 ± 0.088	0.614 ± 0.078
Soldier	2	0.727 ± 0.047	0.726 ± 0.048

	3	0.641 ± 0.070	0.641 ± 0.070
	4	0.636 ± 0.043	0.635 ± 0.045
	5	0.590 ± 0.058	0.594 ± 0.059
Kids	2	0.867 ± 0.020	0.867 ± 0.020
	3	0.787 ± 0.028	0.785 ± 0.028
	4	0.684 ± 0.030	0.682 ± 0.032
	5	0.664 ± 0.014	0.665 ± 0.015
Swimmer	2	0.707 ± 0.074	0.709 ± 0.074
	3	0.625 ± 0.078	0.622 ± 0.085
	4	0.674 ± 0.043	0.663 ± 0.043
	5	0.645 ± 0.027	0.645 ± 0.027
Car	2	0.765 ± 0.021	0.760 ± 0.020
	3	0.711 ± 0.048	0.724 ± 0.030
	4	0.726 ± 0.048	0.736 ± 0.028
	5	0.719 ± 0.013	0.714 ± 0.024
Aeroplane	2	0.800 ± 0.166	0.800 ± 0.166
	3	0.660 ± 0.104	0.660 ± 0.104
	4	0.587 ± 0.050	0.640 ± 0.044
	5	0.628 ± 0.060	0.629 ± 0.053
Flowers	2	0.814 ± 0.029	0.814 ± 0.030
	3	0.726 ± 0.038	0.714 ± 0.029
	4	0.722 ± 0.022	0.722 ± 0.023
	5	0.702 ± 0.057	0.692 ± 0.060
Swan	2	0.903 ± 0.019	0.902 ± 0.021

	3	0.789 ± 0.094	0.745 ± 0.092
	4	0.741 ± 0.063	0.737 ± 0.063
	5	0.719 ± 0.072	0.714 ± 0.070
Statues	2	0.700 ± 0.059	0.690 ± 0.055
	3	0.687 ± 0.010	0.678 ± 0.010
	4	0.663 ± 0.006	0.664 ± 0.004
	5	0.659 ± 0.012	0.659 ± 0.012
Star-fish	2	0.743 ± 0.037	0.743 ± 0.037
	3	0.671 ± 0.036	0.671 ± 0.036
	4	0.668 ± 0.037	0.661 ± 0.038
	5	0.651 ± 0.024	0.646 ± 0.024
Horses	2	0.673 ± 0.037	0.675 ± 0.037
	3	0.662 ± 0.037	0.661 ± 0.037
	4	0.582 ± 0.019	0.588 ± 0.020
	5	0.557 ± 0.023	0.556 ± 0.024

The average silhouette values obtained by the clustering techniques range from 0.50 to 0.90. This implies that the clustering solutions obtained in this study are appropriate.

Figure 5.5 shows that both the Kmeans and Kmedoids clustering techniques follow a similar trend, where the lowest average silhouette value is attained when $K=2$ and the maximum value is obtained at $K=3$. Thus, the optimal number of clusters for the Butterfly image is 3 for both clustering techniques.

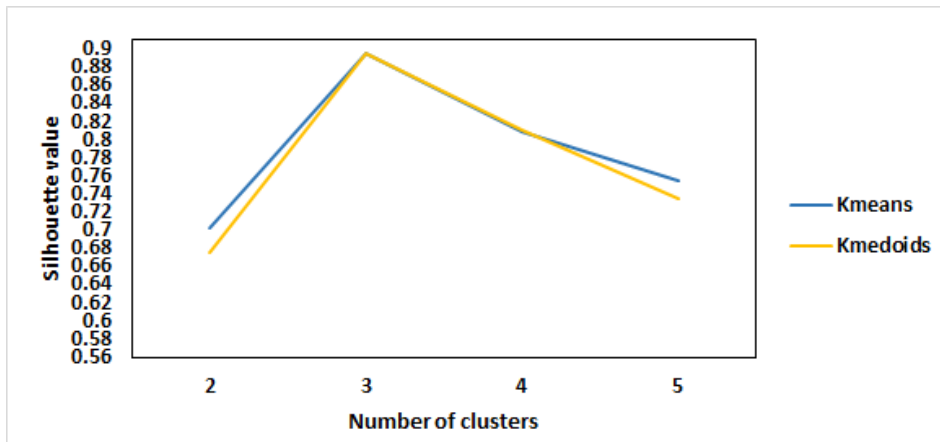


Fig. 5.5: Silhouette validity index versus Number of clusters for the Butterfly image

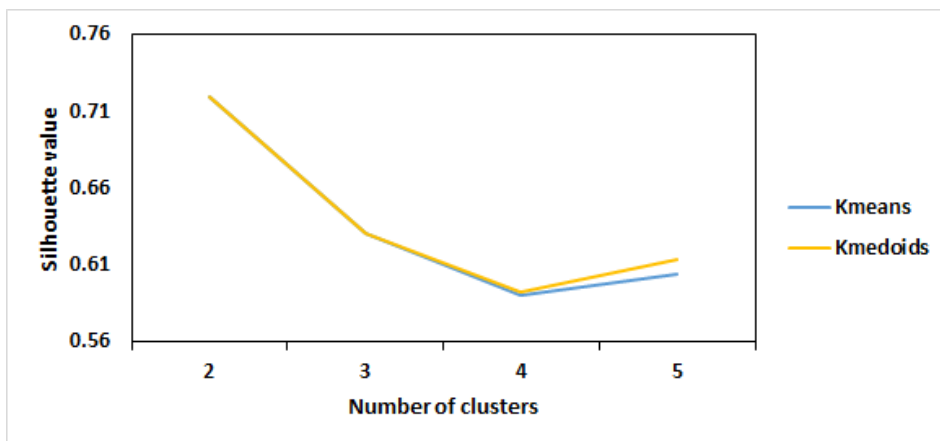


Fig. 5.6: Silhouette validity index versus Number of clusters for the Insect image

The average silhouette values obtained for the Insect image by the Kmeans and Kmedoids clustering techniques suggested that the best clustering solution is observed when $K=2$ as the maximum silhouette values of 0.720 and 0.719 were obtained for the Kmeans and Kmedoids clustering techniques, respectively.

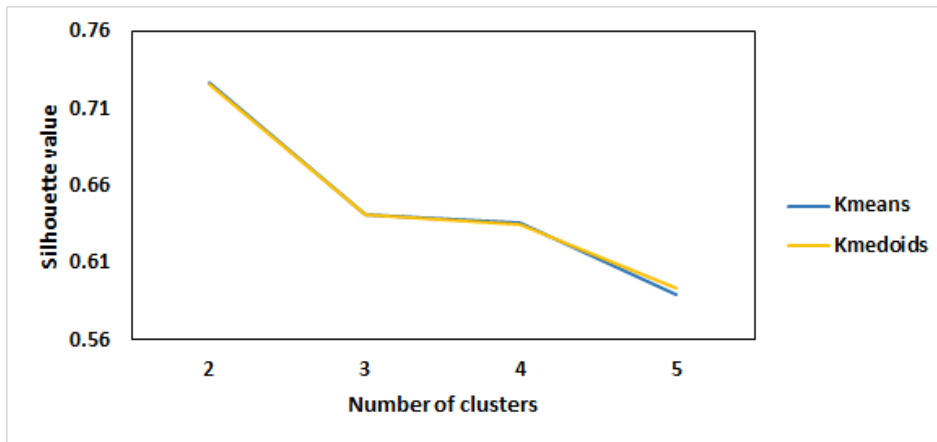


Fig. 5.7: Silhouette validity index versus Number of clusters for the Soldier image

The Kmeans and Kmedoids clustering techniques obtained similar silhouette values for all the number of clusters the experiments were run for. The highest and lowest silhouette values were obtained when $K=2$ and 5 for both the clustering techniques. The Kmeans technique obtained the lowest average silhouette value of 0.587 and highest average silhouette value of 0.727 , while the Kmedoids technique obtained the lowest average silhouette value of 0.588 and the average silhouette value of 0.726 . Thus, the average silhouette value obtained indicates that the optimal number of clusters for the Soldier image is 2 .

Figure 5.8 shows that the lowest average silhouette values obtained by the Kmeans and Kmedoids clustering techniques were 0.664 and 0.665 , respectively. The clustering techniques obtained an equivalent highest average silhouette value of 0.867 each. The Kmeans and Kmedoids clustering techniques obtained the highest average silhouette value when the number of clusters, K , was 2 .

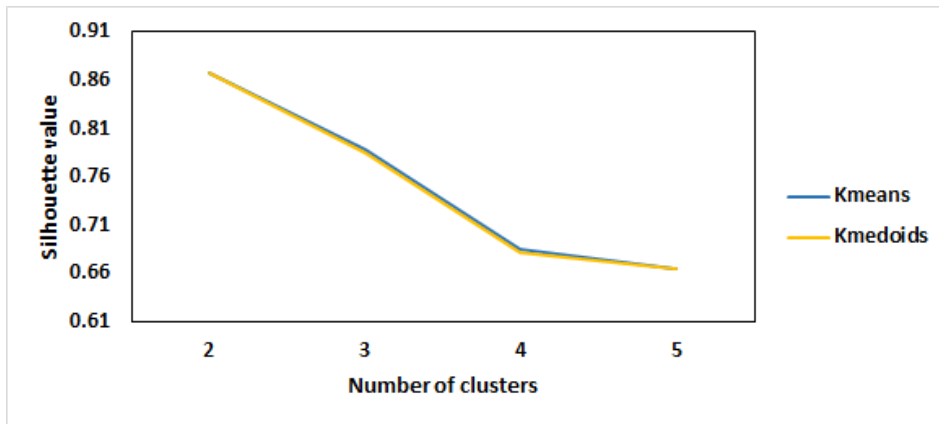


Fig. 5.8: Silhouette validity index versus Number of clusters for the Kids image



Fig. 5.9: Silhouette validity index versus Number of clusters for the Swimmer image

It was observed from Figure 5.9 that both clustering techniques obtained lowest and highest average silhouette values when $K=3$ and $K=2$ for the Swimmer image. This suggests that the suitable number of clusters that obtained the best clustering solution for the image is 2.

The Kmeans and Kmedoids clustering techniques obtained their highest average silhouette values of 0.765 and 0.760, respectively when $K=2$, while the lowest average silhouette values of 0.711 and 0.714 when $K=2$ and $K=5$



Fig. 5.10: Silhouette validity index versus Number of clusters for the Car image

were obtained for the Kmeans and Kmedoids techniques, respectively. In addition, Figure 5.10 shows that the Kmedoids clustering technique found better clustering solutions than the Kmeans technique for all the other number of clusters except when $K=2$ and $K=5$

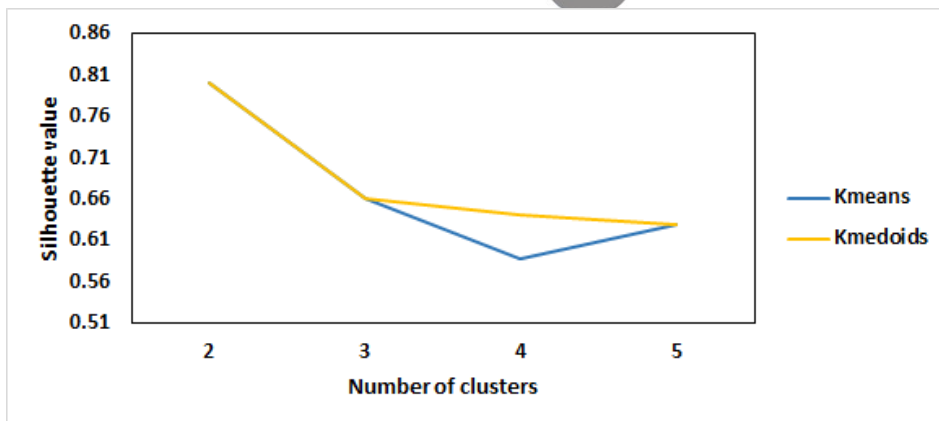


Fig. 5.11: Silhouette validity index versus Number of clusters for the Aeroplane image

Figure 5.11 shows that the Kmedoids clustering technique performed better than the Kmeans clustering technique as the average silhouette values obtained by the Kmedoids technique are greater than or equal those obtained

by the Kmeans technique. In addition, both clustering techniques obtained an equivalent highest average silhouette value of 0.800 when $K=2$. Although this is the case, the Kmeans clustering technique obtained its lowest average silhouette value of 0.587 when $K=4$. On the other hand, the Kmedoids clustering technique obtained its lowest average silhouette value of 0.629 when $K=5$.



University of Fort Hare
Together in Excellence
 Fig. 5.12: Silhouette validity index versus Number of clusters for the Flowers image

It was observed from Figure 5.12, that the Kmeans clustering technique found a more suitable clustering solution than the Kmedoids technique. In addition, both clustering techniques arrived at the best and worst clustering solutions when $K=2$ and $K=5$, respectively. Furthermore, their highest average silhouette value was 0.814 which implies that they both found a strong clustering structure.

The average silhouette values obtained for the Swan image by the Kmeans and Kmedoids clustering techniques suggested that the best clustering solution is observed when $K=2$ as the highest average silhouette values of 0.903 and 0.902 were obtained for the Kmeans and Kmedoids clustering techniques,

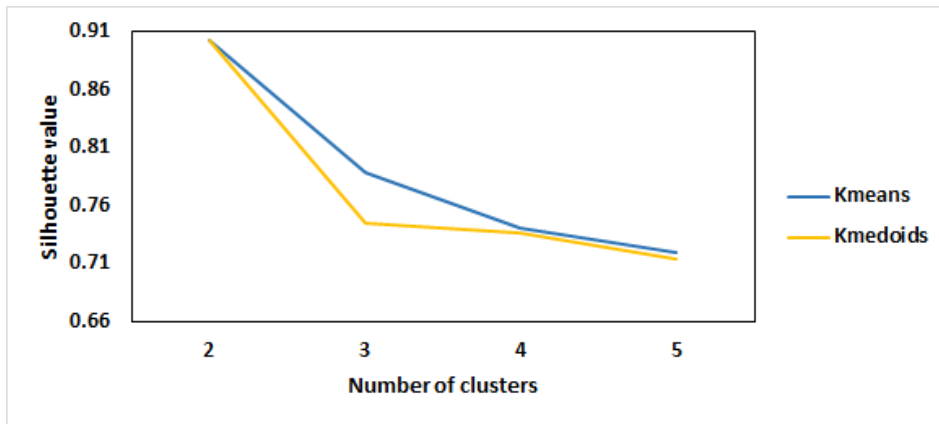


Fig. 5.13: Silhouette validity index versus Number of clusters for the Swan image respectively. In addition, both clustering techniques obtained their lowest average silhouette values when $K=5$. The Kmeans clustering technique is seen to perform better than the Kmedoids technique as all of its average silhouette values are greater than those of the Kmedoids clustering technique.

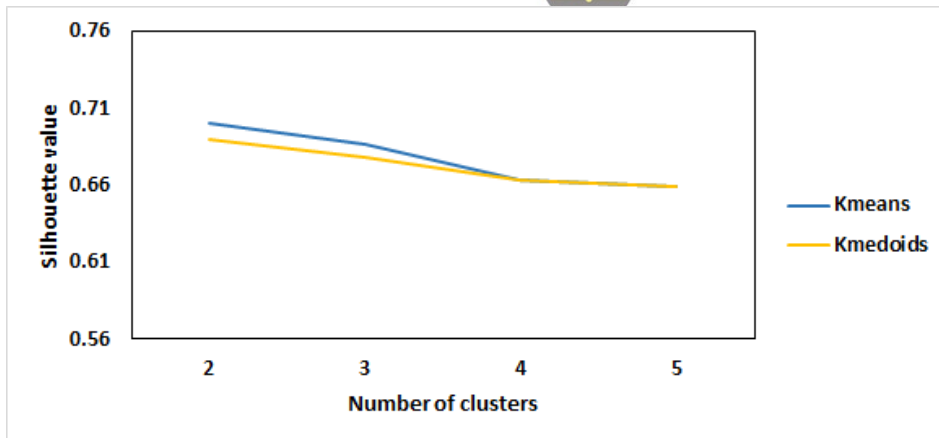


Fig. 5.14: Silhouette validity index versus Number of clusters for the Statues image

The Kmeans and kmediods clustering techniques obtained their highest average silhouette values of 0.700 and 0.690, respectively when $K=2$. While the lowest average silhouette values of 0.663 and 0.664 for the Kmeans and Kmedoids techniques, respectively when $K=4$. In addition, Figure 5.14 shows

that the Kmeans clustering technique found better clustering solutions than the Kmedoids technique for all the other number of clusters except for when $K=4$ and $K=5$.

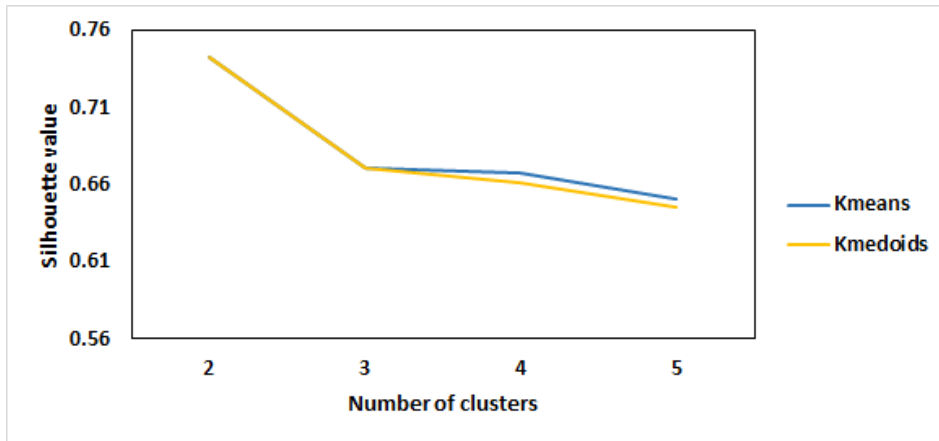


Fig. 5.15: Silhouette validity index versus Number of clusters for the Star-fish image



Together in Excellence

It is observed from Figure 5.15 that the Kmeans and Kmedoids clustering techniques obtained equivalent highest average silhouette values of 0.743 when $K=2$. This implies that the both clustering techniques found strong clustering structures for the Star-fish image. The techniques both obtained their lowest average silhouette values of 0.651 for Kmeans and 0.646 for Kmedoids clustering technique when $K=5$.

It is observed from Figure 5.16 that the Kmedoids clustering technique found more suitable clustering solutions than the Kmeans technique as it obtained higher average silhouette values. In addition, both clustering techniques arrived at the best and worst clustering solutions when $K=2$ and $K=5$, respectively. Their highest average silhouette values were 0.673 and 0.675 for the Kmeans and Kmedoids clustering technique, respectively.

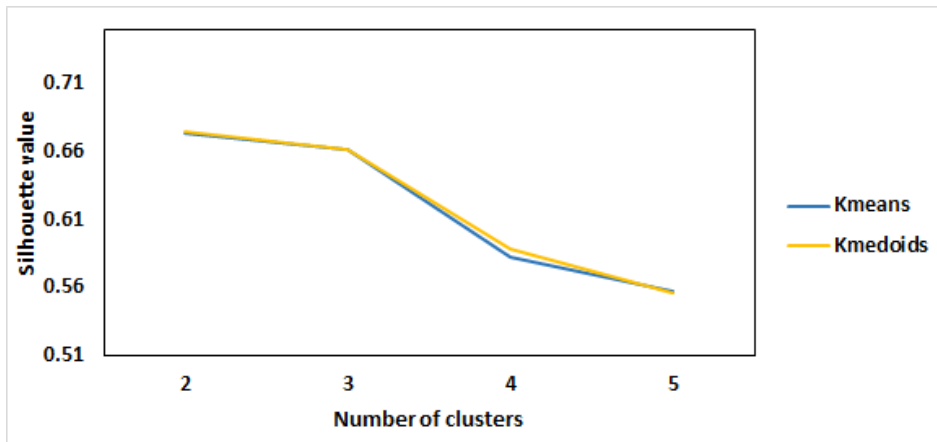


Fig. 5.16: Silhouette validity index versus Number of clusters for the Horses image

It was observed from the average silhouette values obtained by the Kmeans and Kmedoids clustering techniques for the test images that all the images obtained their best clustering solutions when $K=2$. The exception is the Butterfly image whose highest Silhouette value is acquired when $K=3$. In addition, it was seen that the lowest average silhouette values were obtained when $K=5$ for most images. The Swan image obtained the highest average silhouette value for both the Kmeans and Kmedoids clustering techniques amongst all the other images. On the other hand, the Horses image obtained the lowest average silhouette values for both clustering techniques amongst the test images. Furthermore, all the average silhouette values obtained are above 0.5. This implies that reasonable to strong clustering structure were found when the Kmeans and Kmedoids clustering techniques were used on the test images.

The Kmeans and Kmedoids clustering techniques are executed as part of the model used to generate both background and foreground GMMs as

described in the Initialisation stage provided in the Grabcuts algorithm summary in Chapter 3. Thus the researchers measured the runtimes for creation of GMMs and assignment of pixels to both the foreground and background GMMs based on Kmeans (GMM-Kmeans) and those based on Kmedoids (GMM-Kmedoids). Figure 5.17 shows the average runtimes used to generate the background and foreground GMMs for GMM-Kmeans as compared to the ones generated by GMM-Kmedoids.

