University of Nottingham
UK | CHINA | MALAYSIA

Palm Tree Detection in UAV Images: A Hybrid Approach Based on Multimodal Particle Swarm Optimisation

Chen Zi Yan

Supervised by:

Dr. Iman Yi Liao (Main supervisor)
Dr. Amr Ahmed (Co-supervisor)

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy

April 2021

Abstract

In recent years, there has been a surge of interest in palm tree detection using unmanned aerial vehicle (UAV) images, with implications for sustainability, productivity, and profitability. Similar to other object detection problems in the field of computer vision, palm tree detection typically involves classifying palm trees from non-palm tree objects or background and localising every palm tree instance in an image. Palm tree detection in large-scale high-resolution UAV images is challenging due to the large number of pixels that need to be visited by the object detector, which is computationally costly. In this thesis, we design a novel hybrid approach based on multimodal particle swarm optimisation (MPSO) algorithm that can speed up the localisation process whilst maintaining optimal accuracy for palm tree detection in UAV images. The proposed method uses a feature-extraction-based classifier as the MPSO's objective function to seek multiple positions and scales in an image that maximise the detection score. The feature-extraction-based classifier was carefully selected through empirical study and was proven seven times faster than the state-of-the-art convolutional neural network (CNN) with comparable accuracy. The research goes on with the development of a new k-d tree-structured MPSO algorithm, which is called KDT-SPSO that significantly speeds up MPSO's nearest neighbour search by only exploring the subspaces that most likely contain the query point's neighbours. KDT-SPSO was demonstrated effective in solving multimodal benchmark functions and outperformed other competitors when applied on UAV images. Finally, we devise a new approach that utilises a 3D digital surface model (DSM) to generate high confidence proposals for KDT-SPSO and existing region-based CNN (R-CNN) for palm tree detection. The use of DSM as prior information about the number and location of palm trees reduces the search space within images and decreases overall computation time. Our hybrid approach can be executed in non-specialised hardware without long training hours, achieving similar accuracy as the state-of-the-art R-CNN.

List of Publications

The research presented in this thesis has been published in the following conference and journals:

Conference

1. Chen, Z. Y. and Liao, I. Y. (2019). Evaluation of feature extraction methods for classification of palm trees in uav images. In *2019 International Conference on Computer and Drone Applications (IConDA)*, pages 13-18. IEEE. **(Best Paper)**

Journals

1. Chen, Z. Y. and Liao, I. Y. (2020). Improved fast r-cnn with fusion of optical and 3d data for robust palm tree detection in high resolution uav images. *International Journal of Machine Learning and Computing*, 10(1):122-127.

2. Chen, Z. Y., Liao, I. Y., and Ahmed, A. (2021). Kdt-spso: A multimodal particle swarm optimisation algorithm based on k-d trees for palm tree detection. *Applied Soft Computing*, 103:107156.

Acknowledgements

My heartfelt gratitude goes to my main supervisor, Dr. Iman Yi Liao, for her kind advices, suggestions and support during my PhD journey. I am proud to be under her guidance. I would also like to thank my co-supervisor, Dr. Amr Ahmed, for his insightful comments and suggestions. Many thanks go to Dr. Tomas Henrique Maul and my friends in the School of Computer Science for their motivations and feedback on my research.

I would like to express my sincere appreciation and thanks to my wife, Ru Chien, for her patience, support, love, and understanding that enabled me to complete this work. It has been a long journey, we got through it together, which was spiced up with many extraordinary and exciting moments. First, we got married at the beginning of the study. Then, our first baby, Chen Xin, was born prior to my final year review. Next, I started writing up this thesis during the COVID-19 pandemic, and finally, our second baby, Chen Tsen, was born just two weeks before my viva.

I am especially indebted to Mr. Goh Kah Joo and Mr. Tey Seng Heng, the former and current Director of Research of AAR, respectively, for allowing me the opportunity and time to complete this study. Appreciation to Mr. Cheah Li Wen and Dr. Goh You Keng for proofreading and editing my journal articles before publishing. I would also like to thank my AAR colleague, Yit Kheng, for providing me coffee and cookies while I was carrying out the research at the office. Finally, a special thank to Advanced Agriecological Research Sdn. Bhd., Boustead Plantations Berhad, and Kuala Lumpur Kepong Berhad for the financial sponsorship throughout this study.

Contents

Abstract **i**

List of Publications **iii**

Acknowledgements **iv**

List of Tables **x**

List of Figures **xi**

List of Abbreviations **xvi**

Chapter 1 Introduction **1**

1.1 Background 1

1.2 Main challenges in palm tree detection 3

1.3 Objectives of the research 6

1.4 Overview of the methodology 7

1.5 Contributions 8

1.6 Thesis Structure 10

Chapter 2 Overview of Palm Tree Detection **13**

2.1 Bottom-up approach 13

2.2 Top-down approach 15

2.3 Region-based CNN Approach 21

Chapter 3 Evaluation of Feature Extraction Methods for Classification of Palm Trees **24**

3.1 Introduction 24

3.2 Methods 25

3.3 Results and Discussion 39

3.4 Dimensionality reduction of LBP features 44

3.5 Summary 47

Chapter 4 Improved Multimodal Particle Swarm Optimisation Algorithm 50

4.1 Introduction 50

4.2 Background 51

4.3 Optimising SPSO using k-d tree structure 55

4.4 Benchmark functions and experiment settings 64

4.5 Summary 71

Chapter 5 Application of KDT-SPSO for Palm Tree Detection 73

5.1 Introduction 73

5.2 Methods 74

5.3 Optimising KDT-SPSO's parameters for palm tree detection 79

5.4 Evaluation of KDT-SPSO's performance on test images . . . 81

5.5 Results and discussion 86

5.6 Summary 96

Chapter 6 Integration of Digital Surface Model (DSM) into Palm Tree Detection Framework 97

6.1 Introduction 97

6.2 Background 98

6.3 Integration of Digital Surface Model (DSM) into KDT-SPSO 100

6.4 Integration of DSM into region-based CNN (R-CNN) 108

6.5 Summary 118

Chapter 7 Summary and Future Work 120

7.1 Evaluation of feature extraction methods for classification of palm trees 120

7.2 Improved multimodal particle swarm optimisation algorithm 122

7.3 Application of KDT-SPSO for palm tree detection 123

7.4 Integration of digital surface model (DSM) into palm tree detection framework 124

Bibliography **126**

List of Tables

3.1 Details of the AlexNet architecture 30
3.2 Details of the feature extractors and classifiers tested in the study 32
3.3 Sources of the library used for feature extraction 37
3.4 The quantitative accuracy assessment results of different palm tree extraction approaches 40
3.5 Comparison of accuracy and computation runtime of LBP features before and after dimensionality reduction. 47