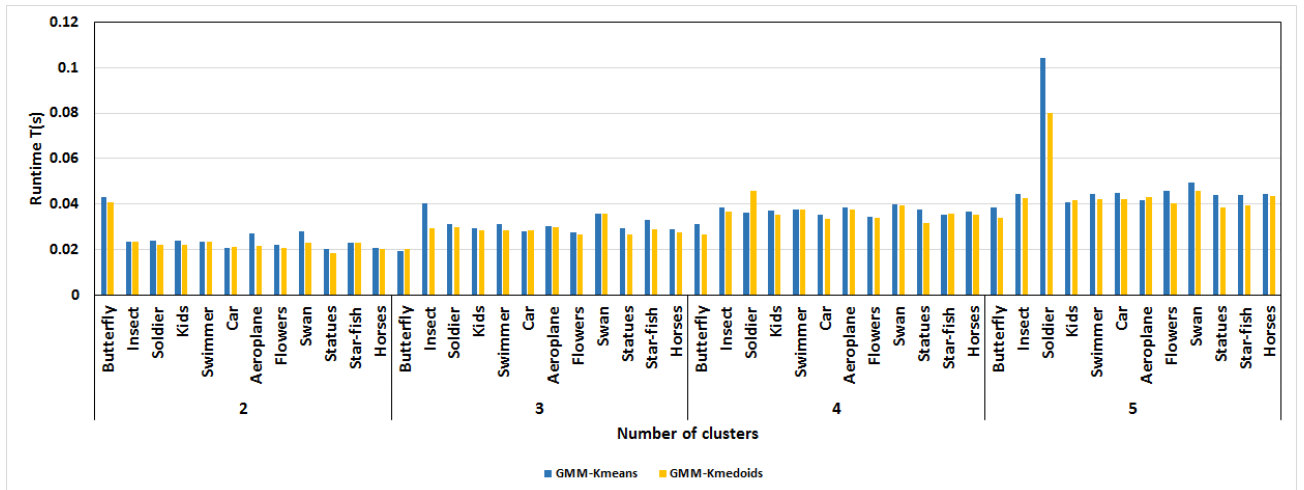


Fig. 5.17: Average runtime GMM-Kmeans versus GMM-Kmedoids

Figure 5.17 shows that although the Kmeans clustering technique outperformed the Kmedoids in terms of runtimes, the time taken to generate the GMMs by GMM-Kmedoids is comparable smaller than the time taken by GMM-Kmeans for most of the images. It is also observed that in some instances the GMM-Kmeans performed equivalently to the GMM-Kmedoids, for instance, for the Insect, Swimmer and Star-fish images when $K=2$. In addition, the results obtained show an upward trend between the average runtime and number of clusters in all images for both the GMM-Kmeans and

GMM-Kmedoids. This implies that as the number of clusters is increased, the more computational time is required by the algorithms to generate GMMs. This is due to the more initial points being randomly selected as centers or medians by the Kmeans or Kmedoids algorithms as described in the steps followed by the two clustering techniques. Furthermore, the runtimes range from 0,020-0,104 seconds for GMM-Kmeans and 0.019-0.0801 seconds for GMM-Kmedoids, respectively.

5.2.3 Segmentation results

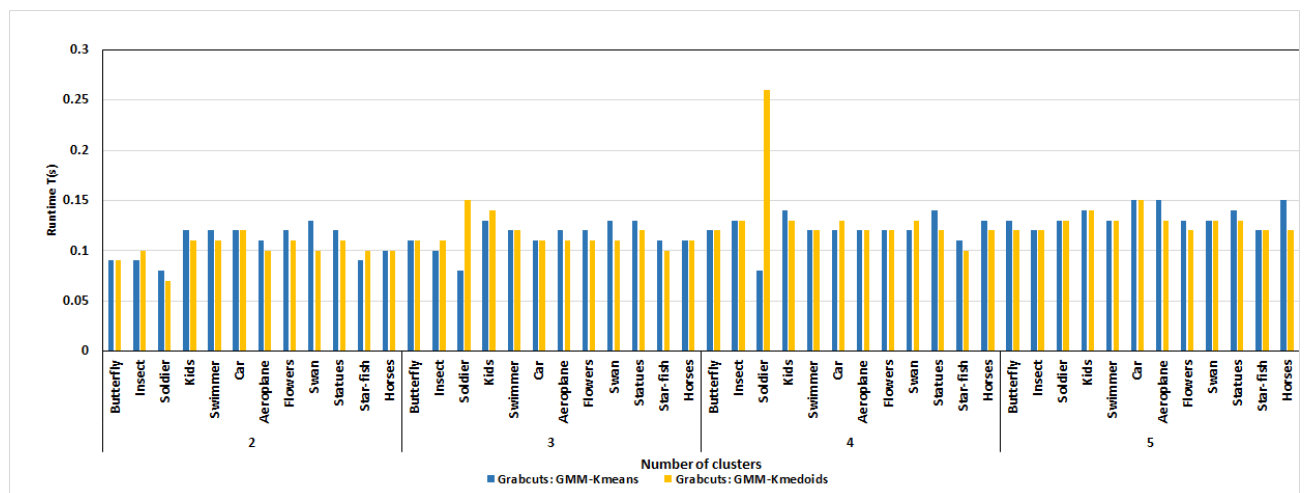


Fig. 5.18: Average runtimes for Grabcuts for image segmentation algorithm: GMM-Kmeans versus GMM-Kmedoids

Figure 5.18 shows that the Grabcuts for segmentation algorithm with the GMM-Kmedoids clustering heuristic outperformed its counterpart. The Grabcuts for image segmentation algorithm with the GMM-Kmeans clustering heuristic obtained the lowest average runtime of 0.08 seconds when K was varied from 2 to 4, whereas it obtained a highest average runtime of 0.12 seconds when K=5. On the other hand, it obtained highest runtimes of 0.13

seconds for $K=2$ and 3, 0.14 seconds for $K=4$ and 0.15 seconds for $K=5$. The Soldier image was observed to require the least running time when K was varied from 2 to 3, while the Car, Aeroplane and Horses images required the highest runtimes amongst all the other images when $K=5$. Most test images obtained runtimes of 0.12 when $K=2$ and 4 for the Grabcuts for image segmentation algorithm with the GMM-Kmeans clustering heuristic. The runtimes of 0.11 and 0.13 seconds were obtained by most test images when $K=3$ and $K=5$, respectively.

The Grabcuts for image segmentation algorithm with the GMM-Kmedoids clustering heuristic obtained the lowest average runtime of 0.07 seconds when K was varied from 2 to 5 for all test images. A highest average runtime 0.26 seconds was obtained for this algorithm across all test images with K varied. The lowest and highest runtimes were obtained when the algorithm was executed with the Soldier image with $K=2$ and $K=4$, respectively. In addition, most of the test images required runtimes of 0.10, 0.11, 0.12 and 0.13 when $K=2, 3, 4$ and 5 respectively.

The segmentation results of the Grabcuts for segmentation algorithm are presented in Figure 5.36. The segmentation results for the Grabcuts for image segmentation with the GMM-Kmeans and GMM-Kmedoids clustering heuristic are presented in two consecutive rows for each test image. The first row shows results for the Grabcuts for image segmentation with the GMM-Kmeans clustering heuristic and the second row shows results for the Grabcuts for image segmentation with the GMM-Kmedoids clustering heuristic. The number of clusters increases from left to right, which implies the left most and right most results are obtained when $K=2$ and $K=5$, respectively.

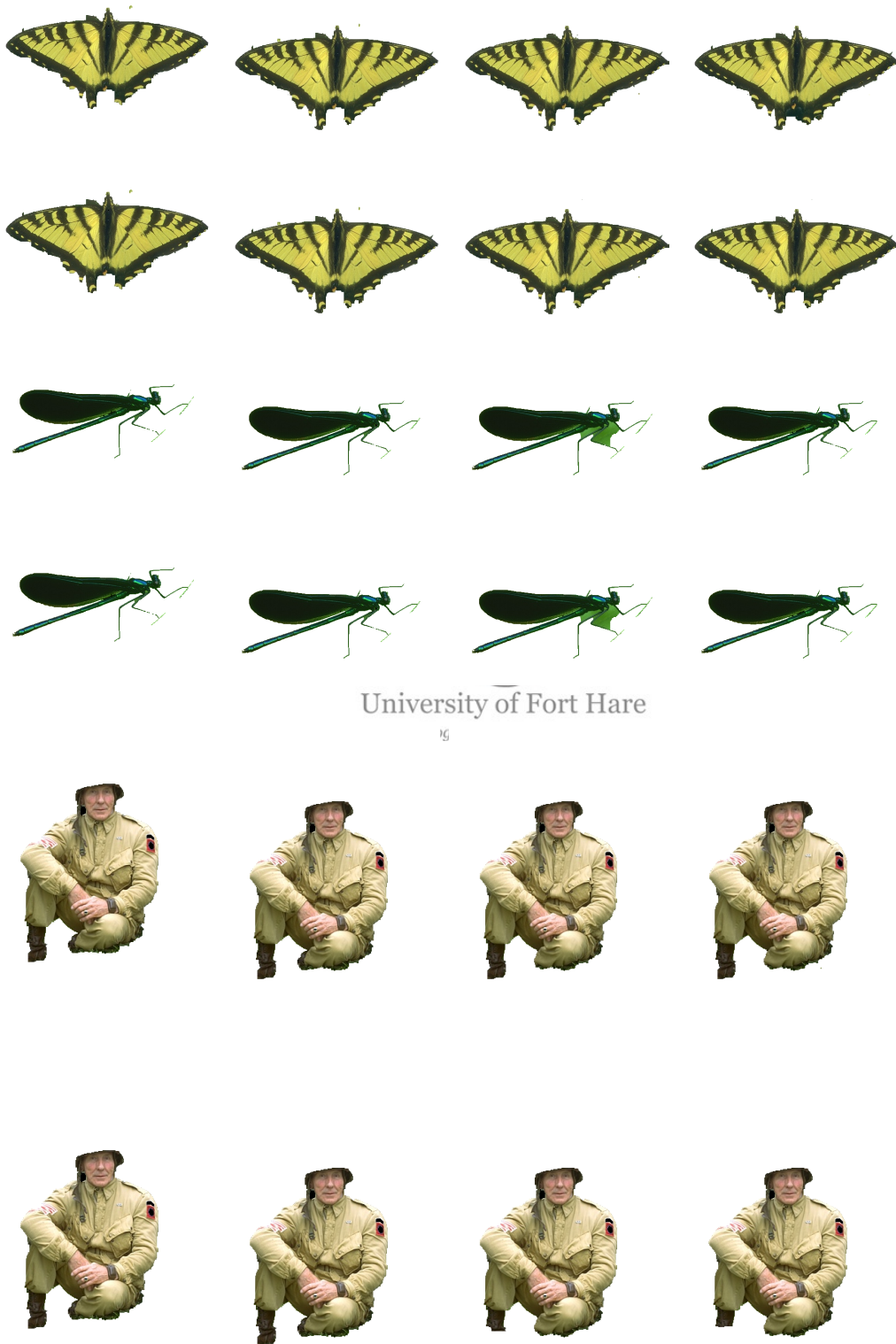


Fig. 5.19: Segmentation results for Grabcuts for image segmentation: GMM-Kmeans versus GMM-Kmedoids



Fig. 5.19: Segmentation results for Grabcuts for image segmentation: GMM-Kmeans versus GMM-Kmedoids



Fig. 5.19: Segmentation results for Grabcuts for image segmentation: GMM-Kmeans versus GMM-Kmedoids



Fig. 5.19: Segmentation results for Grabcuts for image segmentation: GMM-Kmeans versus GMM-Kmedoids

The Grabcuts for image segmentation algorithm based on the GMM-Kmeans and GMM-Kmedoids clustering heuristic obtained segmentation results in which some pixels that were supposed to be in the foreground were classified as background. This was accredited to the low contrast regions seen between the transition areas from background and foreground. The algorithms obtained similar segmentation results when K was varied from 2 to 4 for the Butterfly image. This was not the case when $K=5$, as the Grabcuts for image segmentation algorithm based on the GMM-Kmedoids obtained better segmentation results than its counterpart.

The Grabcuts for image segmentation algorithm based on the GMM-Kmeans and GMM-Kmedoids clustering heuristic obtained similar segmentation results for the Insect image. The misclassification of some background pixels around the body of the insect was observed for all the segmentation results obtained. The algorithms performed worst when $K=4$, as some of the leaf that was supposed to be classified as background ended up being part of the foreground.

The soldier's helmet and parts of his boots were taken to mistakenly belong to the background on all segmentation results obtained. This is due to the overlap in colour space of the foreground and background colour distributions [48]. The Grabcuts for image segmentation algorithm based on the GMM-Kmeans obtained better results as more foreground pixels were misclassified by the Grabcuts for image segmentation algorithm based on the GMM-Kmedoids cluster heuristic, for instance, segmentation results obtained when $K=4$.

The Grabcuts for image segmentation algorithm based on the GMM-Kmeans and GMM-Kmedoids clustering heuristic obtained similar segmentation results for the Kids image. The results showed misclassified background pixels on top of the second boy's clay pot.

The Swimmer image obtained segmentation results in which more background pixels around the swimmer were taken to belong to the set of foreground pixels. This is as a result of the selected ROI's background material not being competently represented by the set of pixels belonging to the background region. The Grabcuts for image segmentation algorithm based on the GMM-Kmeans and GMM-Kmedoids clustering heuristic obtained their best segmentation results when $K=5$.



The tyres and windows of the car were mistaken to belong to the set of background pixels. This was as a result of them sharing a similar colour distribution with the background pixels. In addition, some of the background pixels were observed to have been misclassified as seen around the transition areas from background to foreground regions, around the car. For the Car image, the Grabcuts for image segmentation algorithm with a GMM based on the Kmeans clustering technique obtained better segmentation results than the algorithm with a GMM based on the Kmedoids clustering technique.

The Aeroplane image obtained similar segmentation results for all algorithms. This result is with a few exceptions where the Grabcuts for image segmentation algorithm with a GMM based on the Kmeans clustering technique obtained better segmentation results when $K=2,3$ and 5 . On the other hand, the Grabcuts for image segmentation algorithm with a GMM based on

the Kmedoids clustering technique obtained is better when $K=4$. It was also observed that all algorithms considered got their worst segmentation results when $K=5$.

The Grabcuts for image segmentation algorithm with a GMM based on the Kmeans and Kmedoids clustering techniques obtained identical segmentation results when K was varied from 2 to 4 for the Flowers image. The segmentation results showed a misclassification of background pixels around the flowers which share similar colour distribution as some of the pixels in the foreground region of the flowers. In addition, the Grabcuts for image segmentation algorithm with a GMM based on the Kmedoids clustering technique obtained better segmentation results, than its counterpart when $K=5$.

The Swan image obtained the best segmentation results when $K=2,3$ and 5 for the Grabcuts for image segmentation algorithm with a GMM based on the Kmeans clustering technique. On the other hand, it obtained the best segmentation result only when $K=5$ for the Grabcuts for image segmentation algorithm with a GMM based on the Kmedoids clustering technique.

The Grabcuts for image segmentation algorithm with a GMM based on the Kmedoids clustering technique obtained better segmentation results when $K=2,3$ and 4, than its counterpart for the Statues image. On the other hand, the Grabcuts for image segmentation algorithm with a GMM based on the Kmeans clustering technique obtained better segmentation results when $K=5$, than the Grabcuts for image segmentation algorithm with a GMM based on the Kmedoids clustering technique.

The Star-fish image segmentation results obtained for the Grabcuts for image segmentation algorithm with a GMM based on the Kmeans and Kmedoids clustering techniques are identical. In addition, the segmentation results obtained by this image for all algorithms considered with the variation of K, showed minimal pixel misclassification amongst all the test images considered.

The segmentation results obtained by the Horses image are identical for all algorithms considered, where background pixels are seen to be misclassified around the tail of the horse. This misclassification of pixels is accredited to the overlap of colour distributions around hairy objects.

The precision, recall and BF-score, with $\theta = 0.5$, of the segmentation results obtained from running the Grabcuts image segmentation algorithms: GMM-Kmeans and GMM-Kmedoids clustering techniques that utilised the Squared Euclidean distance are presented in this section.

Tab. 5.5: Average Precision and recall values: Grabcuts: GMM-Kmeans versus Grabcuts: GMM-Kmedoids

Image	K	Grabcuts: GMM-Kmeans		Grabcuts: GMM-Kmedoids	
		Precision	Recall	Precision	Recall
Butterfly	2	0.824	0.588	0.824	0.588
	3	0.823	0.587	0.823	0.586
	4	0.824	0.621	0.824	0.624
	5	0.825	0.625	0.825	0.651
Insect	2	0.496	0.805	0.501	0.814
	3	0.483	0.804	0.483	0.804

	4	0.484	0.802	0.484	0.802
	5	0.495	0.802	0.493	0.805
Soldier	2	0.691	0.949	0.691	0.949
	3	0.692	0.949	0.692	0.949
	4	0.691	0.949	0.691	0.949
	5	0.691	0.949	0.691	0.949
Kids	2	0.790	0.830	0.790	0.858
	3	0.789	0.634	0.790	0.691
	4	0.789	0.634	0.789	0.634
	5	0.789	0.634	0.791	0.832
Swimmer	2	0.697	0.580	0.712	0.611
	3	0.689	0.364	0.689	0.364
	4	0.699	0.584	0.698	0.586
	5	0.699	0.638	0.699	0.638
Car	2	0.667	0.683	0.650	0.683
	3	0.659	0.656	0.659	0.656
	4	0.658	0.683	0.652	0.683
	5	0.662	0.683	0.664	0.683
Aeroplane	2	0.731	0.669	0.700	0.638
	3	0.724	0.669	0.695	0.638
	4	0.730	0.698	0.691	0.669
	5	0.724	0.770	0.692	0.722
Flower	2	0.711	0.602	0.709	0.466
	3	0.712	0.602	0.712	0.602

	4	0.709	0.557	0.709	0.466
	5	0.729	0.939	0.731	0.941
Swan	2	0.475	0.630	0.473	0.630
	3	0.475	0.630	0.473	0.634
	4	0.473	0.630	0.473	0.630
	5	0.475	0.623	0.475	0.630
Statues	2	1.00	0.707	1.00	0.707
	3	1.00	0.831	1.00	0.707
	4	1.00	0.707	1.00	0.707
	5	1.00	0.848	1.00	0.848
Star-fish	2	0.892	0.591	0.892	0.591
	3	0.892	0.591	0.892	0.591
	4	0.892	0.591	0.892	0.591
	5	0.892	0.591	0.892	0.591
Horses	2	0.707	0.579	0.707	0.579
	3	0.707	0.579	0.707	0.579
	4	0.707	0.579	0.707	0.579
	5	0.707	0.579	0.707	0.579

Table 5.5 shows that the segmented Butterfly images obtained similar precision values for both Grabcuts: GMM-Kmeans and Grabcuts: GMM-Kmedioids. On the other hand, the recall values obtained when the Grabcuts: GMM-Kmedioids was used for the Butterfly image were slightly higher than those obtained by its counterpart when K=4 and 5.

The Grabcuts: GMM-Kmedioids was observed to have obtained a segmented Insect image with higher precision and recall values than the Grabcuts: GMM-Kmeans when $K=2$. On the other hand, when $K=5$ a higher precision value is obtained by the Grabcuts: GMM-Kmeans segmented Insect image, while the Grabcuts: GMM-Kmedioids achieves a higher recall value.

The segmented Soldier image obtained similar precision and recall values for both segmentation algorithms considered. The recall values were observed to be much higher than the precision values when K was varied for both the Grabcuts: GMM-Kmeans and Grabcuts: GMM-Kmedioids segmentation results. This implies that the segmented Soldier images suffered from over segmentation, which is as a result of misclassification of some foreground pixels as background pixels.



The precision values obtained from the segmented Kids images are higher than the recall values when K is varied from 3 to 5 for Grabcuts: GMM-Kmeans and when K is varied from 3 to 4 for Grabcuts: GMM-Kmedioids. In addition, both image segmentation algorithms obtain segmentation results with highest precision and recall values when $K=2$.

The segmented Swimmer image obtained by the Grabcuts: GMM-Kmeans achieved its highest precision and recall values when $K=5$. On the other hand, it was observed that the segmented Swimmer image obtained by the Grabcuts: GMM-Kmedioids achieved its highest precision and recall values when $K=2$ and $K=5$, respectively. The segmentation algorithms considered obtained similar precision and recall values when $K=3$ and 5.

The segmented Car images were observed to have obtained similar recall values for both segmentation algorithms considered under scenario 1 when K is varied from 2 to 5. Although this was the case, the segmentation algorithms obtained a similar precision value only when K=3. In addition, it was observed that the recall values obtained for the segmented Car images were higher than the precision values. This implies that the Car images were over segmented.

The highest precision and recall values for the segmented Aeroplane images for both segmentation algorithms considered under scenario 1 was obtained when K=2 and K=5, respectively. Although this was the case, the Grabcuts: GMM-Kmeans obtained segmented images that had slightly higher precision and recall values than its counterpart.



The segmented Flower images obtained their highest precision and recall values for the Grabcuts: GMM-Kmeans and Grabcuts: GMM-Kmediods when K= 5. In addition, when K= 3 similar precision and recall values were obtained for the segmented images for both segmentation algorithms.

The segmented Swan images obtained by the Grabcuts: GMM-Kmeans achieved their highest precision value when K=2, 3 and 5. The highest recall value was achieved when K=2,3 and 4. On the other hand, the segmented Swan images obtained by the Grabcuts: GMM-Kmediods achieved their highest precision and recall values when K=5 and K=3.

The segmented Statues images obtained similar precision and recall values when K was varied for 2 to 5 under both segmentation algorithms considered.