4.1 Test functions. 65
4.2 Parameters and criteria for test functions. 66
4.3 Comparison of average number of evaluations (ANE). . . . 68
4.4 Comparison of computation time (CT) in seconds. 69
4.5 Comparison of success rate (SR). 71
4.6 Comparison of peak ratio (PR). 71

5.1 Experimental results with respect to the performance of KDT-SPSO with different levels of POP_{size} and POP_{max} on the test image based on 50 independent runs. 81
5.2 Description of the methods tested in our palm tree detection experiment. 86
5.3 Comparison of recall rate obtained by different methods on UAV images (%). 87
5.4 Comparison of precision rate obtained by different methods on UAV images (%). 89

5.5 Comparison of F1-Score obtained by different methods on UAV images (%). 90
5.6 Comparison of runtime (s) 92

6.1 Comparison of performance between KDT-SPSO and KDT-SPSO+D on the first 3 set of large-scale images 105
6.2 Comparison of performance between KDT-SPSO and KDT-SPSO+D on additional 25 images of 500 × 500 pixels. . . . 107
6.3 Comparative results between Faster R-CNN, our approach, and Yolo in respective to recall ,precision, F1-score, and detection time in three study areas 114
6.4 Comparative results between Faster R-CNN and our approach based on AlexNet backbone. 115

List of Figures

1.1 This figure shows the overview of our methodology in designing an efficient method for palm tree detection. The yellow boxes indicate the components that are integrated into our hybrid approach. The k-d tree structure is implemented to improve the overall performance of the multimodal PSO algorithm. 9

2.1 The SIFT key points extracted from a mature palm area where the canopies are heavily overlapped following the method proposed by Malek et al. (2014). The grouping of points with similar properties into different clusters representing individual palms is difficult using the bottom-up approach. . 15

3.1 Transformation of pixel values into gradient information using HOG feature extractor 26

3.2 Transformation of pixel values into binary digit using LBP feature extractor 27

3.3 The sample of 128 feature descriptors computed by SIFT operator in the region of one point 28

3.4 Architecture of AlexNet used in this study 31

3.5 Examples of the Haar-like features used in the experiment . 35

3.6 Training accuracy of the proposed AlexNet model 36

3.7 Training samples used in the study. 37

3.8 Comparison of performance between feature extractors, averaging over different classifiers 42

3.9 Comparison of performance between classifiers, averaging over different feature extractors 43

3.10 Comparison of computation time between feature extractors, averaging over different classifiers 44

3.11 Comparison of computation time between classifiers, averaging over different feature extractors 45

3.12 Cross-validation performance against the number of LBP features used for classification 46

4.1 The chart on top (a) and the tree graph at the bottom (b) show how a two-dimensional space is structured using the k-d tree. The blue dots in the chart represent data points and also correspond to the tree nodes subdividing the space into two regions. The black vertical lines correspond to the splits in the x-axis while the red horizontal lines correspond to the splits in the y-axis. The space is split along a different dimension at each level of the tree. The points with lower values than the split node in the corresponding axis are partitioned into the left/bottom subspace, while the points with higher values will be partitioned into the right/upper subspace. The deeper the level of the splits, the thinner the line. The splitting procedure is repeated until each data point is transformed into a node. 60

4.2 The chart on top (a) and the tree graph at the bottom(b) show how nearest neighbour search is performed using the k-d tree. The blue dots in the chart represent data points, which also correspond to tree nodes subdividing the space into two regions. The black vertical lines correspond to splits in the x-axis while the red horizontal lines correspond to splits in the y-axis. To find the neighbours of P7 (red dot), the algorithm only needs to evaluate P16, P19, P7, P6, P8, and P9 (yellow dots). The yellow nodes in the tree graph indicate visited nodes while the grey nodes indicate eliminated nodes that are not visited. 61

4.3 Comparison of computation time of three different algorithms. 70

5.1 Pictorial description of the KDT-SPSO algorithm for palm tree detection. 77

5.2 Flowchart of the proposed KDT-SPSO algorithm for palm tree detection. 78

5.3 Effect of POP_{size} and POP_{max} on SR and ANE. 82

5.4 F1-score results against different levels of NMS's overlap ratio. 83

5.5 F1-score results against different levels of QUBO's overlap ratio. 85

5.6 Test images showing results of KDT-SPSO detection. The blue circles indicate detected palm trees. The red circles indicate false positives while the yellow circles indicate false negatives. The images are best viewed in colour. 94

5.7 Snapshots of the KDT-SPSO's searching process at 1st, 5th and 20th iteration, respectively in Image 1 (a-c), Image 2(d-f), Image 3 (g-i), and Image 4 (j-l). The red asterisks indicate particles, the blue asterisks indicate species seeds while the blue circles indicate detected canopy sizes. The canopy diameter corresponds to the width of the image patch. The red circles in Image 4 indicate false positives. The images are best viewed in colour. 95

6.1 3D representation of palm trees 98
6.2 The deliverables of the SfM photogrammetric process from sequences of overlapping UAV images. The blue colour in the resultant DSM represents a lower elevation value 100
6.3 The 4-connected regional maxima operator that is used to identify potential palm tree proposals in the experiment . . . 101
6.4 The three large-scale UAV images used in testing the effect of integrating DSM into palm tree detection pipeline. 102
6.5 The 25 additional small-scale UAV images of 500 × 500 pixels each used in testing the effect of integrating DSM into palm tree detection pipeline 104
6.6 Test images showing results of KDT-SPSO (left column) and KDT-SPSO+DSM (right column) detections. The blue circles indicate detected palm trees. The red circles indicate false positives while the yellow circles indicate false negatives. The images are best viewed in colour. 106
6.7 Comparison of the initial proposals (red asterisk) generated by KDT-SPSO(a) and KDT-SPSO+D(b). 106
6.8 Architecture of the proposed method used in this experiment.112

6.9 Test images showing results of Faster R-CNN (first column), our approach (second column) and YOLO (third column). The blue circles indicate detected palm trees. The red circles indicate false positives while the yellow circles indicate false negatives. The images are best viewed in colour. 114

6.10 Comparison of palm detection F1 score with respect to different image dimensions. 116

6.11 Comparison of palm detection speed with respect to different image dimensions. 117

List of Abbreviations

ACO	Ant colony optimisation
ALS	Airborne laser scanning
ANE	Average number of evaluations
CHM	Canopy height model
CNN	Convolutional neural network
Conv	Convolutional layers
CT	Computation time
DE	Differential evolution
DOG	Difference of Gaussians
DSM	Digital surface model
DTM	Digital terrain model
ELM	Extreme learning machine
FC	Fully-connected layers
FN	False negative
FP	False positive
GA	Genetic algorithm
HOG	Histogram of oriented gradients
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge
IoU	Intersection over Union
KDT-SPSO	Improved SPSO based on k-d trees
KDT-SPSO+D	Improved KDT-SPSO with DSM
LBP	Local binary patterns
LBP(SVMRBF)	LBP features trained with SVMRBF
LBP(SVMRBF)-R	Selected LBP features trained with SVMRBF
LM	Local maximum
MMO	Multimodal optimisation
MNE	Maximum number of evaluation
NIR	Near-infrared
NMS	Non-maximal suppression

NNS	Nearest neighbour search
PR	Peak ratio
PSO	Particle swarm optimisation
QUBO	Quadratic unconstrained binary optimisation
r3PSO	PSO based on ring topology
RBF	Radial basis function
R-CNN	Region-based convolutional neural network
RF	Random forest
RGB	Red, green, Blue
RoIs	Region of interests
RPN	Region proposal network
SA	Simulated annealing
SfM	Structure-from-motion
SGD	Stochastic gradient descent
SIFT	Scale-invariant feature transform
S-NMS	Sampling approch and final selection using NMS
SPSO	Species-based particle swarm optimisation
S-QUBO	Sampling approach and final selection using QUBO
SR	Success rate
SS	Selective search
SSD	Single Shot Multibox Detector
SVM	Support vector machine
SVML	Support vector machine with linear kernel
SVMRBF	Support vector machine with radial basis function kernel
SW	Sliding window approach
TP	True positive
UAV	Unmanned aerial vehicle
VJ	Viola-Jones algorithm
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Background

Oil palm (*Elaeis guineensis*) is widely grown in the tropics especially in Malaysia and Indonesia due to its economic importance as the highest yielding oil crop and most consumed vegetable oil in the world (USDA-FAS, 2020). Accurate palm density records are crucial for estate managers to project yields and estimate the number of resources required, particularly fertiliser cost estimation, which accounts for approximately 24% of the total production cost (Goh et al., 1999). It is common practice to obtain palm tree records by manual inspection on the ground, which is laborious, costly, and error-prone. The results obtained from the manual counting is also difficult to be verified.

Due to the technological advancements, the use of remote sensing images provides a practical means for large-scale plantation management and vegetation monitoring. Palm counting is one of the estate operations that benefits from the use of remote sensing technologies. The detection of palm trees and other features on the ground can be automated through the application of image processing and machine learning techniques on the imageries. The need for physical counting on the ground can be partially or completely eliminated while the speed of counting can also be accelerated with the help of remote sensing images. Space-borne remote sensing data captured from IKONOS, Landsat, SPOT, and RapidEye are commonly used in the agricultural sector as they have high discrimination power to differentiate various objects on the ground due to the multispectral sensors mounted on the satellites. However, the images taken with satellite-borne sensors are often interrupted by clouds, which is the major problem encountered in the tropics. The usage of small and low-cost unmanned aerial vehicles (UAVs) has received wide attention as an alternative to satellite-borne sensors to monitor large scale oil palm plantations (Rokhmana, 2015). Hence, this research focuses on palm tree detection in UAV images.

The task of palm tree detection in UAV images has been performed similarly to other object detection problems in the field of computer vision, such as face detection, human detection, or vehicle counting, by scanning a multi-scale rectangular detector across the input image to exhaustively search for the locations and scales that likely contain instances of specific object. Though it is technically simple, its application in large-scale UAV images for palm tree detection remains the biggest challenge. It is computationally expensive to evaluate a large number of image pixels without prior knowledge of the size and location of objects. For example, a UAV image that covers a medium-sized plantation of about 1,000 ha can be made up of 100,000,000$\times$100,000,000 pixels, which is much larger than the size of a high-resolution photo captured using an ordinary camera.

1.2 Main challenges in palm tree detection

Various heuristics have been devoted to improving the efficiency of the exhaustive searching process, i.e., reducing the computation required for each extracted region and/or the number of regions Nguyen et al. (2016). The first method aims at filtering out non-object regions as early as possible. For example, Li et al. (2019) proposed multiscale CNN models where the first CNN model was used to extract potential palm tree regions at coarse resolution. The second CNN model was used to search for individual palm trees at finer resolution. Despite the filtering process, its efficiency degrades significantly for palm tree detection, where the area of interest usually occupies a large image ratio. On the other hand, the second method aims to reduce the number of function evaluations required in an image, which is performed by searching an image over a coarse grid or oversampling many proposals on the image, and selecting the best subset of them. As palm trees are densely planted in plantation, this will inevitably increase the risk of completely missing the objects. Given the limitations of exhaustive searching and subsampling strategies, the multimodal optimisation (MMO) approach that utilises bioinspired optimisation algorithms seems to be the best solution to locate objects in an image with speeded up localisation process without compromising the accuracy. Despite the success achieved by MMO algorithms with species-based particle swarm optimisation (SPSO) (Parrott and Xiaodong Li, 2004) as one of the most outstanding and influential methods, they were still inefficient due to the repetitive Euclidean distance calculations needed in their integrated nearest neighbour search.

Recently, state-of-the-art object detection methods that predict bounding boxes' locations using region proposal network (RPN) have also widely been applied, such as Faster R-CNN (Ren et al., 2015), YOLO (Redmon et al., 2015), SSD (Liu et al., 2015), and their variants. These region-based convolutional neural networks (R-CNN) take an input image, scale it down to a predefined size, and divide it into multiple non overlapping regions. Then, they classify the presence of an object in each region and regress the location and size of the object's bounding box in the region simultaneously without the need for a greedy search. However, it is well known that the training phase of the deep learning model is the most resource-intensive task, which requires high performance computing infrastructure to efficiently learn models (Zhang et al., 2019; Chen et al., 2019). Specialised hardware with high computing power may not be feasible for the small plantation holders on a restricted budget. For example, a high-performance computer equipped with Intel i9 CPU, Geforce RTX 24GB GPU, 2TB SATA HDD, and 64GB memory will cost about RM 17,000. Although R-CNNs can achieve high detection accuracy, extraordinarily large amounts of labelled images are needed, with each manually annotated by humans, which requires a significant amount of time, energy and money. Since each sub-region of a plantation has its environmental features, a site-specific model has to be developed for different sub-regions, which involves repetitive works. Otherwise, the model performance will deteriorate if the same model is applied across different areas (Zheng et al., 2021). Furthermore, directly applying the R-CNN method to detect small objects like palm trees in remote sensing images usually renders unsatisfactory performance mainly due to the coarseness of its feature maps (Kong et al., 2016; Ren et al., 2018).