This is with the exception of the segmented images obtained when $K=3$ for both segmentation algorithms, which achieved different recall values of 0.831 and 0.707.

The segmented images for the Star-fish obtained similar precision and recall values when K was varied from 2 to 5 for both segmentation algorithm considered. The same trend was observed from Table 5.5 for the segmented Horses images. In addition, it was observed that the precision values were much larger than the recall values. This implies that these segmented images suffer from under segmentation.

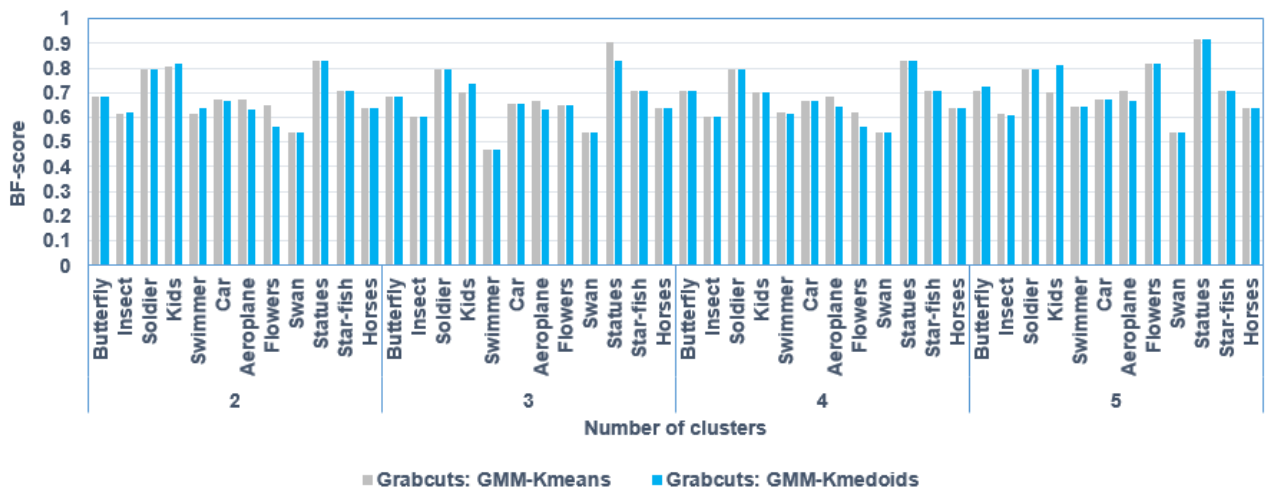


Fig. 5.20: Average BF-scores for Grabcuts for image segmentation algorithm: GMM-Kmeans versus GMM-Kmedoids

Figure 5.20 shows the BF-scores for the segmented images. All these results indicate that the quantitative evaluation measure chosen matches the relative visual quality well. The segmented Statues images is observed to have the highest BF-scores when K is varied for both the Grabcuts: GMM-Kmeans as well as the Grabcuts: GMM-Kmedoids. On the other hand, the

segmented Swan images obtain the lowest scores when K is varied for both the Grabcuts: GMM-Kmeans and Grabcuts: GMM-Kmedoids, except for when K=3. Figure 5.20 shows that the Grabcuts: GMM-Kmedoids produces images with slightly higher BF-scores than its counterpart when K was varied from 2 to 5 for most of the test images.

5.2.4 Performance of Grabcuts for image segmentation algorithm under Scenario 2

The researchers in this section present the results obtained after the execution of the Grabcuts image segmentation algorithms with a GMM based on Kmeans and Kmedoids clustering techniques that utilised the City block distance. These algorithms were implemented on the images provided in Figure 5.1 and the number of clusters, K, was varied from 2 to 5. The results of the average runtimes obtained by the Kmeans and Kmedoids clustering techniques are shown in Table 5.6.

Tab. 5.6: Average runtime: Kmeans versus Kmedoids

Image	K	Average runtime in seconds (s)	
		Kmeans	Kmedoids
Butterfly	2	0.0529	1.55
	3	0.116	2.67
	4	0.221	3.39
	5	0.254	4.65
Insect	2	0.131	1.61
	3	0.207	3.17

	4	0.162	2.82
	5	0.218	2.75
Soldier	2	0.136	2.08
	3	0.181	2.80
	4	0.230	4.05
	5	0.224	4.38
Kids	2	0.0868	2.22
	3	0.187	2.63
	4	0.218	3.18
	5	0.296	3.47
Swimmer	2	0.0887	1.39
	3	0.163	2.56
	4	0.212	3.09
	5	0.204	3.43
Car	2	0.0955	0.723
	3	0.146	1.26
	4	0.207	1.67
	5	0.231	2.22
Aeroplane	2	0.0729	1.33
	3	0.124	1.72
	4	0.174	1.94
	5	0.115	2.76
Flower	2	0.0973	0.863
	3	0.156	1.96



University of Fort Hare
Together in Excellence

	4	0.165	2.37
	5	0.248	3.45
Swan	2	0.0710	1.15
	3	0.121	1.13
	4	0.139	2.16
	5	0.158	2.36
Statues	2	0.152	1.36
	3	0.217	2.23
	4	0.232	3.09
	5	0.248	3.91
Star-fish	2	0.111	2.00
	3	0.155	2.74
	4	0.234	3.57
	5	0.283	4.79
Horses	2	0.167	2.16
	3	0.199	3.44
	4	0.203	3.41
	5	0.262	4.26

Table 5.6 shows that the Kmeans clustering technique outperforms the Kmedoids techniques for all the images. In addition, the Kmeans clustering technique achieved the lowest and highest average runtimes of 0.0529 and 0.269 seconds, respectively. On the other hand, the Kmedoids clustering technique achieved the lowest and highest average runtimes of 1.13 and 4.79 seconds. The highest average runtimes for both clustering techniques were

achieved when the Grabcuts for image segmentation algorithms was executed on the Kids and Star-fish images with $K=5$ for the Kmeans and Kmedoids clustering techniques, respectively. In addition, most images achieved lowest and highest average runtimes when $K=2$ and $K=5$ for both the Kmeans and Kmedoids clustering techniques. The average number of iterations taken by the Kmeans and Kmedoids clustering techniques to reach convergence are shown in Figure 5.21. These are the average number of iterations required in order for the algorithms to create K Gaussian components which are required for the assignment of both foreground and background pixels to an appropriate GMM. The Kmeans clustering technique on average required

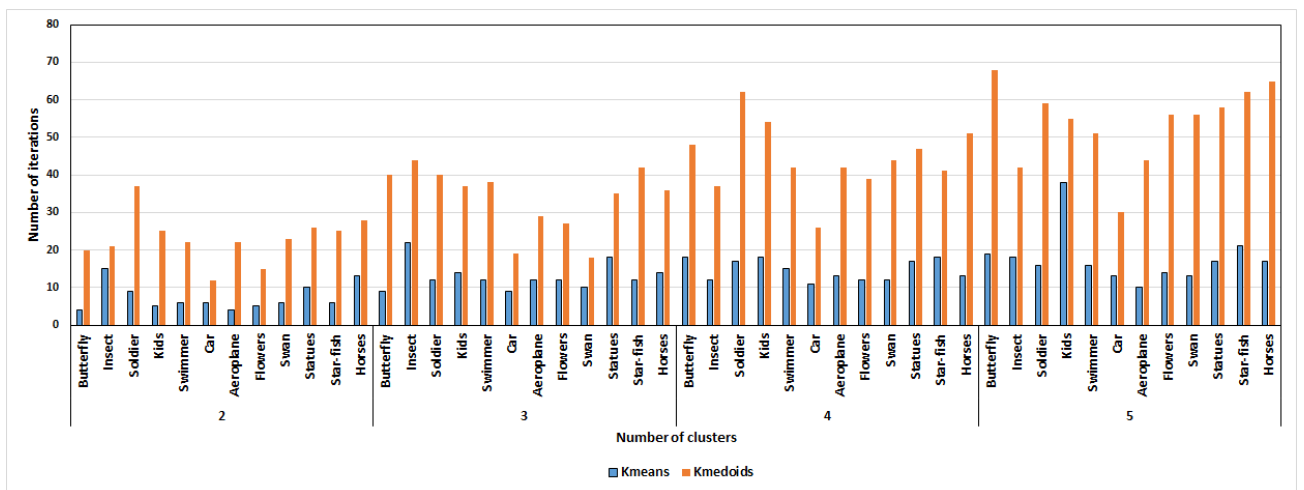


Fig. 5.21: Number of iterations: Kmeans versus Kmedoids clustering technique

less iterations to reach convergence than the Kmedoids clustering technique. The lowest and highest observed number of iterations required by the Kmeans clustering technique to reach convergence were 4 and 38 iterations, respectively. On the other hand, the Kmedoids technique managed to converge at the lowest and highest number of iterations of 12 and 68 iterations, respec-

tively. In addition, the lowest number of iterations is achieved when $K=2$ for both clustering techniques in all images. Furthermore, when $K=5$ most images required the highest number of iterations.

The compactness of the clustering solution obtained by the Kmeans and Kmedoids techniques was measured using the intra-cluster distance. Table 5.7 presents a comparison of the average intra-cluster distance and the standard deviations obtained by the Kmeans and Kmedoids clustering techniques.

Tab. 5.7: Averages and Standard deviations of Intra-cluster distance

Image	K	Average intra-cluster distance (1×10^6)	
		Kmeans	Kmedoids
Butterfly	2	4.428 ± 2.255	4.432 ± 2.260
	3	3.333 ± 1.498	3.312 ± 1.536
	4	2.786 ± 1.241	2.729 ± 1.269
	5	2.416 ± 1.023	2.420 ± 1.029
Insect	2	54.59 ± 28.67	53.58 ± 26.39
	3	37.75 ± 22.61	35.81 ± 19.53
	4	1.818 ± 1.286	1.815 ± 1.285
	5	1.645 ± 1.109	1.640 ± 1.109
Soldier	2	4.948 ± 1.943	4.949 ± 1.945
	3	3.853 ± 1.697	3.855 ± 1.699
	4	3.257 ± 1.607	3.259 ± 1.609
	5	2.884 ± 1.510	2.869 ± 1.491

Kids	2	3.518 ± 1.392	3.544 ± 1.351
	3	2.628 ± 0.8999	2.586 ± 1.031
	4	2.085 ± 0.7815	2.084 ± 0.8152
	5	1.784 ± 0.6738	1.812 ± 0.6586
Swimmer	2	5.264 ± 3.556	5.238 ± 3.684
	3	4.193 ± 2.807	4.163 ± 2.792
	4	3.490 ± 2.383	3.421 ± 2.125
	5	2.950 ± 2.078	2.965 ± 1.891
Car	2	4.489 ± 1.192	4.424 ± 1.299
	3	3.155 ± 0.8706	3.107 ± 0.8783
	4	2.660 ± 0.9744	2.464 ± 0.6749
	5	2.179 ± 0.6490	2.101 ± 0.3234
Aeroplane	2	2.178 ± 0.2116	2.189 ± 0.2060
	3	1.703 ± 0.2365	1.666 ± 0.2112
	4	1.420 ± 0.1408	1.436 ± 0.1387
	5	1.264 ± 0.1540	1.263 ± 0.2099
Flowers	2	3.985 ± 0.8132	3.914 ± 0.7777
	3	2.929 ± 1.136	2.874 ± 0.9503
	4	2.380 ± 0.7788	2.263 ± 0.7417
	5	2.050 ± 0.4879	2.050 ± 0.5044
Swan	2	3.067 ± 1.094	3.076 ± 1.102
	3	2.131 ± 0.7834	2.222 ± 0.7471
	4	1.816 ± 0.7567	1.759 ± 0.7114
	5	1.480 ± 0.6320	1.494 ± 0.5853

Statues	2	5.005 ± 2.032	4.977 ± 1.980
	3	3.727 ± 1.344	3.714 ± 1.390
	4	2.880 ± 0.9251	2.922 ± 0.9124
	5	2.401 ± 0.8553	2.393 ± 0.8094
Star-fish	2	4.665 ± 1.764	4.666 ± 1.765
	3	3.415 ± 1.432	3.486 ± 1.390
	4	2.879 ± 1.185	2.860 ± 1.162
	5	2.469 ± 1.063	2.467 ± 1.045
Horses	2	4.085 ± 1.017	4.075 ± 0.9983
	3	3.187 ± 0.8655	3.188 ± 0.8667
	4	2.655 ± 0.7303	2.669 ± 0.7339
	5	2.348 ± 0.6521	2.360 ± 0.6540

University of Fort Hare

It is observed from Table 5.7 that the intra-cluster distance is a decreasing function of K for both clustering techniques. The Kmeans clustering technique obtained lowest and highest average intra-cluster distances of 1.264×10^6 and 54.59×10^6 , respectively. On the other hand, the Kmedoids clustering technique obtained lowest and highest average intra-cluster distances of 1.263×10^6 and 53.58×10^6 , respectively. In addition, both clustering techniques obtained their lowest and highest average intra-cluster distances when K=5 for the Aeroplane image and K=2 for the Insect image, respectively.

The separability of the clustering solution was measured using the inter-cluster distance. Table 5.8 shows the average inter-cluster distance and standard deviations for the clustering solution obtained by the Kmeans clus-

tering technique as compared to that obtained by the Kmedoids clustering technique.

Tab. 5.8: Averages and Standard deviations of Inter-cluster distance

Image	K	Inter-cluster distance (1×10^7)	
		Kmeans	Kmedoids
Butterfly	2	3.201 \pm 1.331	3.198 \pm 1.339
	3	4.686 \pm 1.928	4.685 \pm 1.974
	4	6.417 \pm 2.768	6.467 \pm 2.846
	5	8.299 \pm 3.177	8.374 \pm 3.236
Insect	2	98.00 \pm 51.56	99.45 \pm 51.96
	3	148.7 \pm 33.74	156.1 \pm 22.44
	4	2.189 \pm 1.607	2.141 \pm 1.189
	5	2.815 \pm 1.413	2.818 \pm 1.355
Soldier	2	1.954 \pm 0.5090	1.955 \pm 0.5108
	3	3.214 \pm 1.006	3.214 \pm 1.012
	4	4.560 \pm 1.603	4.565 \pm 1.589
	5	6.017 \pm 2.463	6.019 \pm 2.208
Kids	2	2.460 \pm 1.177	2.475 \pm 1.133
	3	3.631 \pm 1.536	3.522 \pm 1.713
	4	4.773 \pm 2.016	4.768 \pm 2.094
	5	5.949 \pm 2.328	6.130 \pm 2.746
Swimmer	2	2.411 \pm 1.745	2.401 \pm 1.791
	3	3.896 \pm 2.848	3.249 \pm 2.677

	4	5.446 ± 4.079	5.091 ± 3.618
	5	7.009 ± 5.358	7.023 ± 4.980
Car	2	2.195 ± 0.4970	2.168 ± 0.5065
	3	3.488 ± 0.8039	3.466 ± 0.7781
	4	4.807 ± 1.135	4.748 ± 0.9954
	5	5.904 ± 1.062	5.884 ± 0.8173
Aeroplane	2	1.579 ± 0.7658	1.579 ± 0.7517
	3	2.156 ± 0.8170	2.107 ± 0.9856
	4	2.676 ± 0.8998	2.727 ± 0.8537
	5	3.321 ± 0.8907	4.002 ± 1.862
Flowers	2	1.649 ± 0.3625	1.886 ± 0.3854
	3	2.920 ± 0.4559	2.995 ± 0.6421
	4	4.245 ± 0.6518	4.181 ± 0.5048
	5	5.323 ± 0.8184	5.126 ± 0.7896
Swan	2	2.399 ± 1.079	2.362 ± 1.059
	3	3.247 ± 1.416	3.439 ± 1.531
	4	4.602 ± 1.867	4.974 ± 2.069
	5	5.753 ± 2.148	6.442 ± 2.284
Statues	2	2.023 ± 1.130	1.982 ± 1.062
	3	3.514 ± 1.378	3.408 ± 1.392
	4	5.280 ± 1.772	5.910 ± 1.469
	5	7.427 ± 2.312	7.376 ± 2.081
Star-fish	2	2.128 ± 0.6895	2.129 ± 0.6926
	3	3.302 ± 1.156	3.364 ± 1.136

	4	4.540 ± 1.528	4.448 ± 1.505
	5	5.792 ± 2.299	5.674 ± 2.118
Horses	2	1.409 ± 0.3310	1.404 ± 0.3136
	3	2.683 ± 0.7336	2.689 ± 0.7374
	4	3.689 ± 0.8743	3.730 ± 0.9090
	5	4.650 ± 0.9936	4.664 ± 1.037

Table 5.8 shows a positive relationship between the inter-cluster distance and K for both clustering techniques. The Kmedoids clustering technique found more separated clusters than the Kmeans technique across all test images with K varied. This is due to the highest average inter-cluster distances obtained by the Kmedoids and Kmeans clustering techniques of 156.1×10^7 and 148.7×10^7 , respectively.



University of Fort Hare
Together in Excellence

5.2.5 Optimal number of clusters

The silhouette validity index is used as an internal validity index to measure the goodness of the clustering solutions found by the Kmeans and Kmedoids clustering techniques. Table 5.9 presents the comparison of the average silhouette values and their standard deviations for the Kmeans and Kmedoids clustering techniques. A graphical comparison of the average silhouette values is also provided in Figures 5.22 to 5.33.

Tab. 5.9: Averages and Standard deviations of Silhouette validity index

Image	K	Silhouette value	
		Kmeans	Kmedoids
Butterfly	2	0.895 ± 0.022	0.895 ± 0.022
	3	0.793 ± 0.051	0.797 ± 0.050
	4	0.678 ± 0.035	0.696 ± 0.026
	5	0.644 ± 0.082	0.640 ± 0.080
Insect	2	0.789 ± 0.178	0.798 ± 0.163
	3	0.636 ± 0.046	0.649 ± 0.029
	4	0.519 ± 0.107	0.493 ± 0.106
	5	0.461 ± 0.088	0.473 ± 0.070
Soldier	2	0.691 ± 0.084	0.691 ± 0.084
	3	0.616 ± 0.083	0.615 ± 0.083
	4	0.582 ± 0.089	0.582 ± 0.089
	5	0.559 ± 0.090	0.558 ± 0.091
Kids	2	0.859 ± 0.023	0.860 ± 0.022
	3	0.742 ± 0.068	0.717 ± 0.084
	4	0.672 ± 0.020	0.670 ± 0.018
	5	0.638 ± 0.027	0.615 ± 0.062
Swimmer	2	0.699 ± 0.061	0.705 ± 0.050
	3	0.587 ± 0.067	0.569 ± 0.054
	4	0.534 ± 0.087	0.524 ± 0.079
	5	0.537 ± 0.100	0.576 ± 0.088
Car	2	0.765 ± 0.019	0.767 ± 0.031

	3	0.710 ± 0.049	0.713 ± 0.046
	4	0.711 ± 0.063	0.710 ± 0.051
	5	0.691 ± 0.058	0.690 ± 0.052
Aeroplane	2	0.767 ± 0.202	0.770 ± 0.199
	3	0.580 ± 0.113	0.544 ± 0.110
	4	0.527 ± 0.051	0.505 ± 0.065
	5	0.527 ± 0.101	0.516 ± 0.074
Flowers	2	0.561 ± 0.236	0.693 ± 0.218
	3	0.657 ± 0.115	0.686 ± 0.011
	4	0.673 ± 0.067	0.665 ± 0.070
	5	0.618 ± 0.055	0.618 ± 0.048
Swan	2	0.890 ± 0.037	0.889 ± 0.036
	3	0.709 ± 0.087	0.703 ± 0.125
	4	0.653 ± 0.070	0.691 ± 0.076
	5	0.651 ± 0.093	0.683 ± 0.080
Statues	2	0.668 ± 0.069	0.664 ± 0.068
	3	0.641 ± 0.048	0.621 ± 0.072
	4	0.631 ± 0.054	0.654 ± 0.022
	5	0.633 ± 0.013	0.626 ± 0.016
Star-fish	2	0.739 ± 0.036	0.740 ± 0.036
	3	0.660 ± 0.049	0.656 ± 0.048
	4	0.609 ± 0.049	0.585 ± 0.043
	5	0.577 ± 0.055	0.566 ± 0.056
Horses	2	0.590 ± 0.058	0.589 ± 0.060

3	0.636 ± 0.053	0.635 ± 0.053
4	0.549 ± 0.016	0.554 ± 0.019
5	0.500 ± 0.048	0.495 ± 0.051

The average silhouette values obtained by the clustering techniques range from 0.50 to 0.90 for both clustering techniques. This implies that reasonable to strong clusters are found by the clustering techniques considered in this study.

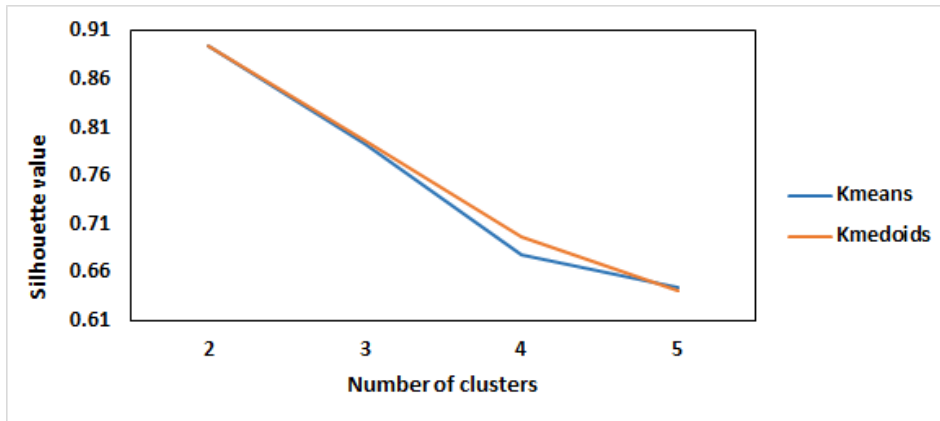


Fig. 5.22: Silhouette validity index versus Number of clusters for the Butterfly image

Figure 5.22 shows that the Kmedoids clustering techniques obtained higher silhouette values than the Kmeans technique for all the number of clusters the experiments were run for, except when $K=2$. The highest and lowest average silhouette values were obtained when $K=2$ and 5 for both clustering techniques. The Kmeans and Kmedoids clustering techniques obtained the lowest average silhouette values of 0.644 and 0.640, respectively. The clustering techniques obtained equivalent highest average silhouette value of

0.895 each. Thus, the average silhouette values obtained indicates that the optimal number of clusters for the Butterfly image is 2.