Accurate and timely palm tree records are important for estate managers to make a decision particularly on fertiliser procurement. Under normal circumstances, the managers are given a year's time to update the palm records of their estates before fiscal year-end. However, in extreme cases, they are only given a few weeks' time to confirm the updated palm records to secure a fertiliser contract with the best price (personal communication, 2021). As such, there is a need to improve the efficiency and accuracy of the existing object detection algorithms to cater to the palm tree detection task in large-scale high-resolution UAV images. The computation cost scales up exponentially with image size. More specifically, the challenges of palm tree detection in UAV images can be characterised by the following:

1. The existing feature extractors were not designed specifically for palm tree detection. This may cause potential performance deterioration if applied to the data that was not encompassed in the initial design. Also, the more complex extractors will usually yield higher accuracy, but take more computation time than the less complex ones.

2. Objection localisation in the object detection pipeline is the most time consuming process as the detector has to scan through the entire image to locate all objects and at multiple scales. The typical exhaustive sliding window approach requires high computation cost, while the subsampling method will result in a high missing rate. On the other hand, the multimodal optimisation approach, which is the middle way of both approaches, is inefficient due to its exhaustive nearest neighbour search mechanism.

3. The parameters defined in bioinspired algorithms are problem dependent, the same set of parameters may not be suitable for another set of problems. Using the parameters blindly for palm tree detection may result in low accuracy. In addition, the efficiency of bioinspired algorithms is largely affected by the number of population/possible solutions that need to be evaluated. Finding the optimum population for specific problems is a non-trivial task as too many of them will increase computer burden, too few of them will increase the missing rate.

4. The modern CNN models achieve state-of-the-art performance at the expense of extensive collection of labelled data and high-performance computers. These data need to be prepared and checked by one or more experts, which is costly and time-consuming. The high-performance computer is normally not available in plantations, especially for smallholders.

1.3 Objectives of the research

The general objective of this thesis is to design an approach that can speed up the localisation process whilst maintaining optimal accuracy for palm tree detection in UAV images. The following specific objectives were set to address the challenges articulated in Section 1.2:

1. To select the best performing feature extraction and classification methods in terms of speed and accuracy for palm tree classification.
2. To improve the efficiency of the existing multimodal species-based PSO (SPSO) algorithm's nearest neighbour search mechanism.
3. To optimise the model parameters of the improved SPSO for palm tree detection.
4. To enhance the computation speed and detection accuracy of the developed method.
5. To improve the performance of the existing R-CNN models for palm tree detection.

1.4 Overview of the methodology

In order to accomplish the research objectives stated in Section 1.3, the methodology illustrated in Figure 1.1 was implemented. Since our research aims to design an efficient method that can be executed in non-specialised hardware for palm tree detection, we mainly focused on the top-down classical approach that involves model construction and localisation stages. The details of the approaches are further discussed in Chapter 2. In the object detection framework, the performance of object classifier is crucial to ensure that all palm tree instances are correctly classified in the input image. First of all, we evaluated the performance of classical human-engineered feature extractors, namely HOG, LBP, and SIFT, requiring lower computing cost against the state-of-the-art CNNs approaches requiring higher computing cost. The best performing feature extractor and classifier combination were further enhanced by means of the feature selection approach. The optimised feature-extraction-based classifier is served as the objective function for the multimodal PSO algorithm.

Next, we delved into the localisation stage that targets detecting all palm tree instances in an input image. The classical greedy search approach is computationally expensive, while the subsampling approach may result in high missing and error rates. We resolved in multimodal optimisation approach, which is the middle way of both that maintains speed and accuracy. We leveraged the multimodal species-based PSO (SPSO) algorithm as the main framework for our study. However, in the original SPSO algorithm, and other multimodal optimisation algorithms, distance calculations must be performed for all particle pairs at every iteration to find the query points' nearest neighbours, which was typically carried out by an exhaustive naive search technique. We introduced a special tree-based structure, called the k-d tree, to speed up the nearest neighbour search. The improved

SPSO, KDT-SPSO, was tested on several benchmark functions to prove its generality in solving multimodal optimisation problems.

Then, we applied KDT-SPSO in actual palm tree detection. Prior to this, we optimised KDT-SPSO's parameter settings for the palm tree detection task through a set of factorial experiments. We introduced a restart mechanism to speed up its convergence and improve the recall rate. The best performing feature-extraction-based classifier was selected for the KDT-SPSO's objective function to seek the best set of object proposals that maximised the classification score. KDT-SPSO's performance was evaluated and compared with the greedy search and subsampling approaches.

Lastly, we extracted a set of high confidence object proposals from digital surface model (DSM) using the regional maxima image processing operator and passed the extracted proposals to the KDT-SPSO framework for optimisation. This strategy further enhanced the performance of KDT-SPSO by reducing the computation cost and false detection rate. We also used the same methodology to extract and incorporate the high confidence object proposals into the Fast R-CNN framework to test the applicability of DSM in other object detection frameworks. We compared the performance of all approaches.

1.5 Contributions

The major contributions resulting from the research can be summarised as follows:

1. We established the best combination of feature extractor and classifier for palm tree classification through empirical study and feature selection approach. The comparison in this study also provided us with

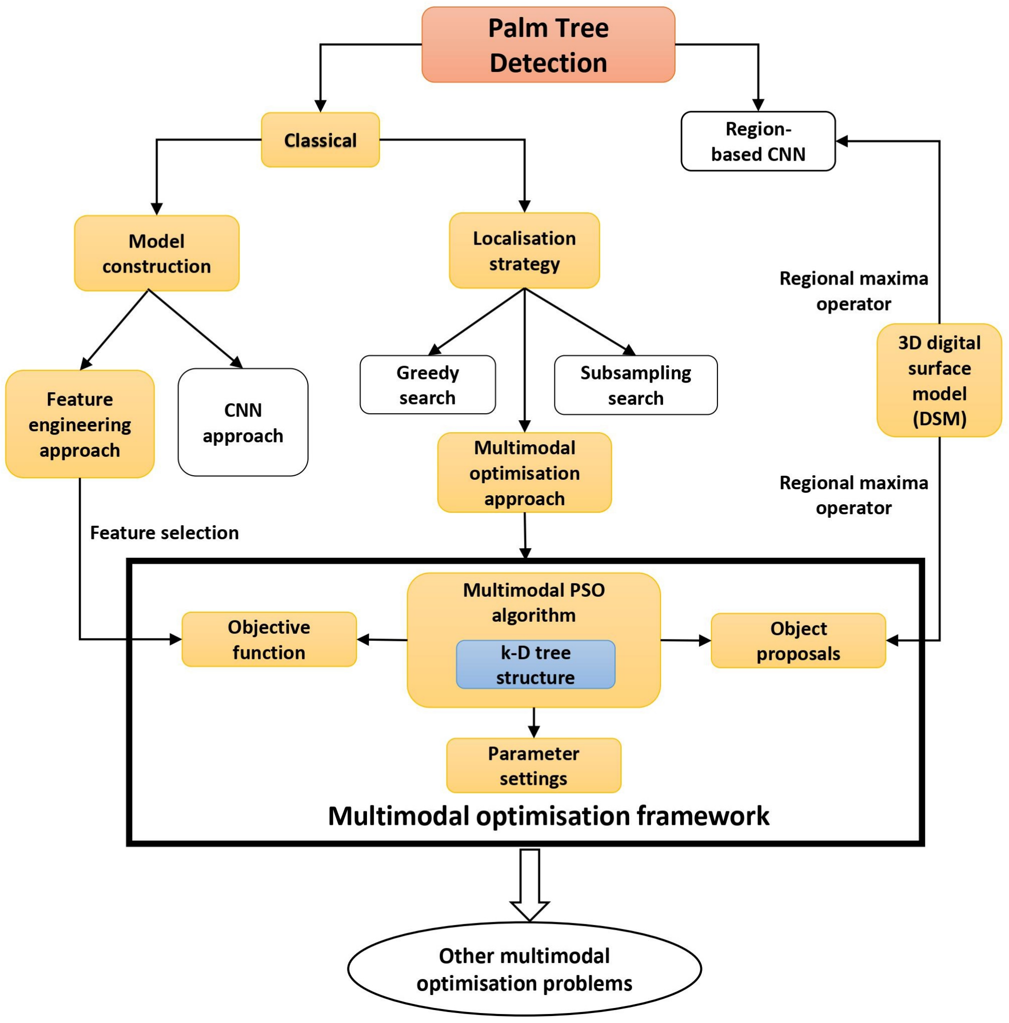


Figure 1.1: This figure shows the overview of our methodology in designing an efficient method for palm tree detection. The yellow boxes indicate the components that are integrated into our hybrid approach. The k-d tree structure is implemented to improve the overall performance of the multimodal PSO algorithm.

a better understanding of the advantages and limitations of each approach (Chapter 3). The main findings have been published in **Chen and Liao (2019)**.

2. A novel algorithm that incorporated k-d tree framework into multimodal PSO was proposed to reduce the nearest neighbour search complexity. The framework can be integrated into other multimodal algorithms requiring distance evaluations (Chapter 4). The main findings have been published in **Chen et al. (2021)**.

3. For the first time, we formulated the palm tree detection problem as multimodal optimisation problem and applied the improved multimodal PSO (KDT-SPSO) to solve the palm tree detection problem. Compared to Al-Ruzouq et al. (2018), who applied ant colony optimisation (ACO) algorithm to select the most significant features from aerial images to detect palm trees through an image segmentation approach, our study applied the PSO algorithm to solve the palm tree localisation problem directly without involving the image segmentation process. We also introduced a modified restart mechanism into KDT-SPSO to avoid premature convergence. The optimum parameter settings of KDT-SPSO obtained through our empirical study can be reused in new images. Our approach is also applicable to other types of object detection (Chapter 5). The main findings have been published in **Chen et al. (2021)**.

4. We introduced a novel approach that incorporated high confidence object proposals generated from the 3D digital surface model (DSM) into KDT-SPSO's initialisation process to reduce computation cost. We demonstrated that our approach could achieve state-of-the-art accuracy without the need for specialised hardware, as is the case in region-based CNNs (Chapter 6).

5. We also presented the fusion of DSM into Fast R-CNN region-based CNN, for the first time, for palm tree detection in Chapter 6. The main findings have been published in **Chen and Liao (2020)**.

1.6 Thesis Structure

The thesis is structured as follows to address the problems:

1.6.1 Chapter 2

This chapter provides, in general, the approaches used in object detection focusing on palm tree detection. It briefly describes the advantages and limitations of each approach, which provide the interest area of this research.

1.6.2 Chapter 3

This chapter comprehensively investigates feature extraction methods and classifiers for palm tree classification. This is a vital process as the performance of the selected classifier affects the accuracy of palm localisation in the later stage.

1.6.3 Chapter 4

This chapter introduces a special binary search tree, called the k-D tree, into the PSO algorithm that significantly improves the efficiency of the nearest neighbour search. The improved algorithm, called KDT-SPSO, is evaluated on several benchmark functions against other competitors to prove its applicability in solving multimodal optimisation problems.

1.6.4 Chapter 5

This chapter presents the application of KDT-SPSO in palm tree detection. The parameters of KDT-SPSO is fine-tuned through empirical study to find the optimal parameter settings for palm tree detection in UAV images. The optimised KDT-SPSO is statistically tested on 25 test images, and its performance is compared with the greedy and subsampling approaches.

1.6.5 Chapter 6

This chapter explains the integration of DSM in KDT-SPSO to generate high confidence proposals to speed up convergence speed and improve accuracy. It also presents how DSM is fused into the Fast R-CNN model. The performance of all approaches is compared and detailed.

1.6.6 Chapter 7

This chapter briefly discusses the general summary and conclusions drawn from each chapter, and gives directions for future work.

Chapter 2

Overview of Palm Tree Detection

This chapter aims to provide an overview of image processing, computer vision, machine learning, and CNN techniques that are commonly used in palm tree detection. As with any other problem of object detection in an image, palm tree detection involves two key tasks: (1) to detect the presence of palm tree in an image and (2) to locate accurately all instances of palm tree in the image regardless of scale, pose, view points, occlusions and illumination changes, which can be performed by either bottom-up or top-down object detection strategy.

2.1 Bottom-up approach

Traditionally, palm tree detection has been performed using bottom-up approaches, which involve image segmentation in the first step by separating foreground and background pixels. The foreground pixels are annotated by the object label, and they are grouped into segments with homogenous properties such as colour, texture, and intensity to form object proposals representing object of interest, e.g. trees, humans, buildings, vehicles, etc. We call this strategy a bottom-up approach because object detection starts from the low-level features (pixels) to higher-level features (regions). Image processing techniques such as region-growing, watershed transform, clustering algorithm, active contour method, morphological operators, etc., are often applied to split or merge the image pixels during the palm tree detection process (Bazi et al., 2014; Daliman et al., 2014; Chemura et al., 2015; Shafri et al., 2011; Mansoori et al., 2018; Santoso et al., 2016). The main advantage of bottom-up approaches is that the shape or boundary of the object can be demarcated in a more representative way. The process of classifying image pixels into different categories/labels and clustering them to represent objects remains one of the most common research questions in image processing (Chouhan et al., 2019). Since there is no universal segmentation solution for all kinds of images and every image can be segmented intuitively using different segmentation techniques, domain specific knowledge must be used to tackle each problem effectively (Zaitoun and Aqel, 2015). Also, it is challenging to delineate objects with arbitrary shapes, and extract objects in a cluttered and noisy background, such as trees and bushes (Hossain and Chen, 2019). The effect of shadows, the fuzziness of tree canopy, an overlapping crown will affect the segmentation accuracy. Another bottleneck is the lack of discrimination power to distinguish palm trees and background by using UAV images. UAV images usually consist of only red, green, blue (RGB), and/or near-infrared bands (NIR), which are far less than the multi-spectral or hyper-spectral bands offered by satellite images. This approach is most suitable in areas where the colour of the object of interest is significantly different from the background, otherwise, it will be challenging to group the pixels into separate regions to represent