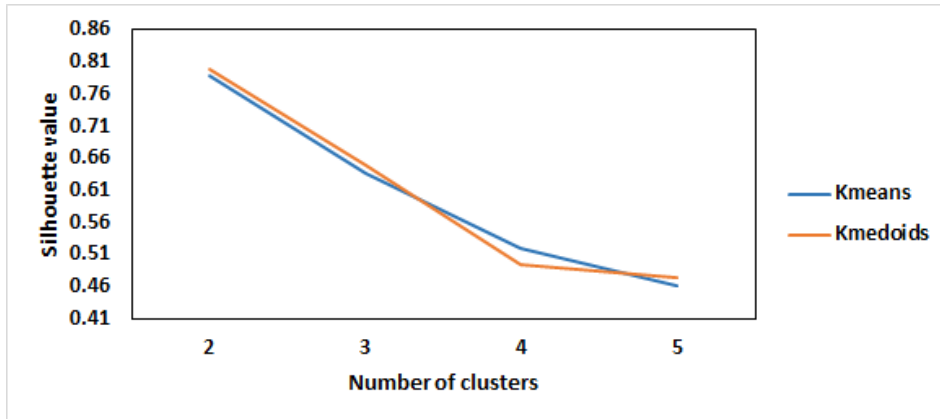


Fig. 5.23: Silhouette validity index versus Number of clusters for the Insect image

It was observed from Figure 5.23 that the highest and lowest silhouette values were obtained when $K=2$ and 5 for both the clustering techniques. The Kmeans technique obtained the lowest average silhouette value of 0.461 and the highest average silhouette value of 0.789 , while the Kmedoids technique obtained the lowest average silhouette value of 0.473 and highest average silhouette value of 0.798 . Thus, the average silhouette value obtained suggests that strongest clustering structures were found when $K=2$.

Figure 5.24 shows that the lowest average silhouette values obtained by the Kmeans and Kmedoids clustering techniques were 0.559 and 0.558 , respectively. The clustering techniques obtained an equivalent highest average silhouette value of 0.691 . The Kmeans and Kmedoids clustering techniques obtained the highest average silhouette value when the number of clusters, K , was 2 .

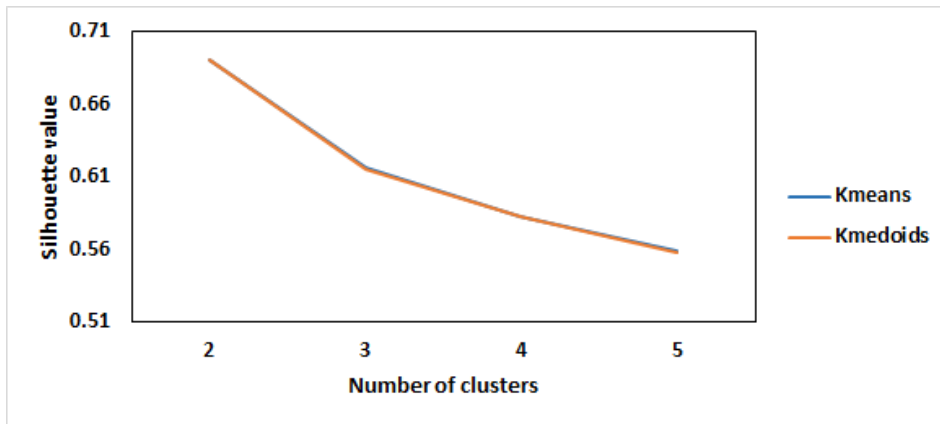


Fig. 5.24: Silhouette validity index versus Number of clusters for the Soldier image

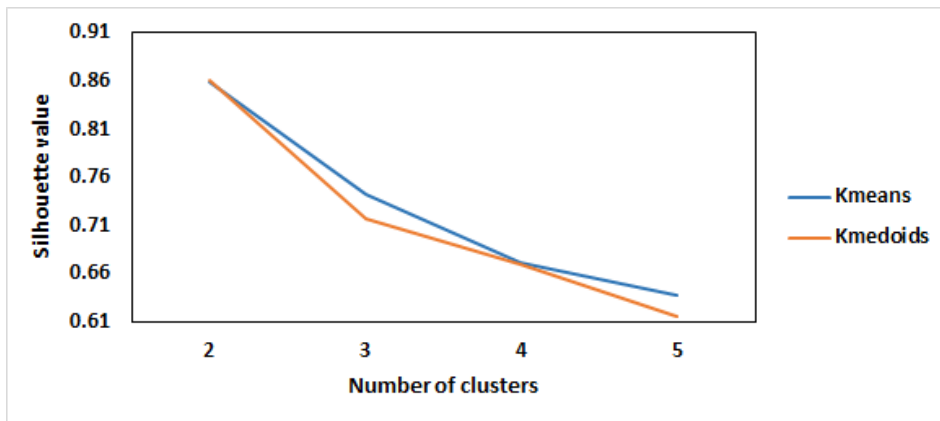


Fig. 5.25: Silhouette validity index versus Number of clusters for the Kids image

The average silhouette values obtained for the Kids image indicate that the optimal number of clusters for both clustering techniques is 2. This is due to the highest average silhouette values of 0.859 and 0.860 obtained by the Kmeans and Kmedoids clustering techniques. In addition, both clustering techniques obtained the least average silhouette values when $K=5$.

The average silhouette values obtained for the Swimmer image by the Kmeans and Kmedoids clustering techniques suggested that the best clustering solution is observed when $K=2$ as the maximum silhouette values of

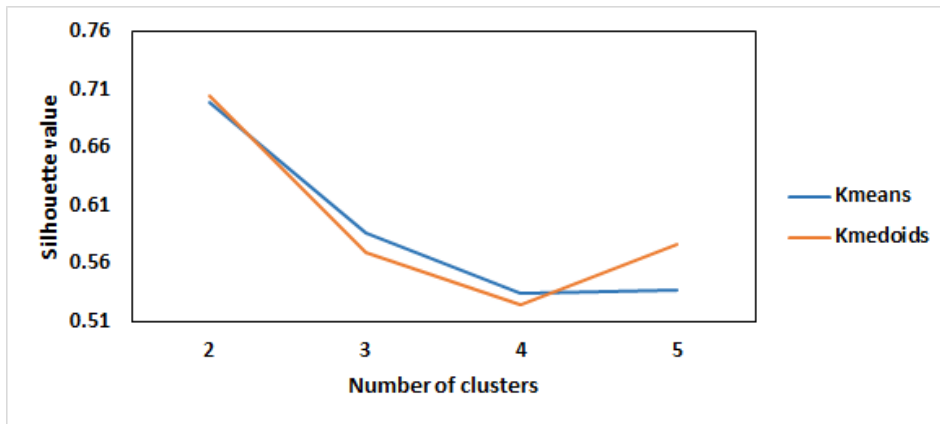


Fig. 5.26: Silhouette validity index versus Number of clusters for the Swimmer image

0.699 and 0.705 were obtained for the Kmeans and Kmedoids clustering techniques, respectively. In addition, the lowest averages silhouette values were obtained when $K=4$ where the Kmeans technique obtained a value of 0.534 and the Kmedoids technique obtained 0.524.

University of Fort Hare

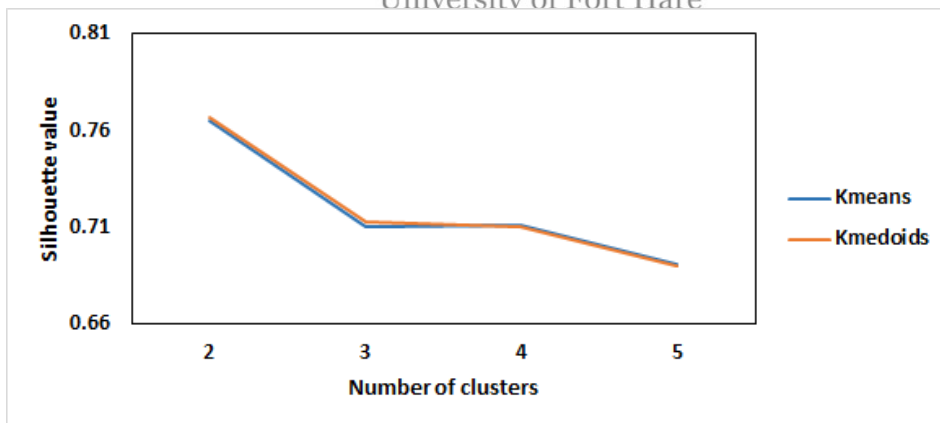


Fig. 5.27: Silhouette validity index versus Number of clusters for the Car image

Figure 5.27 shows that the Kmeans and Kmedoids clustering techniques obtained similar average silhouette values. The techniques obtained their highest and lowest average silhouette value when $K=2$ and 5. Therefore, ac-

According to the average silhouette values obtained, the best clustering structures are found when $K=2$.

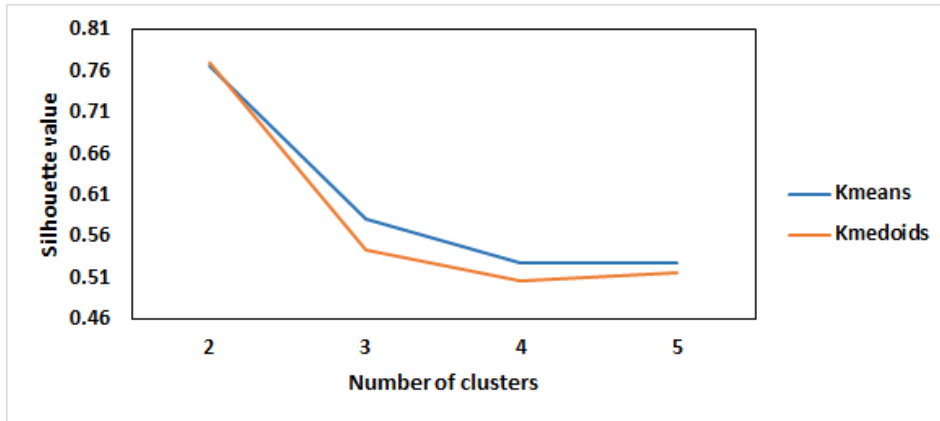


Fig. 5.28: Silhouette validity index versus Number of clusters for the Aeroplane image



The Kmeans clustering technique finds better clustering structures than the Kmedoids technique for the Aeroplane image. This is as a result of the average silhouette values of the Kmeans technique being greater than to those obtained by the Kmedoids technique, except when $K=2$. The Kmeans technique obtained the lowest average silhouette value of 0.527 and highest average silhouette value of 0.767, while the Kmedoids technique obtained the lowest average silhouette value of 0.505 and highest average silhouette value of 0.770. In addition, the highest and lowest average silhouette values are obtained when $K=2$ and 4, respectively.

It was observed from Figure 5.29 that the Kmeans clustering technique obtains the lowest and highest average silhouette values of 0.561 when $K=2$ and 0.673 when $K=4$, respectively. On the other hand, the Kmedoids clustering techniques obtained the lowest and highest average silhouette values

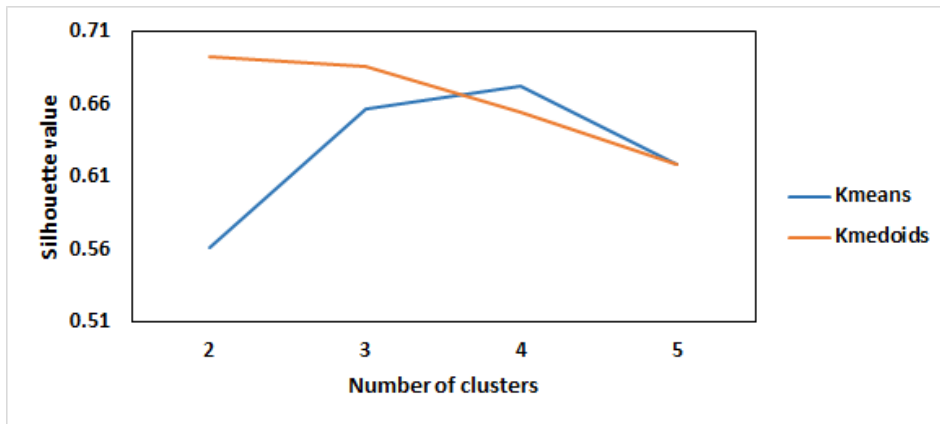


Fig. 5.29: Silhouette validity index versus Number of clusters for the Flowers image of 0.618 when $K=5$ and 0.693 when $K=2$, respectively. Thus, the best clustering structures are found when $K=4$ for the Kmeans clustering technique and when $K=2$ for the Kmedoids technique.

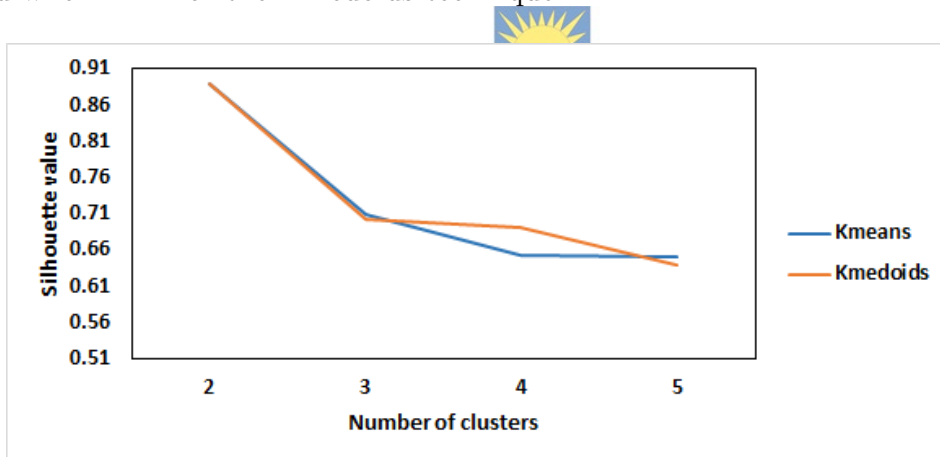


Fig. 5.30: Silhouette validity index versus Number of clusters for the Swan image

The average silhouette values obtained for the Swan image indicate that the optimal number of clusters for both clustering techniques is 2. This is due to the highest average silhouette values of 0.890 and 0.889 obtained by the Kmeans and Kmedoids clustering techniques. In addition, both clustering techniques obtained the least average silhouette values when $K=5$.

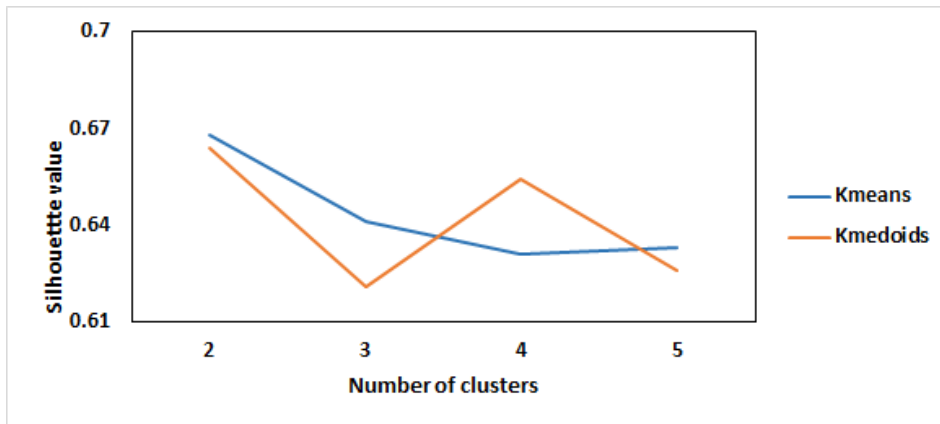


Fig. 5.31: Silhouette validity index versus Number of clusters for the Statues image

It was observed from Figure 5.31 that the highest and lowest silhouette values were obtained when $K=2$ for both the clustering techniques. The Kmeans clustering technique obtained its lowest silhouette value when $K=4$, while the Kmedoids technique obtained it when $K=3$. The Kmeans clustering technique obtained the lowest average silhouette value of 0.631 and the highest average silhouette value of 0.668, while the Kmedoids technique obtained the lowest average silhouette value of 0.621 and highest average silhouette value of 0.664.

Figure 5.32 shows that the Kmeans and Kmedoids clustering techniques obtained their highest and lowest average silhouette values when $K=2$ and 5. The Kmeans clustering technique obtained the lowest average silhouette value of 0.577 and the highest average silhouette value of 0.739, while the Kmedoids technique obtained the lowest average silhouette value of 0.566 and highest average silhouette value of 0.740. Therefore, according to the average silhouette values obtained, the best clustering structures are found when $K=2$.

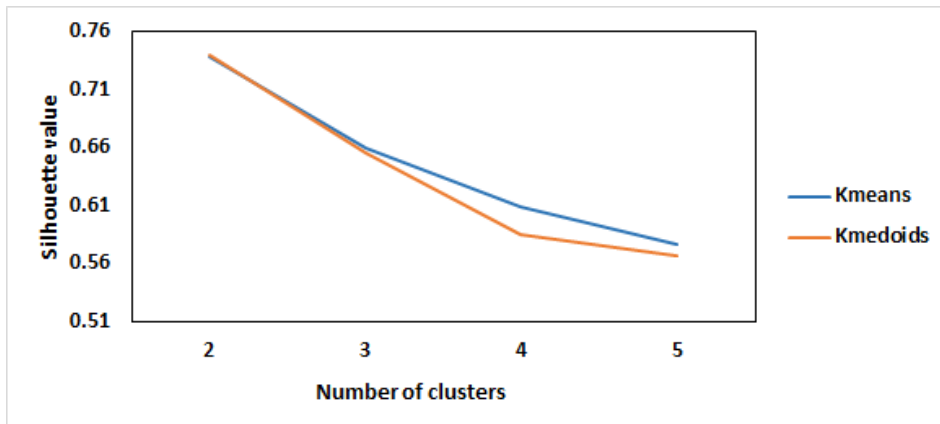


Fig. 5.32: Silhouette validity index versus Number of clusters for the Star-fish image



Fig. 5.33: Silhouette validity index versus Number of clusters for the Horses image

Figure 5.33 shows that the Kmeans and Kmedoids clustering techniques obtained similar average silhouette values. The techniques obtained their highest and lowest average silhouette value when $K=3$ and 5. The Kmeans clustering technique obtained the lowest average silhouette value of 0.500 and the highest average silhouette value of 0.636, while the Kmedoids technique obtained the lowest average silhouette value of 0.495 and highest average silhouette value of 0.635. Therefore, the best clustering structures are found

when $K=3$.

It was observed from the average silhouette values obtained by the Kmeans and Kmedoids clustering techniques for the test images that most of the images obtained their best clustering solutions when $K=2$. This is with the exception of the Flowers image whose highest average silhouette value was acquired when $K=4$ by the Kmeans clustering technique. In addition, it was seen that the lowest average silhouette values were obtained when $K=5$ for most images. The Butterfly image obtained the highest average silhouette value for both the Kmeans and Kmedoids clustering techniques amongst all the other images. On the other hand, the Insect image obtained the lowest average silhouette values for both clustering techniques amongst the test images. Furthermore, all the average silhouette values obtained ranged from 0.26-1.00. This implies that a few weak clustering structures were found by the Kmeans and Kmedoids clustering techniques in some test images.

The runtimes for GMMs based on Kmeans (GMM-Kmeans) and those based on Kmedoids (GMM-Kmedoids) were monitored. The average runtimes used to generate the background and foreground GMMs for the GMM based on the Kmeans clustering technique as compared to the ones generated by the GMM based on the Kmedoids clustering technique are presented in Figure 5.34.

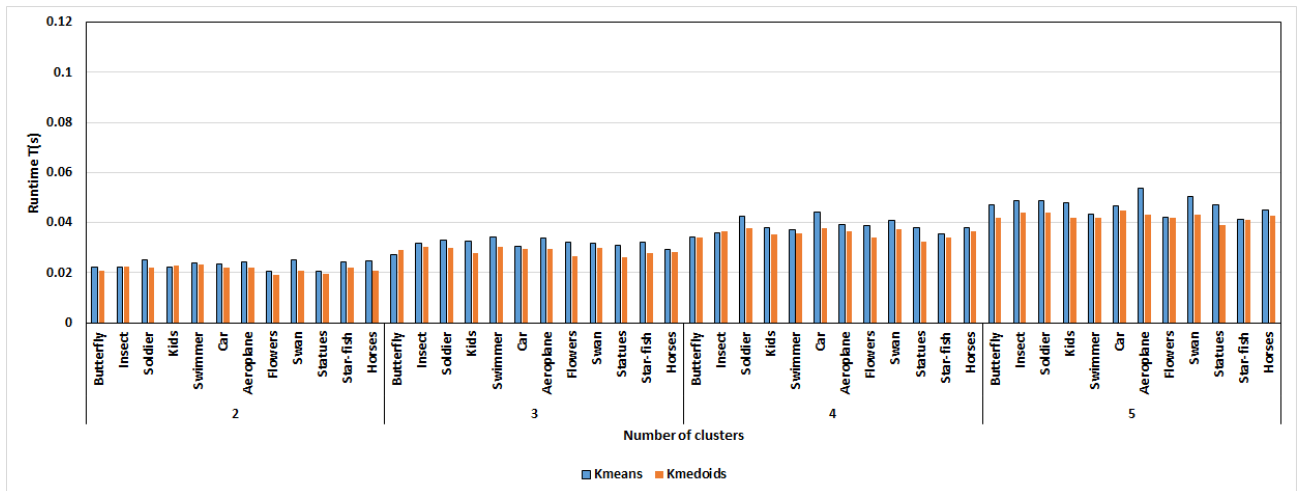


Fig. 5.34: Average runtime GMM-Kmeans versus GMM-Kmedoids

Figure 5.34 shows that the time taken by the Kmeans clustering technique to generate GMMs is slightly higher than the time taken to generate GMM-Kmedoids for most of the images. It is also observed that in some instances the GMM-Kmeans utilised runtimes that are equal to those used by GMM-Kmedoids, for instance, for the Insect image when $K=2$. In addition, the results obtained show a positive relationship between the average runtime and number of clusters in all images for both the GMM-Kmeans and GMM-Kmedoids. This implies that as the number of clusters is increased the more computational time is required by the algorithms to generate GMMs. Furthermore, the runtimes ranged from 0,0205-0,0536 seconds for GMM-Kmeans and 0.0193-0.0447 seconds for GMM-Kmedoids, respectively.

5.2.6 Segmentation results

Figure 5.35 shows that the Grabcuts for segmentation algorithm with the GMM-Kmedoids clustering heuristic outperformed its counterpart. The Grab-

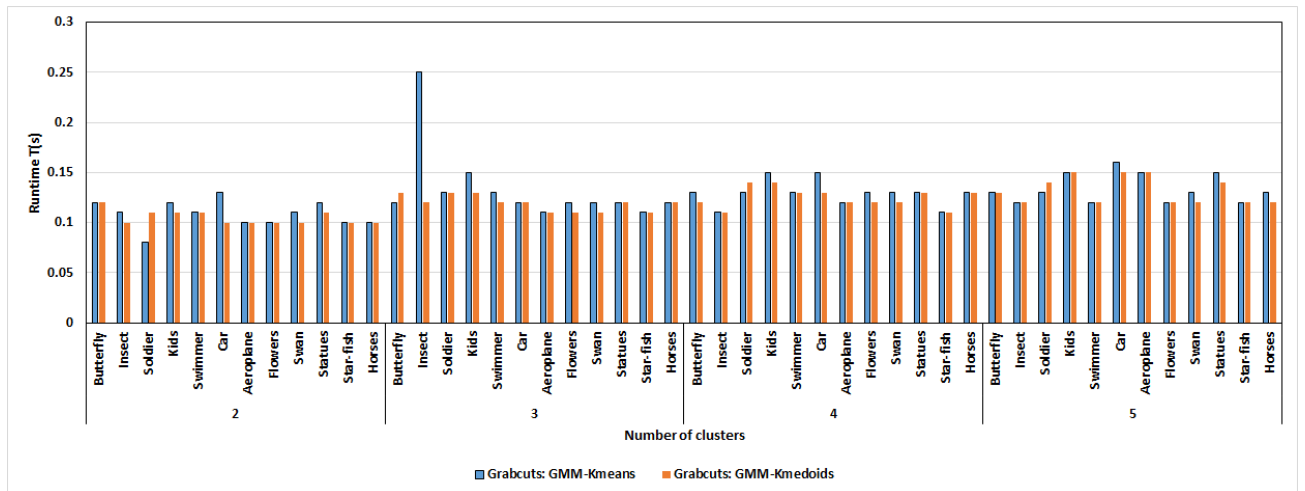


Fig. 5.35: Average runtimes for Grabcuts for image segmentation algorithm: GMM-Kmeans versus GMM-Kmedoids

cuts for image segmentation algorithm with the GMM-Kmeans clustering heuristic obtained the lowest average runtime of 0.08 seconds when K was varied from 2 to 5 across all images. On the other hand, it obtained the highest average runtime of 0.26 seconds when K was varied from 2 to 5 across all test images. Most images required runtimes of 0.10 seconds when K=2, 0.12 seconds when K=3 and 0.13 seconds when K=4 and 5. The Soldier image was observed to require the least runtime when K=2, whereas the Star-fish image required the least runtimes when K was varied from 3 to 4.

The Grabcuts for image segmentation algorithm with the GMM-Kmedoids clustering heuristic obtained the lowest runtime of 0.10 seconds when K was varied from 2 to 5 for all test images. A highest runtime of 0.15 seconds was obtained for this algorithm across test images with K varied. The lowest and highest runtimes were obtained when the algorithm was executed on the Car and Aeroplane images with K=2 and K=5, respectively. In addition, most of the test images required average runtimes of 0.10 seconds when K was 2

and 0.12 seconds when K was varied from 3 to 5. In addition, the Insect, Swan and Star-fish images required the least average runtimes for all K=2 to 5 as compared to the rest of the test images. On the other hand, the Kids, Soldier and Butterfly images required more average runtimes.

The segmentation results of the Grabcuts for segmentation algorithm are presented in Figure 5.36. The segmentation results for the Grabcuts for image segmentation with the GMM-Kmeans and GMM-Kmedoids clustering heuristic are presented in two consecutive rows for each test image. The first row shows results for the Grabcuts for image segmentation with the GMM-Kmeans clustering heuristic and the second row shows results for the Grabcuts for image segmentation with the GMM-Kmedoids clustering heuristic. The number of clusters increases from left to right, which implies the left most and right most results are obtained when K=2 and K=5, respectively.



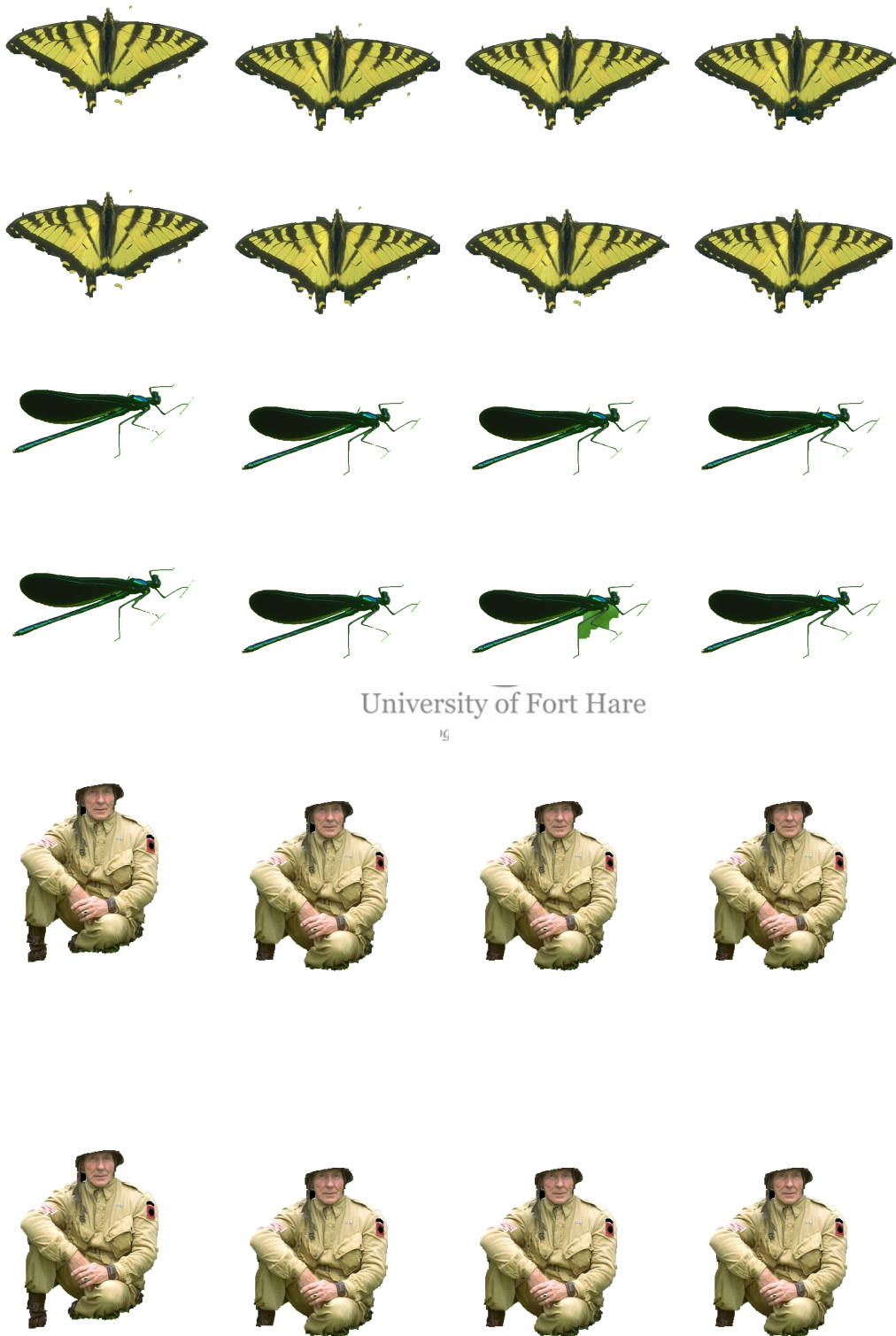


Fig. 5.36: Segmentation results for Grabcuts for image segmentation: GMM-Kmeans versus GMM-Kmedoids



Fig. 5.36: Segmentation results for Grabcuts for image segmentation: GMM-Kmeans versus GMM-Kmedoids



Fig. 5.36: Segmentation results for Grabcuts for image segmentation: GMM-Kmeans versus GMM-Kmedoids



Fig. 5.36: Segmentation results for Grabcuts for image segmentation: GMM-¹³⁶Kmeans versus GMM-Kmedoids

The Grabcuts for image segmentation algorithm based on the GMM-Kmeans and GMM-Kmedoids clustering heuristic obtained segmentation results in which some pixels that were supposed to be in the foreground were classified as background. This was accredited to the low contrast regions seen between the transition areas from background and foreground. The algorithms obtained identical segmentation results when K was varied from 2 to 3 for the Butterfly image. This was not the case when $K=4$ and 5, as the Grabcuts for image segmentation algorithm based on the GMM-Kmeans obtained better segmentation results when $K=4$ than its counterpart. While, the Grabcuts for image segmentation algorithm based on the GMM-Kmedoids obtained better segmentation results when $K=5$ than its counterpart.

The Grabcuts for image segmentation algorithm based on the GMM-Kmeans and GMM-Kmedoids clustering heuristic obtained similar segmentation results for the Insect image. The misclassification of some background pixels around the body of the insect was observed for all the segmentation results obtained. The Grabcuts for image segmentation algorithm based on the GMM-Kmedoids clustering heuristic performed worst when $K=4$, as some of the leaf that was supposed to be classified as background ended up being part of the foreground.

The soldier's helmet and parts of his boots were taken to mistakenly belong to the background on all segmentation results obtained. This is due to the overlap in colour space of the foreground and background colour distributions [48]. The Grabcuts for image segmentation algorithm based on the GMM-Kmeans and GMM-Kmedoids cluster heuristic obtained identical

segmentation results with K varied from 2 to 5.

The Grabcuts for image segmentation algorithm based on the GMM-Kmeans and GMM-Kmedoids clustering heuristic obtained similar segmentation results for the Kids image. The results showed misclassified background pixels on top of the second boy's clay pot. In addition, when $K=2$ some pixels belonging to the first boy's clay pot and face are misclassified for the Grabcuts for image segmentation algorithm based on the GMM-Kmeans clustering heuristic, whereas this occurred when $K=2$ and 3 for the Grabcuts for image segmentation algorithm based on the GMM-Kmedoids clustering heuristic.

The Swimmer image obtained segmentation results in which more background pixels around the swimmer were taken to belong to the set of foreground pixels. Also, some foreground pixels on the eyes and left shoulder of the swimmer were taken to belong to the set of background pixels. This is as a result of the selected ROI's background material not being adequately represented by the set of pixels belonging to the background region. The Grabcuts for image segmentation algorithm based on the GMM-Kmeans and GMM-Kmedoids clustering heuristic obtained their best segmentation results when $K=5$.

The tyres and windows of the car were mistaken to belong to the set of background pixels. This was as a result of them sharing a similar colour distribution with the background pixels. In addition, some of the background pixels were observed to have been misclassified as seen around the transition areas from background to foreground regions, around the car. For the Car

image, the Grabcuts for image segmentation algorithm with a GMM based on the Kmeans clustering technique obtained identical segmentation results with the algorithm with a GMM based on the Kmedoids clustering technique when $K=2$ and 3 . It obtained better and inferior segmentation results than the algorithm with a GMM based on the Kmedoids clustering technique when $K=4$ and when $K=5$, respectively.

The Aeroplane image obtained similar segmentation results for all algorithms. This result is with a few exceptions where the Grabcuts for image segmentation algorithm with a GMM based on the Kmedoids clustering technique obtained better segmentation results when $K=5$. It was also observed that all algorithms considered got their worst segmentation results when $K=2$.



The Grabcuts for image segmentation algorithm with a GMM based on the Kmeans and Kmedoids clustering techniques obtained identical segmentation results when K was varied from 3 to 5 for the Flowers image. The segmentation results showed a misclassification of background pixels around the flowers which share similar colour distribution as some of the pixels in the foreground region of the flowers. In addition, the Grabcuts for image segmentation algorithm with a GMM based on the Kmeans clustering technique obtained better segmentation results, than its counterpart when $K=2$.

The Swan image obtained the best segmentation results when $K=3, 4$ and 5 for the Grabcuts for image segmentation algorithm with a GMM based on the Kmeans and Kmedoids clustering techniques. On the other hand, the algorithms obtained their worst segmentation results when $K=2$.

The Grabcuts for image segmentation algorithm with a GMM based on the Kmeans clustering technique obtained better segmentation results when $K=2$ and 3 , than its counterpart for the Statues image. On the other hand, the Grabcuts for image segmentation algorithm with a GMM based on the Kmedoids clustering technique obtained better segmentation results when $K=2$, than the Grabcuts for image segmentation algorithm with a GMM based on the Kmedoids clustering technique. The algorithms obtain identical segmentation results when $K=5$.

The Star-fish image segmentation results obtained for the Grabcuts for image segmentation algorithm with a GMM based on the Kmeans and Kmedoids clustering techniques are identical. In addition, the segmentation results obtained by this image for all algorithms considered with the variation of K , showed minimal pixel misclassification amongst all the test images considered.

The segmentation results obtained by the Horses image are identical for all algorithms considered, where background pixels are seen to be misclassified around the tail of the horse. This misclassification of pixels is accredited to the overlap of colour distributions around hairy objects.

Tab. 5.10: Average Precision and recall: Grabcuts: GMM-Kmeans versus Grabcuts: GMM-Kmedoids

Image	K	Grabcuts: GMM-Kmeans		Grabcuts: GMM-Kmedoids	
		Precision	Recall	Precision	Recall
Butterfly	2	0.824	0.588	0.824	0.588
	3	0.823	0.586	0.823	0.586

	4	0.824	0.624	0.825	0.650
	5	0.825	0.625	0.826	0.651
Insect	2	0.496	0.805	0.501	0.814
	3	0.483	0.804	0.483	0.804
	4	0.499	0.845	0.483	0.802
	5	0.498	0.845	0.497	0.803
Soldier	2	0.691	0.949	0.691	0.949
	3	0.691	0.949	0.691	0.949
	4	0.691	0.949	0.692	0.949
	5	0.691	0.949	0.691	0.949
Kids	2	0.789	0.807	0.789	0.634
	3	0.789	0.634	0.789	0.634
	4	0.789	0.634	0.789	0.700
	5	0.789	0.634	0.789	0.634
Swimmer	2	0.690	0.464	0.698	0.601
	3	0.689	0.380	0.691	0.518
	4	0.692	0.384	0.691	0.518
	5	0.701	0.565	0.700	0.573
Car	2	0.659	0.656	0.659	0.656
	3	0.659	0.656	0.659	0.656
	4	0.660	0.683	0.662	0.832
	5	0.647	0.683	0.660	0.683
Aeroplane	2	0.695	0.626	0.696	0.629
	3	0.693	0.651	0.693	0.649

	4	0.664	0.654	0.694	0.636
	5	0.694	0.652	0.696	0.703
Flower	2	0.734	0.954	0.711	0.602
	3	0.711	0.602	0.711	0.602
	4	0.712	0.670	0.709	0.557
	5	0.721	0.895	0.722	0.899
Swan	2	0.473	0.630	0.473	0.630
	3	0.475	0.630	0.475	0.630
	4	0.475	0.630	0.475	0.630
	5	0.473	0.630	0.472	0.569
Statues	2	1	0.707	1	0.707
	3	1	0.707	1	0.707
	4	1	0.707	1	0.833
	5	1	0.848	1	0.848
Star-fish	2	0.892	0.591	0.892	0.591
	3	0.892	0.591	0.892	0.591
	4	0.892	0.591	0.892	0.591
	5	0.892	0.591	0.892	0.591
Horses	2	0.707	0.579	0.707	0.579
	3	0.707	0.579	0.707	0.579
	4	0.707	0.579	0.707	0.579
	5	0.707	0.579	0.708	0.634

Table 5.10 shows that the segmented Butterfly images obtained similar precision values for both Grabcuts: GMM-Kmeans and Grabcuts: GMM-

Kmedioids, except for when $K=5$. On the other hand, the recall values obtained when the Grabcuts: GMM-Kmedioids was used for the Butterfly image were slightly higher than those obtained by its counterpart when $K=4$ and 5 . In addition, it is observed that the precision values are higher than the recall values for all the segmented Butterfly images. This implies that these image have been under segmented.

The Grabcuts: GMM-Kmedioids was observed to have obtained a segmented Insect image with higher precision and recall values than the Grabcuts: GMM-Kmeans when $K=2$. On the other hand, when $K=4$ and 5 a higher recall value is obtained by the Grabcuts: GMM-Kmeans segmented Insect image, while the Grabcuts: GMM-Kmedioids achieves a higher recall and precision values when $K=2$. In addition, the precision values of all the segmented Insect values are lower than their recall values. This implies that these results suffer from over segmentation.

The segmented Soldier image obtained similar precision and recall values for both segmentation algorithms considered. The recall values were observed to be much higher than the precision values when K was varied for both the Grabcuts: GMM-Kmeans and Grabcuts: GMM-Kmedioids segmentation results. This implies that the segmented Soldier images suffered from over segmentation, which is as a result of misclassification of some foreground pixels as background pixels.

It is observed that the precision values obtained from the segmented Kids images are higher than the recall values when K is varied from 3 to 5 for Grabcuts: GMM-Kmeans and when K is varied from 2 to 4 for Grabcuts:

GMM-Kmedioids. In addition, both image segmentation algorithms obtain similar precision values when K is varied from 2 to 5. Furthermore, it was observed that the similar recall values were obtained by the segmented Kids images, except for when K=2 for the Kids image segmented by the Grabcuts: GMM-Kmeans and when K=4 the Kids image segmented by the Grabcuts: GMM-Kmedioids.

The segmented Swimmer image obtained by the Grabcuts: GMM-Kmeans achieved its highest precision and recall values when K=5. On the other hand, it was observed that the segmented Swimmer image obtained by the Grabcuts: GMM-Kmedioids achieved its highest precision and recall values when K=5 and K=2, respectively. It was observed that the segmented Swimmer images had obtained higher precision values for both segmentation algorithms considered.



University of Fort Hare
Together in Excellence

The segmented Car images were observed to have obtained similar precision and recall values for both segmentation algorithms considered under scenario 2 when K from 2 to 3. In addition, the segmentation algorithms obtained images with higher precision values the recall values when K=2 and only when K=3. On the other hand, it was observed that the recall values were higher than the precision values when K= 4 and 5 for the segmented Car images obtained by both segmentation algorithms. This implies that the Car images were under segmented when K was varied from 2 to 3 and over segmented when K was varied from 4 to 5.

The highest precision and recall values for the segmented Aeroplane images obtained by the Grabcuts: GMM-Kmeans achieves their highest precision

and recall values when $K=2$ and $K=4$, respectively. On the other hand, the Grabcuts: GMM-Kmedioids obtained segmented images that achieve their highest precision when $K= 2$ and 5 . The highest recall value for segmented images obtained by the Grabcuts: GMM-Kmedioids is observed when $K=5$.

The segmented Flower images obtained their highest precision and recall values for the Grabcuts: GMM-Kmeans and Grabcuts: GMM-Kmedioids when $K= 2$ and $K=5$, respectively. In addition, it can be observed that under segmented Flower images are obtained when K is varied from 3 to 4 by the Grabcuts: GMM-Kmeans and when K is varied from 2 to 4 by the Grabcuts: GMM-Kmedioids.

The segmented Swan images are observed to have obtained similar precision and recall values when K was varied for 2 to 4 under both segmentation algorithms considered. The segmented Swan images obtained higher recall values than precision values, thus implying that the segmentation results produced are over segmented.

The segmented Statues images obtained similar precision and recall values when K was varied for 2 to 5 under both segmentation algorithms considered. This is with the exception of the segmented images obtained when $K=4$ for both segmentation algorithms, which achieved different recall values of 0.707 and 0.833 . The precision values are observed to be at their maximum value for all the segmented Statues images.