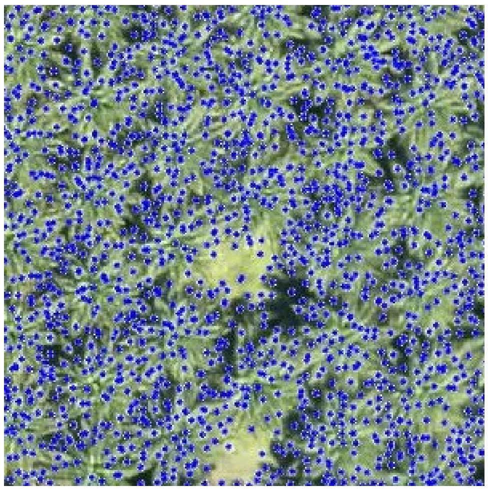


Figure 2.1: The SIFT key points extracted from a mature palm area where the canopies are heavily overlapped following the method proposed by Malek et al. (2014). The grouping of points with similar properties into different clusters representing individual palms is difficult using the bottom-up approach.

the object as shown in Figure 2.1.

2.2 Top-down approach

Due to the advent of machine learning and deep learning techniques, palm tree detection have shifted from bottom-up towards top-down approaches. Top-down approaches typically comprise of model construction stage and localisation stage. In the model construction stage, a set of salient features are extracted from training images of different object classes (e.g. palm trees and non-palm trees) using feature extraction techniques. These features define the raw images using small representative numerical data. The extracted features are passed to a machine learning algorithm to build a model that can classify different image categories. The localisation stage is basically where the trained model is applied on a new input image to find the instance(s) of a known object in the image that best matches the model. Instead of using the tedious image processing techniques to group image pixels to form object proposals, the top-down approach assumes that an object is bounded in a rectangular box, and the box with a high confidence score likely contains the object of interest.

2.2.1 Model construction stage

The features used in the learning stage can be broadly categorised into traditional human-engineered features, and convolutional neural network (CNN) extracted features. Human-engineered methods utilise colour histogram, gradient, intensity, and textures, such as histogram of gradient (HOG) (Dalal and Triggs, 2005), local binary patterns (LBP) (Ojala et al., 2002), scale-invariant feature transform (SIFT) (Lowe, 1999), and Haar-like rectangle features(Viola and Jones, 2001), for example, to represent the appearance of the object of interest and used on top of strong classifiers such as SVM, extreme learning machine (ELM), random forest (RF), AdaBoost, logistic regression, among others for learning. Earlier work on these was based on the template matching technique (Briechle and Hanebeck, 2001), aiming to match the region in the input image with the highest cross-correlation coefficient with the given image patch. Although template matching is easy to be implemented, it is very sensitive to scale, orientation, occlusion, and illumination that affect how an object appears in the image. The modern human-engineered techniques involve using feature extractors to extract salient information invariant to the changing image conditions for matching. For example, Malek et al. (2014) first extracted scale-invariant feature transform (SIFT) keypoints of date palms from UAV image and trained an extreme learning machine to classify these features into palm and non palm categories. In Manandhar et al. (2016), a shape feature called circular autocorrelation of the polar shape matrix was used to represent palm tree in UAV images and SVM was applied for classification. Similarly, histogram of gradients (HOG) (Wang et al., 2019), local binary patterns (LBP) (Rueda et al., 2016) and Haar-like features (Daliman et al., 2016) were used to extract shape and texture features from satellite or UAV images and achieved up to 100% detection accuracy, especially in young palm areas. The challenges in traditional methods are the need for good design for feature extractors and fine-tuning the model parameters during training. The results are also variable depending on the choice of classifiers. The feature extractor and classifier should be based on the problem domain as there is no single best combination that can solve the universal problem.

Different from traditional methods, CNNs can learn data representation through multiple processing layers without the user having to go through the complex feature engineering stage. The key difference of both approaches is the method used to extract features, i.e. the vector of parameters that represent input images. The features used in human-engineered approaches are created by experts while the features used in the CNN approach are derived from training data through CNN learning. One of the earliest CNN models was introduced by LeCun et al. (2015). The features extracted from CNN approaches can either be passed to SVM or SoftMax (Multinomial Logistic Regression) for classification. Although both classifiers perform similarly, most CNNs incorporate SoftMax into their models because SoftMax maps the networks' output into probability about the input data, which is easily understandable by the users (Qi et al., 2017). On the other hand, SVM maps the networks' output in the form of distance to the decision hyperplane, which has no physical meaning to the users. The use of CNN-based features for palm tree detection has been widely reported (Cheang et al., 2017; Mubin et al., 2019; Li et al., 2017). Although CNN-based feature extraction methods reduce the need for human intervention in the feature designing stage, CNNs require higher computer capacity to run complex models, a long training time, and a large amount of training data to achieve high accuracy.

2.2.2 Object localisation stage

In the object localisation stage, the trained feature-based classifier is applied to check whether the object of interest is present in the input image. The search strategy can be categorised into an exhaustive sliding window, subsampling, and multimodal optimisation approach.

Exhaustive sliding window approach

The sliding window is the most commonly used and typical approach to locating objects in an input image due to its simplicity. Objects can be located at any position and scale in the image. The sliding window detector is run at multiple scales and locations to generate sub-windows representing object proposals. Feature-based classifier is then applied to check the presence of the object in each of the sub-windows. The sliding window approach is computationally prohibitive because it will generate many object proposals, particularly in high-resolution UAV images. Determining the parameters of the windows is a non-trivial task as the size and stride of the moving window will affect the recognition accuracy. If the window size and stride are too large, it may increase the risk of missing out on objects, which sacrifices the accuracy to achieve acceptable speed. In contrast, if the moving size is too small, it may increase the risk of false detections and computation costs.

Subsampling approach

On the contrary, instead of searching for proposals systematically on an image, subsampling approaches select a subset of object proposals that are largely and randomly generated on the input image. Then, the simple yet efficient non-maximal suppression (NMS) is typically used to select high-quality proposals and discard those with low quality. Some studies proposed to use optimisation algorithms such as branch-and-bound (Lampert et al., 2008), quadratic unconstrained binary optimisation algorithm (QUBO) (Pham et al., 2016; Li and Ghosh, 2020), simulated annealing (Ge and Collins, 2009), among others, to select the best proposals by maximising the quality function whilst QUBO was claimed to be the best performing algorithm. However, the subsampling approach encounters similar issues as the sliding window approach. The initial sample size has to be large to encompass all objects with variable sizes present in the image. If the sample size is small, it will also have the risk of not detecting the object.