The segmented images for the Star-fish obtained similar precision and recall values when K was varied from 2 to 5 for both segmentation algorithm con-

sidered. The same trend was observed from Table 5.10 for the segmented Horses images, except for when $K=5$. In addition, it was observed that the precision values were much larger than the recall values. This implies that these segmented images suffer from under segmentation.

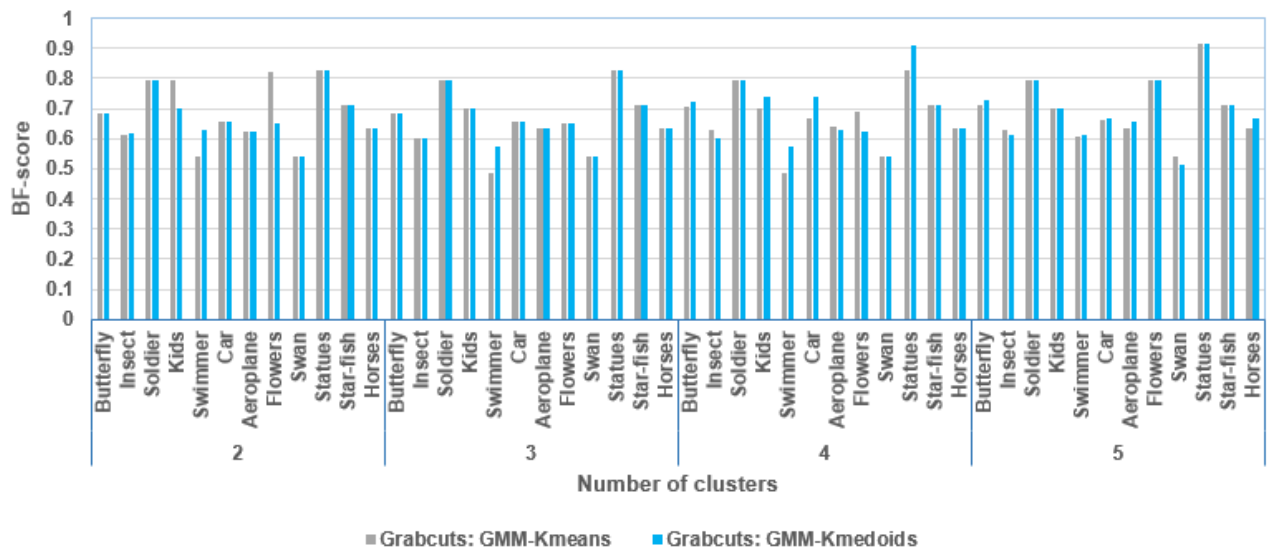


Fig. 5.37: Average BF-scores for Grabcuts for image segmentation algorithm: GMM-Kmeans versus GMM-Kmedoids

Figure 5.37 shows the BF-scores for the segmented images. All these results indicate that the quantitative evaluation measure chosen matches the relative visual quality well. The segmented Statues images is observed to have the highest BF-scores when K is varied for both the Grabcuts: GMM-Kmeans as well as the Grabcuts: GMM-Kmedoids. On the other hand, the segmented Swan images obtain the lowest scores when $K=2$ and 5 for both the Grabcuts: GMM-Kmeans and Grabcuts: GMM-Kmedoids, except for when $K=3$ and 4 . Figure 5.37 shows that the Grabcuts: GMM-Kmedoids produces images with higher BF-scores than its counterpart when K was

varied from 2 to 5 for most of the test images.

5.3 Discussions

The development of a computationally inexpensive and efficient algorithm to perform image segmentation has been highlighted by many researchers [25, 36, 47, 48]. Thus the need to explore option of clustering techniques that could make the Grabcuts for image segmentation algorithm less computational expensive was realised. This study compared the performance of the Grabcuts for image segmentation algorithm developed by [48] when GMM Kmeans and Kmedoids were used to cluster pixels in an image. This comparison was done under two scenarios where the distance measures used to calculate the similarities and dissimilarities between the pixels of an image were varied from the Squared Euclidean to the City block distance measure. The average runtimes for the Kmeans and Kmedoids clustering techniques were measured. The average runtimes for the GMM-Kmeans and GMM-Kmedoids were also measured. Ultimately, the average runtimes required for running the Grabcuts for image segmentation with a GMM-Kmeans and GMM-Kmedoids were measured and reported. Additionally, the clustering structures found by the Kmeans and Kmedoids clustering techniques were evaluated using the average intra- and inter- cluster distances. These clustering structures were further evaluated using the silhouette validity index to check their goodness.

It was observed that the Kmeans clustering technique took less time to find a clustering structure for all the test images with K varied from 2 to 5,

than the Kmedoids clustering technique. In addition, the runtimes obtained under scenario 2 for both the Kmeans and Kmedoids clustering technique were less than those obtained under scenario 1. This implies that the use of the Squared Euclidean distance is computationally expensive than the use of the City block distance measure as observed in [8,33]. This is due to the fact that the City block distance weakens the effect of outliers, while the Squared Euclidean distance progressively strengthens the effect of outliers by squaring them. The average intra-cluster distances obtained for the test images under scenario 2 are $\times 10$ smaller than those obtained under scenario 1 for both the Kmeans and Kmedoids clustering techniques. This implies that the City block distance aided the Kmeans and Kmedoids clustering techniques in finding more compact clusters than the Squared Euclidean distance. The Kmeans clustering technique found more compact clusters than the Kmedoids clustering technique when the Squared Euclidean distance was used for calculating similarities and dissimilarities between pixels. Although this was the case, the Kmedoids clustering technique obtained the smallest average intra-cluster distance when considering all the results obtained under scenario 1. The Kmedoids clustering technique found more compact clusters than the Kmeans clustering technique when the City block distance was used. The Kmedoids clustering technique obtained the minimum average intra-cluster distance when all intra-cluster distances results were obtained under scenario 2. In addition, the lowest and highest average intra-cluster distances were obtained when $K=2$ for the Insect and Butterfly images by both clustering techniques when the Squared Euclidean distance was used.



University of Port Harcourt

Together in Excellence

On the contrary, both clustering techniques obtained their lowest and highest intra-cluster distances when $K=5$ for the Aeroplane image and $K=2$ for Insect when the City block distance was used.

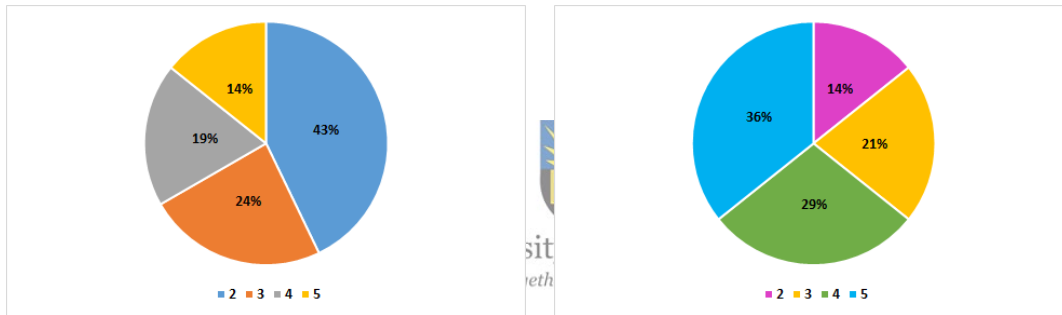
It was observed that the average inter-cluster distances obtained under scenario 2 were $\times 10^2$ smaller than those obtained under scenario 1. This implies that the Squared Euclidean distance aided the Kmeans and Kmedoids clustering techniques in finding more separate clusters than the City block distance. The Kmeans clustering techniques obtained the most separate clusters under scenario 1, while the Kmedoids clustering obtained the most separate clusters under scenario 2. In addition, the Kmeans clustering technique obtained the highest average inter-cluster distance among all results obtained under scenario 1. Although this is the case, the Kmedoids clustering technique obtained the highest average inter-cluster distance among all results obtained under scenario 2. Furthermore, the lowest and highest average inter-cluster distances were obtained when $K=2$ for the Insect and Butterfly images by both clustering techniques when the Squared Euclidean distance was used. Similarly, both clustering techniques obtained their lowest inter-cluster distances when $K=2$ for the Horses image, whereas they obtained their highest inter-cluster distances when $K=3$ for the Insect image when the City block distance was used.

It was observed from the average silhouette values obtained by the Kmeans and Kmedoids clustering techniques for the test images that all the images obtained their best clustering solutions when $K=2$. This is with the exception of the Butterfly image whose highest Silhouette value is acquired when



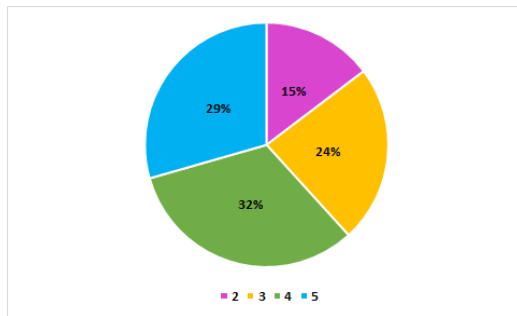
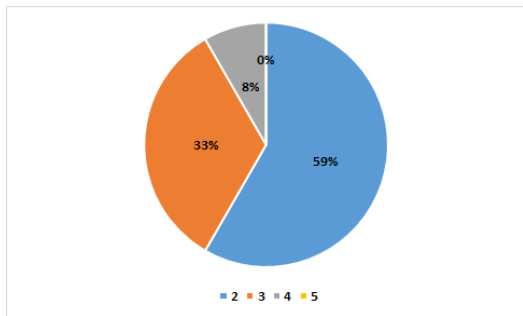
University of Port Harcourt
Together in Excellence

$K=3$. In addition, it was seen that the lowest average silhouette values were obtained when $K=5$ for most images. The Swan image obtained the highest average silhouette value for both the Kmeans and Kmedoids clustering techniques amongst all the other images, while the Horses image obtained the lowest average silhouette values for both clustering techniques amongst the test images. Furthermore, all the average silhouette values obtained are above 0.5. This implies that reasonable to strong clustering structure were found when the Kmeans and Kmedoids clustering techniques were used on the test images.



(a) Number of average silhouette values between 0.71-1.00 versus K (b) Number of average silhouette values between 0.51-0.70 versus K

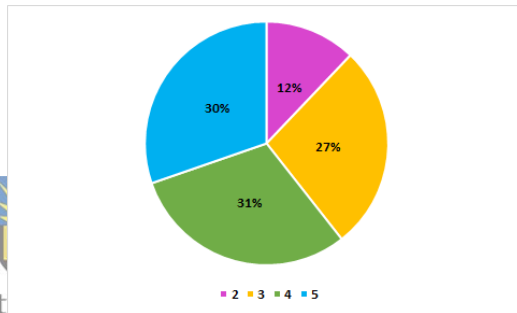
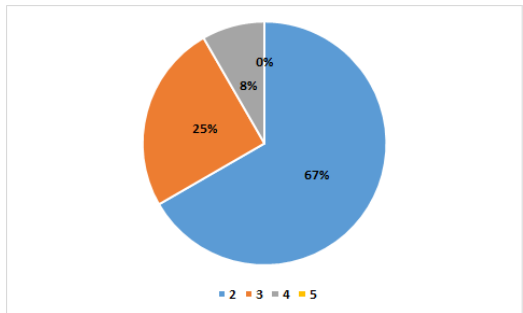
Fig. 5.38: Average Silhouette values for Kmeans and Kmedoids clustering techniques across all test images



(a) Number of average silhouette values between 0.71-1.00 versus K

(b) Number of average silhouette values between 0.51-0.70 versus K

Fig. 5.39: Average Silhouette values for Kmeans clustering techniques across all test images



(a) Number of average silhouette values between 0.71-1.00 versus K

(b) Number of average silhouette values between 0.51-0.70 versus K

Fig. 5.40: Average Silhouette values for Kmedoids clustering techniques across all test images

The Kmeans and Kmedoids clustering techniques obtained equivalent number of average silhouette values for $K=2$ to 5. Hence we present the number of average silhouette values between 0.51-0.70 and those between 0.71-1.00 on the same chart for both the Kmeans and Kmedoids clustering technique. It is observed from Figure 5.38(a) that the when K is increased, the number of images that obtain a strong clustering structure decreases. This is due to the fact that the majority of the test image's average silhouette values favoured the lowest number of clusters for both the kmeans and

Kmedoids clustering technique. Figure 5.38(b) further supports the notion of the average silhouette values favouring the smallest number of clusters, by showing the inverse trend of what is observed in Figure 5.38(a).

The time taken to generate the GMMs by GMM-Kmedoids was comparably smaller than the time taken by GMM-Kmeans for most of the images under the two scenarios. In addition, it was observed that in some instances the GMM-Kmeans performed equivalently to the GMM-Kmedoids under both scenarios. Furthermore, the results obtained showed an upward trend between the average runtime and number of clusters in all images for both the GMM-Kmeans and GMM-Kmedoids. This implies that as the number of clusters is increased the more computational time is required by the algorithms to generate GMMs. The average runtimes ranged from 0,020-0,104 seconds for GMM-Kmeans and 0.019-0.0801 for GMM-Kmedoids under scenario 1, while they ranged from 0,0205-0,0536 seconds for GMM-Kmeans and 0.0193-0.0447 for GMM-Kmedoids under scenario 2. This shows less computation time was required to generate foreground and Background GMMs under scenario 2 than in scenario 1.

The average runtimes for the Grabcuts for image segmentation algorithm with a GMM-Kmeans and GMM-Kmedoids clustering heuristic were measured and reported. It was observed that the Grabcuts for image segmentation algorithm with the GMM-Kmeans obtained the lowest average runtimes of 0.08 seconds amongst all results obtained under scenario 1 and 2. On the other hand, the highest average runtime of 0.15 and 0.26 seconds were obtained by the Grabcuts for image segmentation algorithm with the GMM-

Kmeans clustering heuristic amongst all results obtained under scenario 1 and 2, respectively. The Grabcuts for image segmentation algorithm with the GMM-Kmedoids clustering heuristic obtained the lowest average runtimes of 0.07 and 0.10 seconds amongst all results obtained under scenario 1 and 2, respectively, whereas it obtained the highest average runtimes of 0.26 and 0.15 seconds amongst all results obtained under scenario 1 and 2, respectively. Table 5.11 and 5.12 present a comparison in the average runtime required by the Grabcuts for image segmentation algorithms based on the GMM-Kmeans and GMM-Kmedoids clustering heuristic.

Tab. 5.11: Average runtime required by Grabcuts: GMM-Kmeans Scenario 1 versus Scenario 2

K	Average runtime required in seconds (s)	
	Scenario 1	Scenario 2
2	0.12	0.10
3	0.11	0.12
4	0.12	0.13
5	0.13	0.13

Tab. 5.12: Average runtime required by Grabcuts: GMM-Kmedoids Scenario 1 versus Scenario 2

K	Average runtime required in seconds (s)	
	Scenario 1	Scenario 2
2	0.10	0.10
3	0.11	0.12
4	0.12	0.12
5	0.13	0.12

The images tested in this study are considered to be difficult images [48,54], thus most segmentation results obtained included misclassified pixels either in the foreground or background of the images. This is supported by

the visual quality of the segmented images as well as the precision, recall and BF-scores obtained for the segmented images when K was varied from 2 to 5. It was observed that under scenario 1 38% of the segmentation results suffered with over segmentation, meanwhile 62% of the segmented images were under segmented for both segmentation algorithm considered. In scenario 2, 33% of the segmentation results suffered with over segmentation, meanwhile 67% of the segmented images were under segmented for both segmentation algorithm considered. The researchers limited this study to the initial segmentation, thus the user editing and border matting stages were not considered. The Grabcuts for image with the GMM based on the Kmeans clustering techniques obtained slightly better segmentation results when the visual quality is concerned, than its counterpart under the two scenarios considered. On the other hand, the BF-scores showed that the Grabcuts for image segmentation algorithm with the GMM based on Kmedoids produces images with higher BF-scores than its counterpart when K was varied from 2 to 5 for most of the test images. In addition, most of the images obtained the majority of their best segmentation results when K=2. This was observed to be true under scenario 1 as well as scenario 2. Therefore, the Kmedoids clustering technique with K=2 would be the best option for the segmentation of difficult images in BSDS500. This is due to its ability to generate GMMs and segment difficult images more efficiently (i.e. time complexity, higher BF-scores, more under segmented rather than over segmented images, inter alia.) while producing comparable segmentation results to those obtained by the Grabcuts for image segmentation: GMM-Kmeans.



University of Fort Hare
Rising Above Excellence

6. CONCLUSION

This chapter summarises the research presented in this thesis by giving a summary of the work with major observations seen from the experiments conducted as well as provide possible future work.

6.1 Summary

Image segmentation is the partitioning of a digital image into small segments such as pixels or sets of pixels. It is significant as it allows for the visualization of structures of interest, removing unnecessary information. Image segmentation has seen many applications in different fields such as health-care, construction amongst other fields. There are many image segmentation techniques used depending on the given problem at hand. In this study, the Grabcuts for image segmentation technique was used. This technique is an extension of the graph cut approach. In addition, the Grabcuts for image segmentation technique chooses a segmentation by iteratively revising the set of foreground and background pixels. This approach uses GMMs for the set of foreground and background pixels to determine the statistics of the image and thus model the digital image as a directed weighted graph. The min-cut/max-flow algorithm is then used to arrive at a segmentation of the digital image. The aim of this study is to provide a comparison of cluster-

ing techniques that can be used in generating the GMMs for the Grabcuts for image segmentation algorithm as well as to suggest which of these clustering techniques will yield an efficient and inexpensive Grabcuts for image segmentation algorithm.

The Kmeans and Kmedoids clustering techniques were chosen as the techniques which were used to generate the foreground and background GMMs. In addition, the algorithms developed in this study were run on the CPU with the same test images obtained from the BSDS500 database. The number of clusters used for all the algorithms was varied from 2 to 5 for all test images under the two scenarios considered in this study. It was found that the Kmeans clustering technique outperformed the Kmedoids clustering technique in terms of runtime under the two scenarios considered. In addition, the Kmeans clustering technique found more compact and separate clustering structures for most of the test images than its counterpart under scenario 1, whereas the Kmedoids clustering technique found more compact and separate clustering structures for most of the test images than the Kmeans clustering technique under scenario 2. The silhouette validity index was used to assess the solutions of clustering techniques. The silhouette validity index was seen to favour the smaller number of clusters for most experiments that were run. As a result, most images obtained reasonable to strong clustering structures when $K=2$ under scenario 1 and 2.

Although the Kmeans clustering technique required less computation time for all the experiments that were run, the Grabcuts for image segmentation algorithm with a GMM based on the Kmedoids clustering technique



University of Port Hare
Together in Excellence

and GMM-Kmedoids required less computation time for most test images than their counterparts under the two scenarios considered. In addition, the Grabcuts for image segmentation with the GMM based on the Kmeans clustering techniques obtained slightly better segmentation results when the visual quality is concerned, than its counterpart under the two scenarios considered. On the other hand, the BF-scores showed that the Grabcuts for image segmentation algorithm with the GMM based on Kmedoids produces images with higher BF-scores than its counterpart when K was varied from 2 to 5 for most of the test images. In addition, most of the images obtained the majority of their best segmentation results when K=2. This was observed to be true under scenario 1 as well as scenario 2. Therefore, the Kmedoids clustering technique under scenario 2 with K=2 would be the best option for the segmentation of difficult images in BSDS500. This is due to its ability to generate GMMs and segment difficult images more efficiently (i.e. time complexity, higher BF-scores, more under segmented rather than over segmented images, inter alia.) while producing comparable visual segmentation results to those obtained by the Grabcuts for image segmentation: GMM-Kmeans.

6.2 *Future work*

This study has used several test images from BSDS500 image database to perform experiments. In addition, the number of clusters was varied from 2 to 5 in order to make conclusive suggestions as to which clustering technique ought to be used when performing Grabcuts image segmentation on difficult images. To that effect this study could benefit from testing the algorithms on

the Grabcut database [48]. Furthermore, two distance measures were used to test the algorithms which has shown that changing a distance measure can lead to a change in the clustering solution obtained and hence to different segmentation results. These distance measures can be further extended to include the Jaccard, Hamming and Correlation distance measures. The Grabcut algorithm developed by Rother et al. [48], had a border matting implementation which was not explored in this study, in the future work consideration will be given to the inclusion of border matting implementation. Although the segmentation of videos is not straight forward, the researchers would like to extend the comparison of the Grabcuts for image segmentation with GMM-Kmeans and GMM-Kmedoids to video segmentation. Image segmentation is a very vital process that is used in many fields and thus requires a computationally efficient algorithm. Therefore a reduced runtime of such an algorithm is of paramount importance. The Grabcuts for image segmentation algorithms tested in this study could be improved through the use of the GPU instead of the CPU. Thus, a GPU implementation of the algorithms will be considered.



University of Fort Hare
Together in Excellence

BIBLIOGRAPHY

- [1] D. Adalsteinsson and J. A. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118(2):269–277, 1995.
- [2] S. S. Al-amri, N. V. Kalyankar, and K. S. D. Image segmentation by using threshold techniques. *Journal of Computing*, 2(5):83–86, 2010.
- [3] B. Anenberg and M. Meister. Interactive image segmentation with grabcut. Technical report, Stanford University, 2015. Available: <http://ai.stanford.edu/~anenberg/papers>.
- [4] Z. Ansari, W. A. M.F. Azeem, and A. Babu. Quantitative evaluation of performance and validity indices for clustering the web navigational sessions. *World of Computer Science and Information Technology Journal*, 1(5):217–226, 2011.
- [5] N. Arbin, N. S. N.Z. Mokhtar, and Z. Othman. Comparative analysis between k-means and k-medoids for statistical clustering. In *Proceedings - AIMS 2015, 3rd International conference on Artificial Intelligence, Modelling and Simulation*, pages 117–121. IEEE, 2016.
- [6] P. Arora, V. Deepali, and S. Varshney. Analysis of k-means and k-

- medoids algorithm for big data. *Procedia Computer Science*, 78:507–512, 2016.
- [7] G. Barlas. *Multicore and GPU Programming: An Integrated Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 2015.
- [8] D. J. Bora and A. K. Gupta. Effect of different distance measures on the performance of k-means algorithm: An experimental study in matlab. *International Journal of Computer Science and Information Technologies*, 5(2):2501–2506, 2014.
- [9] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *8th IEEE International Conference on Computer Vision*, pages 105–112. IEEE, 2001.
- [10] Y. Boykov and O. Veksler. *Graph Cuts in Vision and Graphics: Theories and Applications*, pages 79–96. Springer US, 2006.
- [11] R. Canetti, O. Poburinnaya, and M. Venkatasubramaniam. Better two-round adaptive multi-party computation. In *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings Part II*, pages 396–427. Springer-Verlag, 2017.
- [12] J. S. Cardoso and L. Corte-Real. Toward a generic evaluation of image segmentation. *IEEE Transactions on Image Processing*, 14(11):1773–1782, 2005.
- [13] B. G. Chandran and D. S. Hochbaum. A computational study of the

- pseudoflow and push-relabel algorithms for the maximum flow problem. *Operations Research*, 57(2):358–376, 2009.
- [14] T. Collins. Graph cut matching in computer vision. *University of Edinburgh*, 2004. Available: <https://pdfs.semanticscholar.org>.
- [15] J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices, and Groups*. Springer-Verlag, New York, 2nd edition, 1993.
- [16] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, Cambridge, 2nd edition, 2001.
- [17] I. Damski and A. Sharfshtein. Implementing the “grabcut” segmentation technique. Available: <http://www1.idc.ac.il/toky>, n.d.
- [18] R. Dass and S. Devi. Image segmentation techniques 1. *IJECT*, 3(1):66–70, 2012.
- [19] S. P. Deenan and J. SatheeshKumar. Performance evaluation of image segmentation using objective methods. *Indian Journal of Science and Technology*, 9(8), 2016.
- [20] R. Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 3rd edition, 2005.
- [21] G. Dougherty. *Digital image processing for medical applications*. Cambridge University Press, Cambridge, 2009.
- [22] E. Fernandez-Moral, R. Martins, D. Wolf, and P. Rives. A new metric for evaluating semantic segmentation: leveraging global and contour

- accuracy. In *Workshop on Planning, Perception and Navigation for Intelligent Vehicles, PPNIV17*, Vancouver, Canada, 2017. Available: <https://hal.inria.fr/hal-01581525>.
- [23] G. Gandhi and R. Srivastava. Analysis and implementation of modified k-medoids algorithm to increase scalability and efficiency for large dataset. *International Journal of Research in Engineering and Technology*, 3(6):150–153, 2014.
- [24] K. Georgieva. *A computational intelligence approach to clustering of temporal data*. MSc Thesis, University of Pretoria, 2014. Available: <http://respository.up.ac.za/handle/2263/43778>.
- [25] S. Han, W. Tao, D. Wang, X. Tai, and X. Wu. Image segmentation based on grabcut framework integrating multiscale nonlinear structure tensor. *IEEE Transactions on Image Processing*, 18(10):2289–2302, 2009.
- [26] S. Islam and M. Ahmed. Implementation of image segmentation for natural images using clustering methods. *International Journal of Emerging Technology and Advanced Engineering*, 3(3):175–180, 2013.
- [27] S. Jayaraman, S. Esakkirajan, and T. Veerakima. *Digital image processing*. Tata McGraw Hill Education private limited, New Delhi, 1st edition, 2009.
- [28] Y. Kalantidis, L. Kennedy, and L.-J. Li. Getting the look: Clothing recognition and segmentation for automatic product suggestions in everyday photos. In *Proceedings of the 3rd ACM Conference on International Conference on Multimedia Retrieval*, pages 105–112. ACM, 2013.

- [29] G. Karypis and V. Kumar. Parallel multilevel series k-way partitioning scheme for irregular graphs. *Siam Review*, 41(2):278–300, 1999.
- [30] M. R. Khokher, A. Ghafoor, and A. M. Siddiqui. Image segmentation using multilevel graph cuts and graph development using fuzzy rule-based system. *IET Image Processing*, 7(3):201–211, 2013.
- [31] D. B. Kirk and W. W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1st edition, 2010.
- [32] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts?. In *Proceedings of the 7th European Conference on Computer Vision-Part III*, pages 65–81. Springer-Verlag, 2002.
- [33] V. Kumar, J. Chhabra, and D. Kumar. Performance evaluation of distance metrics in clustering algorithms. *INFOCOMP Journal of Computer Science*, 13(1):38–52, 2014.
- [34] S. Lalwani. Push relabel algorithm: Set 1 (introduction and illustration). Available: <http://www.geeksforgeeks.org/push-relabel-algorithm-set-1-introduction-and-illustration>, n.d.
- [35] G. Lee, S. Lee, G. Kim, J. Park, and Y. Park. A modified grabcut using a clustering technique to reduce image noise. *Symmetry*, 8(7):64, 2016.
- [36] H. Lombaert, Y. Sun, L. Grady, and C. Xu. A multilevel banded graph cuts method for fast image segmentation. In *Computer Vision, 2005*.

- ICCV 2005. Tenth IEEE International Conference on*, pages 259–265. IEEE, 2005.
- [37] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the 8th International Conference in Computer Vision*, pages 416–423. IEEE, 2001.
- [38] D. Morley and C. S. Parker. *Understanding computers: Today and tomorrow, introductory*. Cengage Learning, Stamford, 15th edition, 2015.
- [39] R. Muthukrishnan and M. Radha. Edge detection techniques for image segmentation. *International Journal of Computer Science & Information Technology*, 3(6):259, 2011.
- [40] M. Mützell and M. Josefsson. *Max Flow Algorithms : Ford-Fulkerson, Edmond-Karp, Goldberg-Tarjan Comparison in regards to practical running time on different types of randomized flow networks*. KTH, School of Computer Science and Communication (CSC), 2015.
- [41] B. R. Naidu, P. L. Rao, M. P. Babu, and K. Bhavani. Framework for efficient edge detection techniques- comparison among robert, prewitt, sobel, robinson, kirsch and canny. *International Journal of Computer Science and Technology*, 3(2):2229–4333, 2012.
- [42] P. Pahl, F. Pahl, and R. Damrath. *Mathematical Foundations of Computational Engineering: A Handbook*. Springer, Berlin, 2012.



University of Fort Hare
Together in Excellence

- [43] N. R. Pal and S. K. Pal. A review on image segmentation techniques. *Pattern recognition*, 26(9):1277–1294, 1993.
- [44] S. Pandit and S. Gupta. A comparative study on distance measuring approaches for clustering. *International Journal of Research in Computer Science*, 2(1):29–31, 2011.
- [45] B. Peng, L. Zhang, and D. Zhang. A survey of graph theoretical approaches to image segmentation. *Pattern Recognition*, 46(3):1020–1038, 2013.
- [46] E. Pourjam, I. Ide, D. Deguchi, and H. Murase. Segmentation of human instances using grab-cut and active shape model feedback. In *Machine Vision Applications*, pages 77–80. MVA, 2013.
- [47] J. Ramírez, P. Temoche, R. Carrionna, et al. A volume segmentation approach based on grabcut. *CLEI Electronic Journal*, 16(2):4, 2013.
- [48] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.
- [49] J. Sachs. Digital image basics. *Digital Light & Color*, 1996.
- [50] S. Saini and K. Arora. A study analysis on the different image segmentation techniques. *International Journal of Information & Computation Technology*, 4(14):1445–1452, 2014.
- [51] J. Salvatore. Bipartite graphs and problem solving. *University of*

Chicago, 2007. Available: <http://www.math.uchicago.edu/may/VIGRE/VIGRE2007/REUPapers/FINALAPP>.

- [52] S. J. Sangwine and R. E. Horne. *The colour image processing handbook*. Springer Science & Business Media, New York, 2012.
- [53] S. N. Sinha. Graph cut algorithms in vision, graphics and machine learning an integrative paper. *UNC Chapel Hill*, 2004. Available: <https://pdfs.semanticscholar.org/1e3e/e2bfb3535d79d0dea81c0c5ce08174c6a994.pdf>.
- [54] J. F. Talbot and X. Xu. Implementing grabcut. *Brigham Young University*, 2006. Available: <https://pdfs.semanticscholar.org/9246>.
- [55] R. B. Thompson and B. F. Thompson. *PC hardware in a nutshell: a desktop quick reference*. O'Reilly Media, Inc., Sebastopol, CA, 3rd edition, 2003.
- [56] J. K. Udupa, V. R. Leblanc, Y. Zhuge, C. Imielinska, H. Schmidt, L. M. Currie, B. E. Hirsch, and J. Woodburn. A framework for evaluating image segmentation algorithms. *Computerized Medical Imaging and Graphics*, 30(2):75–87, 2006.
- [57] J. S. Weszka and A. Rosenfeld. Threshold evaluation techniques. *IEEE Transactions on systems, man, and cybernetics*, 8(8):622–629, 1978.
- [58] R. J. Wilson. *Introduction to graph theory*. Longman Inc., New York, 4th edition, 1996.
- [59] Y. Yang and S. Huang. Image segmentation by fuzzy c-means clus-

tering algorithm with a novel penalty term. *Computers and Artificial Intelligence*, 26(1):17–31, 2007.

- [60] Y. Zhang. A review of recent evaluation methods for image segmentation. In *Signal Processing and its Applications, Sixth International Symposium on. 2001*, pages 148–151. IEEE, 2001.



University of Fort Hare
Together in Excellence

APPENDIX



University of Fort Hare
Together in Excellence

A. CENTRAL PROCESSING UNIT VERSUS THE GRAPHICS PROCESSING UNIT

We now look at the central processing and graphics processing units so that we can better understand the environment in which the algorithms are executed. In this chapter we will give definitions of a Graphics Processing Unit(GPU) and Central Processing Unit(CPU). In addition, we shall make a comparison between the GPU and CPU.



A.1 *The central processing unit*

University of Fort Hare
Together in Excellence

Definition A.1.1. [55] *The central processing unit is a component in the computer that is responsible for interpreting computer program instructions and processing data.*


Thus the CPU is seen as the brains of the computer and it is said to be the most expensive component of the computer. The CPU is located in a single silicon chip called the microprocessor.

CPUs are rated based on several attributes such as clock rate, computer word size, main memory size, cache memory size, instruction set complexity, bus speed and the number of processing units. In addition, the speed of the CPU is a significant factor that determines the usefulness of a computer.

This speed is determined by the clock frequency as well as the cache memory available.

In order to understand in depth how the speed of the CPU speaks to the usefulness of the computer, let us investigate the core components of a CPU according to Thompson [55].

- Transistors:

These are the fundamental building block of the circuitry that governs the operation of a computer and all other modern electronics. The number of transistors in a CPU has an immense effect on the CPU. In addition, they bring about the possibilities of more powerful multipliers capable of single-cycle speeds. Furthermore, more transistors in a CPU enable a technology called  pipelining. Pipelining is the process of overlapping of instruction execution. That is, the execution of the next instruction starts before the end of the execution of the previous instruction.

- Registers

These are the small amount of very fast computer memory which are found inside a CPU. Registers are not part of the main memory as the CPU implements them on-chip.

Definition A.1.2. *Registers are small holding areas on the processor chip that work much like Random Access Memory(RAM). Registers hold counters, data, instructions and addresses that the Arithmetic Logic Unit(ALU) is currently processing.*

Thus, registers are special memory locations that are internal to the CPU. In addition, registers provide the fastest way for a CPU to access data and are used to speed the execution of computer programs. Registers can be categorized according to their functions. Table A.1 shows a list of registers that are used to perform specific functions discussed in [55].

Tab. A.1: Types of registers and their functions

Register	Functions
Accumulator	Used for arithmetic, logic, or other similar operations
Data register	Used for temporary storage of data
Address register	Stores the addresses of specific memory locations
General purpose register	Can be used as either address or data registers
Control register	Used to control some aspect of processor operation
Program counter	Points to the next executable instruction's memory location



University of Fort Hare
Together in Excellence

- Clock

The clock speed sometimes referred to as the clock rate is the speed at which the CPU executes instructions and is rated in megahertz (MHz). Every computer has an internal clock that is responsible for regulating the rate at which instructions are executed and synchronizing all the various computer components. In addition, the CPU requires a fixed number of clock cycles to execute each instruction. Furthermore, a CPU with a faster clock executes more instructions per second as opposed to one with a slower clock.

Now, before we can start to give a comparison between the CPU and GPU we need to understand the way CPU's work. We will do this by detail-

ing the four operations of the CPU.

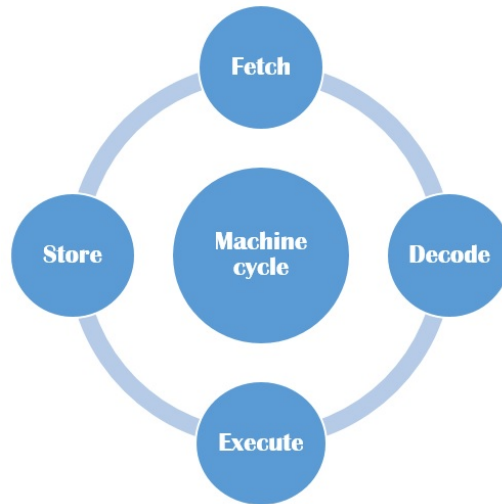


Fig. A.1: Machine cycle of a CPU [38].

Figure A.1 shows the machine cycle as described by Morley et al. [38]. The machine cycle consists of four steps and we shall describe each step below.



University of Twente
Together in Excellence

Fetch:

This step involves the retrieval of instructions from memory. This memory can either be RAM or cache memory. The Control Unit (CU) requests the memory to provide it with the instruction stored in the memory location specified by the program counter. This instruction is then stored in the Instruction Register(IR).

Decode:

The instruction stored in the IR is then translated into chains of computer commands. This process is also done in the CU. In addition, the CU interprets the chains of commands and decides what is to be done.

Execution:

The appropriate circuitry is then activated so as to perform the task required. This is due to the CU sending signals to the Arithmetic Logic Unit(ALU) that enable the carrying out of the correct tasks.

Storing:

This step involves storing of the data or results from the execution that are stored in the registers or RAM. This data or results can be used as input for another computer program.

A.2 The graphics processing unit

According to Barlas [7], the GPU is a dedicated parallel processor optimized for accelerating graphical computations. It is produced in vast numbers for computer graphics and is a single-chip processor like the CPU. It is increasingly being used for computer games, video, audio, medical field and multimedia amongst other. A GPU can have up to 512 cores on single chip [7]. Thus, they have simplified logic since much more of the chip is devoted to floating- point computation. GPU's may be arranged as multiple units with each unit being effectively a vector unit, all cores doing the same thing at the same time. In addition, they have very high bandwidths of up to 190 GB/s and graphics memory of up to 6GB.



University of Port Harcourt
Together in Excellence

A.2.1 Parallelism in GPU and GPU architecture

Kirk et al. [31], maintains that massively parallel programming is motivated by the increased demand for high running speeds in applications. The researchers also argue that a good implementation on a GPU can achieve more than 100 times speedup over its sequential execution. According to Kirk et al. [31], there are two types of parallelism, viz. data and task parallelism. Data parallelism is the ability of a program to execute many arithmetic operations on data structures simultaneously. on the other hand, task parallelism also known as function parallelism refers to the ability of a program to execute different operations in parallel to fully utilize the available processors and memory.



A.2.2 Influences on evolution of the CPU and GPU

University of Fort Hare

Together in Excellence

There are several factors that influence the evolution or development of CPUs and GPUs. These include the following;

1. Technology
2. Theory and design ingenuity
3. User demand
4. Economics and commercial pressure

A.3 CPU versus GPU

There are many researchers that have done a comparison between the CPU and GPU [7, 31, 38, 47]. They allude to the fact that GPU is the best in

terms of large data parallel processing. Although this is the case, some say that these researchers have compared general CPUs with developed GPUs. According to Kirk et al. [31], the semiconductor industry has settled on two categories of designing microprocessors namely; the multicore and many-core microprocessor designs. The multicore design focuses on maintaining the execution speed of sequential programs while moving into multiple cores, whereas the many-cores design seeks to maintain the execution throughput of parallel applications. The Intel[®] Core[™] i7 microprocessor which has four processor cores is an example of a multicore microprocessor, while the NVidia[®] GeForce GTX280 GPU with 240 cores is an example of many-cores microprocessor. Kirk et al. [31] maintains that, the performance of many-core GPUs has improved significantly as compared to that of multicore CPUs. This can be accredited to the fundamental design philosophies between the two types of processors. The CPU design is optimized for sequential code performance and it uses a sophisticated control logic to allow instructions from a single thread of execution to execute in parallel or even out of their sequential order while maintaining the appearance of sequential execution. In addition, large cache memories are provided to minimize the instruction and data access latencies of large complex applications.

The attributes that make GPU processing a more suitable candidate for executing complex programs are shown in Table A.2. According to [38], the CPU can have up to eight cores, for instance, dual-core or quad-core, whilst, the GPU can have up to 512 cores. In addition, the GPU has been said to be

Tab. A.2: CPU versus GPU.

CPU	GPU
Can have up to 8 cores	Can have up to 512 cores
Executes programs serially	Highly parallel execution
Has fewer execution units	Has many execution units
	Has faster memory interfaces

a parallel processor while the CPU commands serial execution of programs [31]. Moreover, the GPU experiences higher memory bandwidths than the CPU due to fewer legacy constraints [31]. Hence GPUs have faster memory interfaces.

The knowledge of the CPU and GPU underpins the understanding of some of the observations made by several researchers in terms of time and memory efficiencies experienced when running image segmentation algorithms [9, 10, 21, 47, 48]. Now that we have learnt about some of the grabcut algorithms, the CPU and GPU, let us consider the approach presented in this study.