Multimodal optimisation approach

Suppose an image is described as a three-dimensional parameter space formed by all possible 2D coordinates and scales at which an object might be present. In that case, object localisation can be formulated as a multimodal optimisation (MMO) problem that seeks multiple positions and scales in an image that match the detector the best (maximum score). Unlike the subsampling approach that fixes the position and scale of the initial object proposals through the entire optimisation process, the MMO approach generates a set of much smaller samples and iteratively updates them through bioinspired algorithms. Despite the smaller sample size, the updating mechanism in the optimisation algorithm can reinitialise the existing samples to explore other regions in the image, which has overcome the issue of missing out objects as in the subsampling approach. The performance of MMO is between the other two approaches, where it could achieve higher accuracy than the subsampling approach while keeping lower computing complexity than the sliding window approach.

Traditional bioinspired algorithms are originally designed to solve unimodal problems that have only one global optimum. The algorithms tend to converge to a single solution if used directly to solve multimodal problems. Among those, genetic algorithm (GA) (Holland, 1992), simulated annealing (SA) (Kirkpatrick et al., 1983), particle swarm optimisation (PSO) (Kennedy and Eberhart, 1995), differential evolution (DE) (Storn and Price, 1997), and ant colony optimisation (ACO) (Dorigo et al., 2006) are the most successful and most cited optimisation approaches in current literatures (Cuevas et al., 2020). The general framework of most of the algorithms is similar, irrespective of the natural phenomenon from which the algorithm is inspired (Cuevas et al., 2020). Nonetheless, the comparative study results presented by Piotrowski et al. (2017) showed that the swarm-based PSO variants took much less time to find a solution compared to evolution-based algorithms, such as DE and GA. The objective function executed using PSO algorithms was also found to approach optimised solutions faster than SA algorithms (Soltani-Mohammadi et al., 2016). This is because the velocity function used by PSO algorithms can lead all particles to move towards the global solution area faster than SA algorithms (Li et al., 2018a). PSO is also well known for its simplicity, ease of implementation, fast convergence and fewer parameter adjustments compared to other algorithms (Binitha et al., 2012). MMO algorithms are modified from their unimodal versions by incorporating niching techniques to locate multiple optima and preserve them until the end of the search. MMO has attracted increasing interest from the bioinspired computation community, e.g., pedestrian detection (Li et al., 2016), multiple traffic signs recognition (Banharnsakun, 2018), non-cooperative target localisation (Liu et al., 2019), breast cancer prediction (Mohan et al., 2020), and embroidery inspection (Dong et al., 2011), to name a few.

2.3 Region-based CNN Approach

The classical machine learning approaches address object detection in two separate tasks: object classification and object localisation. Recently, a notable development in region-based CNN approaches, which unify object classification and object localisation into a single CNN network, has replaced the classical approaches. The state-of-the-art R-CNNs such as Faster R-CNN, Single Shot Multibox Detector (SSD), and You Only Look Once (YOLO) have been commonly used in palm tree detection (Zheng et al., 2021; Li et al., 2018b; Ammar et al., 2021).

The concept of region-based CNN was originally proposed by Girshick et al. (2014), a framework that combines a region proposal network and CNN, called R-CNN, to selectively extract around 2000 region proposals in an image to eliminate the need for an exhaustive selection of proposals. The region proposal stage generates class-agnostic proposals in the image to reduce the number of regions of interest (RoI) and to maintain meaningful proposals. However, R-CNN uses three separate models; feature extraction, bounding box classification, and regressing and fine-tuning the bounding box position and size. Classification of the CNN model will have to run around 2000 times per image, which compromises speed. An improved version called Fast R-CNN was proposed later by the same author (Girshick, 2015). Fast R-CNN combines the feature extraction, classification and bounding regression stages in a unified framework. Instead of feeding each region proposal to CNN, the entire image is fed into the CNN to obtain a convolutional feature map. The features are shared across the 2000 region proposals for subsequent RoI pooling and detection. This saves significant time on performing forward pass. However, the selective search (SS) algorithm used in Fast R-CNN is slow and time-consuming. Later on, Ren et al. (2015) proposed Faster R-RCNN to substitute the SS with a shallow CNN called region proposal network (RPN). It slides a small network over the feature maps by the last shared convolutional layer of CNN to generate proposals and their corresponding probabilities of containing object. This greatly improves the speed of modelling. The outputs of the RPN are passed to the Fast R-CNN component for final classification and bounding box regression. The framework can be considered as a combination of Fast R-CNN and RPN and is trained end-to-end. YOLO (Redmon et al., 2015) further improves the detection speed by unifying the bounding box prediction and CNN into a single neural network. The algorithm divides the input image into regions, and then it predicts the coordinates and probabilities of bounding boxes. Nevertheless, for a feature map of size $W \times H$ with k region proposals at each sliding location, there are still WHk region proposals in total to be processed.

Although the more efficient one stage region-based CNNs such as YOLO and SSD (Liu et al., 2015) unify ROIs generation and computation in the same deep network by sharing convolutional layers of the same data, such network architecture is not flexible to allow non-image data sources. Furthermore, the one-stage region-based CNNs would resize all input images to a fixed size. High-resolution UAV images' dimension is usually huge; resizing them to a fixed small dimension will cause problems in detecting small objects like palm trees. Although large images can be cropped into multiple smaller regions for detection, this is not an efficient approach because the small images have to pass through the convolutional network multiple times, and overlap between two images is required to detect objects which appear at the image edges. The latter results in more images to be processed and subsequently requires higher computation cost than processing a bigger image at once. As with any other variants of CNN, R-CNN requires long training hours, a large number of labelled images, and specialised hardware to run the model efficiently.

Chapter 3

Evaluation of Feature Extraction Methods for Classification of Palm Trees

3.1 Introduction

Palm tree detection aims to accurately classify and localise all palm trees in the input image. The accuracy of the classifier is crucial to avoid false positives and false negatives being detected during the localisation process. While there has been an increase in the number of studies using CNN for palm tree detection, there has been no comparative study addressing the accuracy and efficiency of feature extraction techniques for palm tree detection, either within human-engineered features or between human-engineered and CNN extracted features. CNN extracted features have the advantage of learning data representation through multiple processing layers without the user having to go through the complicated feature engineering stage (LeCun et al., 2015). However, it is well known that CNNs are memory hungry and require high computing facilities to train and run complex models; this will favour human-engineered features that require a lower computing footprint. As the existing feature extractors were not specifically designed for palm tree detection, this chapter presents a detailed comparative study of the features extracted from classical human-engineered approaches and the features extracted from CNN models in classifying palm tree images. The results provide a better understanding of each approach's advantages and limitations prior to utilising them in object localisation. The efficiency of the best combination will be further enhanced by the dimensionality reduction approach before applying it as the objective function for the multimodal PSO algorithm explained in the next chapters.