University of Port Harcourt
Together in Excellence

B. GRABCUTS FOR IMAGE SEGMENTATION MATLAB

CODE

B.1 Grabcuts for image segmentation: GUI

```
1 function varargout = Grabcut_gui(varargin)
2 % GRABCUT_GUI M-file for Grabcut_gui.fig
3 %     GRABCUT_GUI, by itself , creates a new GRABCUT_GUI or
4 %     raises the existing
5 %     singleton*.
6 %     H = GRABCUT_GUI returns the handle to a new GRABCUT_GUI
7 %     or the handle to
8 %     the existing singleton*.
9 %     GRABCUT_GUI('CALLBACK', hObject , eventData , handles ,...)
10 %     calls the local
11 %     function named CALLBACK in GRABCUT_GUI.M with the given
12 %     input arguments.
13 %     GRABCUT_GUI('Property' , 'Value' ,...) creates a new
14 %     GRABCUT_GUI or raises the
15 %     existing singleton*. Starting from the left , property
16 %     value pairs are
```

```

14 %     applied to the GUI before Grabcut_gui_OpeningFunction
    gets called. An
15 %     unrecognized property name or invalid value makes
    property application
16 %     stop. All inputs are passed to Grabcut_gui_OpeningFcn
    via varargin.
17 %
18 %     *See GUI Options on GUIDE's Tools menu. Choose "GUI
    allows only one
19 %     instance to run (singleton)".
20 %ann
21 % See also: GUIDE, GUIDATA, GUIHANDLES
22
23 % Copyright 2002–2003 The MathWorks, Inc.
24
25 % Edit the above text to modify the response to help Grabcut_gui
26
27 % Last Modified regression by GUIDE v2.5 25–Mar–2018 12:07:07
28
29 % Begin initialization code – DO NOT EDIT
30 gui_Singleton = 1;
31 gui_State = struct('gui_Name',       mfilename, ...
32                   'gui_Singleton',  gui_Singleton, ...
33                   'gui_OpeningFcn', @Grabcut_gui_OpeningFcn,
34                   ...
35                   'gui_OutputFcn',  @Grabcut_gui_OutputFcn, ...
36                   'gui_LayoutFcn',  [] , ...
37                   'gui_Callback',   []);
38 if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});

```



```

39 end
40
41 if nargin
42     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:})
43     ;
44 else
45     gui_mainfcn(gui_State, varargin{:});
46 end
47
48 % End initialization code – DO NOT EDIT
49
50 % — Executes just before Grabcut_gui is made visible.
51 function Grabcut_gui_OpeningFcn(hObject, eventdata, handles,
52     varargin)
53 % This function has no output args, see OutputFcn.
54 % hObject    handle to figure
55 % eventdata  reserved – to be defined in a future version of
56 %            MATLAB
57 % handles    structure with handles and user data (see GUIDATA)
58 % varargin   command line arguments to Grabcut_gui (see VARARGIN
59             )
60
61 % Choose default command line output for Grabcut_gui
62 handles.output = hObject;
63
64 % Update handles structure
65 guidata(hObject, handles);
66
67 % UIWAIT makes Grabcut_gui wait for user response (see UIRESUME)
68 % uiwait(handles.figure1);
69
70

```

```

65
66 % —— Outputs from this function are returned to the command
    line .
67 function varargout = Grabcut_gui_OutputFcn(hObject , eventdata ,
    handles)
68 % varargout    cell array for returning output args (see VARARGOUT
    );
69 % hObject      handle to figure
70 % eventdata    reserved – to be defined in a future version of
    MATLAB
71 % handles      structure with handles and user data (see GUIDATA)
72
73 % Get default command line output from handles structure
74 varargout{1} = handles.output ;
75 image=imread( '2.png' );
76 axes(handles.axes4);
77 imshow(image);
78
79
80 % —— Executes on button press in sl_im .
81 function sl_im_Callback(hObject , eventdata , handles)
82 % hObject      handle to sl_im (see GCBO)
83 % eventdata    reserved – to be defined in a future version of
    MATLAB
84 % handles      structure with handles and user data (see GUIDATA)
85 global im bw L enMap dc sc vC hC L1 enMap1 dc3 sc1 vC1 hC1 ;
86 axes(handles.axes1);
87 im = imgetfile ();
88 im = imread(im);
89 image(im);

```

```

90
91
92
93 % — Executes on button press in select_roi.
94 function select_roi_Callback(hObject, eventdata, handles)
95 % hObject    handle to select_roi (see GCBO)
96 % eventdata  reserved - to be defined in a future version of
    MATLAB
97 % handles    structure with handles and user data (see GUIDATA)
98 global im bw L enMap dc sc vC hC L1 enMap1 dc3 sc1 vC1 hC1 ;
99
100 axes(handles.axes1);
101 h = imfreehand(gca);
102 position = wait(h);
103 bw= h.createMask();
104 bw = bw -1;
105
106 % — Executes on button press in rub_grabcut.
107 function rub_grabcut_Callback(hObject, eventdata, handles)
108 % hObject    handle to rub_grabcut (see GCBO)
109 % eventdata  reserved - to be defined in a future version of
    MATLAB
110 % handles    structure with handles and user data (see GUIDATA)
111 global im bw L enMap dc sc vC hC L1 enMap1 dc3 sc1 vC1 hC1 ;
112
113 imd = im;
114 imd = double(imd);
115
116 tic;
117 [L, dc , sc , vC, hC] = GrabCut(imd, bw);

```

```

118 toc ;
119
120 tic ;
121 [L1, dc3 , sc1 , vC1, hC1] = GrabCut_med(imd, bw);
122 toc ;
123
124 Ld = double(L); Ld1=double(L1);
125 Ld = L + 1; Ld1=L1+1;
126 Ld = double(Ld /2); Ld1=double(Ld1/2);
127 enMap = zeros(size(Ld)); enMap1=zeros(size(Ld1));
128 resIm = imd; resIm1=imd;
129 resIm(:,:,1) = imd(:,:,1) .* Ld; resIm1= imd(:,:,1) .* Ld1;
130 resIm(:,:,2) = imd(:,:,2) .* Ld; resIm1(:,:,2) = imd(:,:,2) .*
    Ld1;%resIm2(:,:,2) = imd(:,:,2) .* Ld2;
131 resIm(:,:,3) = imd(:,:,3) .* Ld; resIm1(:,:,3) = imd(:,:,3) .*
    Ld1;%resIm2(:,:,3) = imd(:,:,3) .* Ld2;
132 axes(handles.axes3);
133 image(uint8(resIm));
134 imwrite((uint8(resIm)), 'Me.png');
135 axes(handles.axes5);
136 image(uint8(resIm1));
137 imwrite((uint8(resIm1)), 'ME1.png');
138
139 % — Executes on button press in run_enhance.
140 function run_enhance_Callback(hObject, eventdata, handles)
141 % hObject    handle to run_enhance (see GCBO)
142 % eventdata  reserved – to be defined in a future version of
    MATLAB
143 % handles    structure with handles and user data (see GUIDATA)
144 global im bw L enMap dc sc vC hC L1 enMap1 dc3 sc1 vC1 hC1 ;

```

```

145
146 indxs = find(enMap);
147 L(indxs) = enMap(indxs);
148 dc1 = dc(:, :, 1);
149 dc2 = dc(:, :, 2);
150 findxs = find(enMap > 0);
151 dc1(findxs) = 1000;
152 bindxs = find(enMap < 0);
153 dc2(bindxs) = 1000;
154 dc = cat(3, dc1, dc2);
155     gch = GraphCut('open', dc, sc, vC, hC);
156     gch = GraphCut('set', gch, int32(L==1)); % initial guess -
        previous result
157     [gch L] = GraphCut('expand', gch);
158     L = (2*L-1); % convert {0,1} to {-1,1} labeling
159     gch = GraphCut('close', gch);
160 Ld = double(L);
161 Ld = L + 1;
162 Ld = double(Ld /2);
163 resIm = double(im);
164 resIm(:, :, 1) = resIm(:, :, 1) .* Ld;
165 resIm(:, :, 2) = resIm(:, :, 2) .* Ld;
166 resIm(:, :, 3) = resIm(:, :, 3) .* Ld;
167 axes(handles.axes2);
168 image(uint8(resIm));
169 imwrite((uint8(resIm)), 'EnKmeans.png');
170
171 % —— Executes on button press in sl_bg.
172 function sl_bg_Callback(hObject, eventdata, handles)
173 % hObject     handle to sl_bg (see GCBO)

```

```

174 % eventdata reserved – to be defined in a future version of
    MATLAB
175 % handles structure with handles and user data (see GUIDATA)
176 global im bw L enMap dc sc vC hC L1 dc3 sc1 vC1 hC1 enMap1;
177 axes(handles.axes3);
178 bg_bw = -1 * roipoly();
179 enMap = -1*((enMap + bg_bw) < 0) + (enMap > 0);
180 tempRes = select_enhance(im, enMap);
181 axes(handles.axes1);
182 image(uint8(tempRes));
183
184
185 % — Executes on button press in sl_fg.
186 function sl_fg_Callback(hObject, eventdata, handles)
187 % hObject handle to sl_fg (see GCBO)
188 % eventdata reserved – to be defined in a future version of
    MATLAB
189 % handles structure with handles and user data (see GUIDATA)
190 global im bw L enMap dc sc hC L1 dc3 sc1 vC1 hC1 enMap1;
191 axes(handles.axes3);
192 fg_bw = roipoly();
193 enMap = ((enMap + fg_bw) > 0) - (enMap < 0);
194 tempRes = select_enhance(im, enMap);
195 axes(handles.axes1);
196 image(uint8(tempRes));
197
198
199
200 % — Executes on button press in pushbutton7.
201 function pushbutton7_Callback(hObject, eventdata, handles)

```

```

202 % hObject    handle to pushbutton7 (see GCBO)
203 % eventdata  reserved - to be defined in a future version of
      MATLAB
204 % handles    structure with handles and user data (see GUIDATA)
205
206
207 % —— Executes on button press in exit_btn.
208 function exit_btn_Callback(hObject, eventdata, handles)
209 % hObject    handle to exit_btn (see GCBO)
210 % eventdata  reserved - to be defined in a future version of
      MATLAB
211 % handles    structure with handles and user data (see GUIDATA)
212
213 close all;
214
215 function enIm = select_enhance(im, map)
216 tempRes = im;
217 findxs = find(map>0);
218 red = tempRes(:, :, 1);
219 red(findxs) = 255;
220 green = tempRes(:, :, 2);
221 green(findxs) = 0;
222 blue = tempRes(:, :, 3);
223 blue(findxs) = 0;
224
225 bindxs = find(map<0);
226 red(bindxs) = 0;
227 green(bindxs) = 0;
228 blue(bindxs) = 255;
229

```

```

230 tempRes(:, :, 1) = red;
231 tempRes(:, :, 2) = green;
232 tempRes(:, :, 3) = blue;
233 enIm = tempRes;
234
235
236 % — Executes on button press in Select_f.
237 function Select_f_Callback(hObject, eventdata, handles)
238 % hObject    handle to Select_f (see GCBO)
239 % eventdata  reserved — to be defined in a future version of
    MATLAB
240 % handles    structure with handles and user data (see GUIDATA)
241 global im bw L enMap dc sc vC hC L1 dc3 sc1 vC1 hC1 enMap1;
242 axes(handles.axes5);
243 fg_bw1 = roipoly();
244 enMap1 = ((enMap1 + fg_bw1) > 0) - (enMap1 < 0);
245 tempRes1 = select_enhance(im, enMap1);
246 axes(handles.axes1);
247 image(uint8(tempRes1));
248
249 % — Executes on button press in Select_b.
250 function Select_b_Callback(hObject, eventdata, handles)
251 % hObject    handle to Select_b (see GCBO)
252 % eventdata  reserved — to be defined in a future version of
    MATLAB
253 % handles    structure with handles and user data (see GUIDATA)
254 global im bw L enMap dc sc vC hC L1 dc3 sc1 vC1 hC1 enMap1;
255 axes(handles.axes5);
256 bg_bw1 = -1 * roipoly();
257 enMap1 = -1*((enMap1 + bg_bw1) < 0) + (enMap1 > 0);

```



```

258 tempRes1 = select_enhance(im, enMap1);
259 axes(handles.axes1);
260 image(uint8(tempRes1));
261
262 % —— Executes on button press in Enhance_Kmed.
263 function Enhance_Kmed_Callback(hObject, eventdata, handles)
264 % hObject    handle to Enhance_Kmed (see GCBO)
265 % eventdata  reserved - to be defined in a future version of
    MATLAB
266 % handles    structure with handles and user data (see GUIDATA)
267 global im bw L enMap dc sc vC hC L1 enMap1 dc3 sc1 vC1 hC1 ;
268
269 indxs1 = find(enMap1);
270 L1(indxs1) = enMap1(indxs1);
271 dc4 = dc3(:, :, 1);
272 dc5 = dc3(:, :, 2);
273 findxs1 = find(enMap1 > 0);
274 dc4(findxs1) = 1000;
275 bindxs1= find(enMap1 < 0);
276 dc5(bindxs1) = 1000;
277 dc3 = cat(3, dc4, dc5);
278     gch1 = GraphCut('open', dc3 , sc1 , vC1, hC1);
279     gch1 = GraphCut('set', gch1, int32(L1==1)); % initial guess
    - previous result
280     [gch1 L1] = GraphCut('expand', gch1);
281     L1 = (2*L1-1); % convert {0,1} to {-1,1} labeling
282     gch1 = GraphCut('close', gch1);
283 Ld1 = double(L1);
284 Ld1 = L1 + 1;
285 Ld1 = double(Ld1 /2);

```

```

286 resIm1 = double(im);
287 resIm1(:, :, 1) = resIm1(:, :, 1) .* Ld1;
288 resIm1(:, :, 2) = resIm1(:, :, 2) .* Ld1;
289 resIm1(:, :, 3) = resIm1(:, :, 3) .* Ld1;
290 axes(handles.axes6);
291 image(uint8(resIm1));
292 imwrite((uint8(resIm1)), 'EnKmedoids.png');

```

B.2 Grabcuts based on GMM-Kmeans algorithm

```

1 function [L, dc , sc , vC, hC] = GrabCut(im, initMap)
2 %
3 % Performs segmentation of image into foreground/background
4 % regions
5 %
6 % Usage:
7 %   L = GrabCut(im, initMap)
8 %
9 % Inputs:
10 %   im – color image to be segmented
11 %   initMap – initial labeling of which we are _certain_ of:
12 %           1 – FG
13 %          -1 – BG
14 %           0 – uncertain
15 %
16 % Output:
17 %   L – a segmentation (i.e., {0,1} labeling) of the give image.
18 %
19 % This implementation follows the paper
20 % Rother C., Kolmogorov V. and Blake A. "GrabCut"–Interactive

```

```

21 % Foreground Extraction using iterated Graph Cuts. SIGGRAPH
    2004.
22 %
23
24 % constants:
25 Gamma = 50;
26 MaxItr = 100;
27 K = 6; % number of components in each mixture model
28 INF = 1000;
29
30
31 % working in Lab space
32 labim = im; % RGB2Lab(im);
33
34 [hC vC] = SmoothnessTerm(labim);
35 sc = Gamma.*[0 1; 1 0];
36
37 certain_fg = initMap == 1;
38 certain_bg = initMap == -1;
39
40 oL = initMap;
41
42 % how to label all the uncertain pixels?
43 if (sum(sum(abs(certain_fg))) == 0)
44     oL(initMap==0) = 1;
45 else
46     oL(initMap==0) = -1;
47 end
48
49 done_suc = false;

```

```

50
51 for itr=1:MaxItr ,
52
53 % GMM for foreground and background – global model
54 tic ;
55 logpFG = LocalColorModel(labim , K, oL==1);
56 logpBG = LocalColorModel(labim , K, oL==–1);
57 toc ;
58
59 % force labeling of certain labeling
60 logpBG(certain_fg) = INF;
61 logpFG(certain_bg) = INF;
62
63 dc = cat(3, logpBG, logpFG);
64 gch = GraphCut('open', dc , sc , vC, hC);
65 gch = GraphCut('set', gch, int32(oL==1)); % initial guess –
66 previous result
67 [gch L] = GraphCut('expand', gch);
68 L = (2*L–1); % convert {0,1} to {–1,1} labeling
69
70 % stop if converged
71 if sum(oL(:)~=L(:)) < .001* numel(L)
72     done_suc = true;
73     break;
74 end
75
76 oL = L;
77
78 end

```

```

79
80 if ~done_suc
81     warning('GrabCut:GrabCut','Failed to converge after %d
      iterations', itr);
82 end

```

B.3 Local colour model generated by GMM-Kmeans

```

1 function logp = LocalColorModel(labim, K, select)
2 % estimate global color model
3 % labim - image patch in Lab space
4 % K - number of component in mixture model
5 % select - what pixels in patch belongs to region
6
7 imsiz = size(labim);
8
9 imR = labim(:, :, 1);
10 imG = labim(:, :, 2);
11 imB = labim(:, :, 3);
12
13 fullSize = prod(imsiz(1:2));
14
15 vectIm = [reshape(imR, [1 fullSize]); reshape(imG, [1 fullSize])
           ; reshape(imB, [1 fullSize])];
16
17 selected = find(select);
18 imR = imR(selected);
19 imG = imG(selected);
20 imB = imB(selected);
21 fg = [imR, imG, imB];

```

```

22
23 rng('default');
24 tic;
25 [idx, mu, Sumd, D] = kmeans(fg, K, 'distance', 'cityblock', 'Display
    ', 'iter');
26 toc;
27 disp('Intra cluster mean distance kmeans');
28 disp(sum(Sumd));
29 disp('Inter cluster mean distance kmeans');
30 disp(sum(D(:)));
31
32 x= fg(:,1);
33 y=fg(:,2);
34
35
36 figure;
37 cla;
38 gca;
39 hold on
40 colors='rgckym';
41
42 for num=1:K
43 plot(x(idx==num),y(idx==num),[colors(num) ' . ']);
44
45 end
46 plot(mu(:,1),mu(:,2), 'bx', ...
47       'MarkerSize',15, 'LineWidth',3);
48 legend('Cluster 1', 'Cluster 2', 'Centroids', ...
49        'Location', 'NW')
50 title 'Cluster Assignments and Centroids for kmeans'

```

```

51 hold off
52
53     tic;
54     si=silhouette(fg, idx, 'sqeuclidean');
55     mean_si= mean(si);
56     toc;
57     disp('mean');
58 disp(mean_si);
59 % mu - mean
60 % cvr - covariance
61 % wi - component weight
62
63 % compute data cost according to model
64
65 logp = zeros(prod(imsiz(1:2)),K);
66 for ki=1:K
67     cvr = cov(fg(idx==ki,:));
68     if rank(cvr) < imsiz(3)
69         logp(:,ki) = inf;
70         continue;
71     end
72     wi = sum(idx==ki)./size(fg,1);
73
74     df = vectIm';
75     df(:, 1) = df(:, 1) - mu(ki,1);
76     df(:, 2) = df(:, 2) - mu(ki,2);
77     df(:, 3) = df(:, 3) - mu(ki,3);
78
79     logp(:,ki) = -log(wi)+.5*log(det(cvr)) + ...
80         .5*sum(df*inv(cvr).*df, 2);

```

```

81 end
82
83 logp = min(logp, [], 2);
84 logp = reshape(logp, imsize(1:2));

```

B.4 Grabcuts based on GMM-Kmedoids algorithm

```

1 function [L, dc, sc, vC, hC] = GrabCut(im, initMap)
2 %
3 % Performs segmentation of image into foreground/background
4 % regions
5 %
6 % Usage:
7 %   L = GrabCut(im, initMap)
8 %
9 % Inputs:
10 %   im - color image to be segmented
11 %   initMap - initial labeling of which we are _certain_ of:
12 %           1 - FG
13 %          -1 - BG
14 %           0 - uncertain
15 %
16 % Output:
17 %   L - a segmentation (i.e., {0,1} labeling) of the give image.
18 %
19 % This implementation follows the paper
20 % Rother C., Kolmogorov V. and Blake A. "GrabCut"-Interactive
21 % Foreground Extraction using iterated Graph Cuts. SIGGRAPH
22 %   2004.

```



```

23
24 % constants:
25 Gamma = 50;
26 MaxItr = 100;
27 K = 6; % number of components in each mixture model
28 INF = 1000;
29
30
31 % working in Lab space
32 labim = im; % RGB2Lab(im);
33
34 [hC vC] = SmoothnessTerm(labim);
35 sc = Gamma.*[0 1; 1 0];
36
37 certain_fg = initMap == 1;
38 certain_bg = initMap == -1;
39
40 oL = initMap;
41
42 % how to label all the uncertain pixels?
43 if (sum(sum(abs(certain_fg))) == 0)
44     oL(initMap==0) = 1;
45 else
46     oL(initMap==0) = -1;
47 end
48
49 done_suc = false;
50
51 for itr=1:MaxItr,
52

```

```

53     %GMM for foreground and background – global model
54     tic ;
55     logpFG = LocalColorModel_medoids(labim , K, oL==1);
56     logpBG = LocalColorModel_medoids(labim , K, oL==–1);
57     toc ;
58
59     % force labeling of certain labeling
60     logpBG(certain_fg) = INF;
61     logpFG(certain_bg) = INF;
62
63     dc = cat(3, logpBG, logpFG);
64
65     gch = GraphCut('open', dc , sc , vC, hC);
66     gch = GraphCut('set', gch, int32(oL==1)); % initial guess –
67     previous result
68     [gch L] = GraphCut('expand', gch);
69     L = (2*L–1); % convert {0,1} to {–1,1} labeling
70
71     % stop if converged
72     if sum(oL(:)~=L(:)) < .001*numel(L)
73         done_suc = true;
74         break;
75     end
76
77     oL = L;
78 end
79
80 if ~done_suc
81     warning('GrabCut:GrabCut', 'Failed to converge after %d

```

```

    iterations', itr);
82 end

```

B.5 Local colour model generated by GMM-Kmedoids

```

1 function logp = LocalColorModel_medoids(labim, K, select)
2 % estimate global color model
3 % labim – image patch in Lab space
4 % K – number of component in mixture model
5 % select – what pixels in patch belongs to region
6
7 imsiz = size(labim);
8
9 imR = labim(:, :, 1);
10 imG = labim(:, :, 2);
11 imB = labim(:, :, 3);
12
13 fullSize = prod(imsiz(1:2));
14
15 vectIm = [reshape(imR, [1 fullSize]); reshape(imG, [1 fullSize])
    ; reshape(imB, [1 fullSize])];
16
17 selected = find(select);
18 imR = imR(selected);
19 imG = imG(selected);
20 imB = imB(selected);
21 fg = [imR, imG, imB];
22
23
24 options= statset('Display', 'iter');

```

```

25 rng('default'); %For reproducibility
26 tic;
27 [idx,C,sumd,D_d] = kmedoids(fg, K, 'distance', 'cityblock', '
    Options', options);
28 toc;
29
30 disp('Intra cluster mean distance');
31 disp(sum(sumd));
32 disp('Inter cluster mean distance');
33 disp(sum(D_d(:)));
34
35 x= fg(:,1);
36 y=fg(:,2);
37
38 figure;
39 cla;
40 gca;
41 hold on
42 grid on
43 colors='rgbcym';
44
45 for num=1:K
46 plot(x(idx==num),y(idx==num),[colors(num) '.']);
47 end
48 plot(C(:,1),C(:,2), 'kx', ...
49     'MarkerSize',15, 'LineWidth',3);
50
51 legend('Cluster 1', 'Cluster 2', 'Medoids', ...
52     'Location', 'NW')
53 title 'Cluster Assignments and Medoids for kmedoids';

```

```

54 hold off
55     tic ;
56     s=silhouette(fg , idx , 'sqeuclidean ');
57     mean_s= mean(s);
58     toc ;
59
60 disp('mean');
61 disp(mean_s);
62
63 % C – medoid
64 % cvr – covariance
65 % wi – component weight
66
67 % compute data cost according to model
68 logp = zeros(prod(imsiz(1:2)),K);
69 for ki=1:K
70     cvr = cov(fg(idx==ki, :));
71     if rank(cvr) < imsiz(3)
72         logp(:, ki) = inf;
73         continue;
74     end
75     wi = sum(idx==ki) ./ size(fg,1);
76
77     df = vectIm';
78     df(:, 1) = df(:, 1) - C(ki,1);
79     df(:, 2) = df(:, 2) - C(ki,2);
80     df(:, 3) = df(:, 3) - C(ki,3);
81
82     logp(:, ki) = -log(wi)+.5*log(det(cvr)) + ...
83         .5*sum(df*inv(cvr).*df, 2);

```

```
84 end
85
86 logp = min(logp, [], 2);
87 logp = reshape(logp, imsiz(1:2));
```



University of Fort Hare
Together in Excellence