3.2 Methods

In this section, we briefly introduce the feature extraction methods used in this comparative study. Feature extraction methods transform original data into a more compact and representative feature vector, which is then fed into classifiers for training, and subsequently classifying objects in new images. A total of 6 extraction methods were tested in this work.

3.2.1 Human-Engineered Features

For human-engineered features in this study, we evaluated Histogram of Oriented Gradients (HOG), Local Binary Pattern (LBP) and Scale-Invariant Feature Transform (SIFT) features as recently reported in (Wang et al., 2019), (Rueda et al., 2016) and (Bazi et al., 2014), respectively, for palm tree detection.

Histogram of Oriented Gradients (HOG)

Histograms of oriented gradients (HOG) was introduced by Dalal (Dalal and Triggs, 2005) for pedestrian detection, but it can also be used for general object detections. The gradient information that comprises magnitude and orientation in each image pixel was computed to extract HOG features from the input image (Figure 3.1). Then, the gradient map was divided into 6 × 6 pixels non-overlapping cells. A 9-bin histogram summarizing the gradient information of each pixel within each cell was generated. The pixels were categorised into one of the bins depending on their orientation. Overlapping blocks composed of 2 × 2 cells were slid across the input image, a total of 36 feature vectors extracted by the 4 cells were concatenated into the final HOG feature vector. The overlapping ratio was 50% of the block size. The output of the HOG feature extraction was a histogram with 2916 dimensions (bins) for each image.

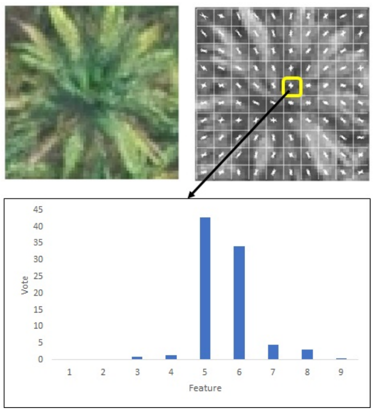


Figure 3.1: Transformation of pixel values into gradient information using HOG feature extractor

Local Binary Pattern (LBP)

Local binary patterns (LBP) was introduced by Ahonen et al. (2006), which is one of the commonly used techniques to describe the local texture characteristic of data due to its low computation cost and high accuracy. To extract LBP features from a sub-image, firstly the image was scanned by a 3 × 3 pixels window; each centre pixel in the window was compared with its 8 neighbours. If the neighbour's value was higher than the centre value, a "1" is denoted, "0" if not. Then, the 8-bit binary numbers were converted to decimal resulting in 256 possible values and assigned to the centre pixel, e.g. 10101010 was converted to 170 and subsequently produced a LBP feature map as illustrated in Figure 3.2. The generated LBP feature map was divided into 15 × 15 pixels non-overlapping blocks. In each block, a histogram comprising 59 bins was constructed following the method suggested in Ojala et al. (2002), where each bin summarised the frequency of occurrence of a unique uniform pattern containing at most two 0-1 or 1-0 transitions. Other non-uniform patterns were assigned to the 59th bin. The output of the LBP feature extraction was a histogram with 944 dimensions (bins) for each image formed by 16 blocks × 59 bins.

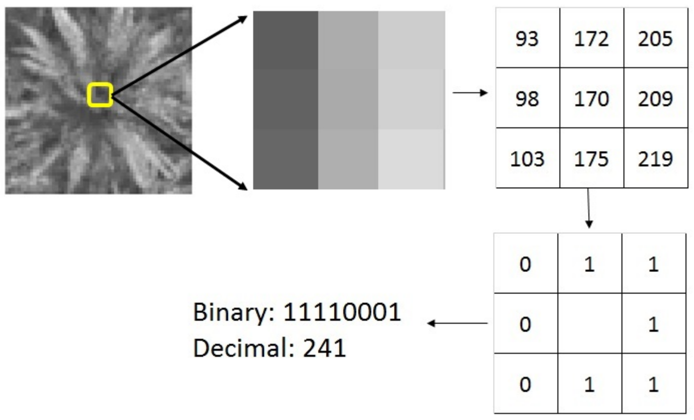


Figure 3.2: Transformation of pixel values into binary digit using LBP feature extractor

Scale-Invariant Feature Transform (SIFT)

Scale-invariant feature transform (SIFT) extractor, which was proposed by (Lowe, 1999) can to extract features that are invariant to scale, rotation, illumination, and translation. In the original SIFT extractor, the first step is to identify salient points from the input image. This is done by blurring the image in multiple scales by means of Gaussian operator. Then, the corners and edges are detected from the resultant image using the difference of Gaussians (DOG) approximation. Each extracted feature in the DOG map is compared with its 8 neighbours to select the local maximum or minimum point. Then, the 16 × 16 pixels surrounding the selected point are divided into sixteen 4 x 4 pixels subregions. An 8-bin histogram is used to summarise the gradient orientation and magnitude of the pixels within each subregion, and this forms the final 128 feature descriptor.

As the original SIFT operator generate random points in an image without storing the spatial information of the points. It is important to preserve the point's spatial information because each cell corresponds to a specific position of the palm tree. The combination of all points can only represent the structure of the whole palm tree. Instead of finding random salient points as proposed in the original SIFT, we generated uniformly and densely spaced points at 12 pixels step and summarised the gradient information on these points. The 128-feature descriptors extracted by all points were concatenated into one feature vector as shown in Figure 3.3. The output of the SIFT feature extraction was a histogram with 3200 dimensions (bins) for each image.

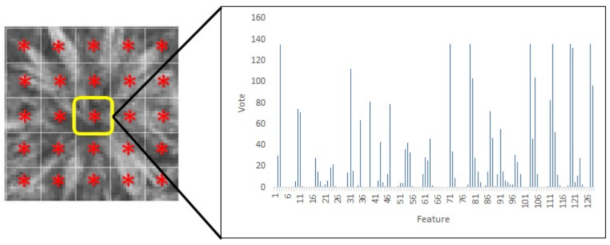


Figure 3.3: The sample of 128 feature descriptors computed by SIFT operator in the region of one point

3.2.2 CNN Extracted Features

The use of CNNs in object classification has burgeoned over the last decade since the introduction of the AlexNet model (Krizhevsky et al., 2012) which won the 2012 ILSVRC (ImageNet Large-Scale Visual Recognition Challenge). Since its inception, various forms and types of CNNs have been developed. There is a direct relationship between the number of convolutional layers and accuracy, but it trades off with computation time (Özgenel and Sorguç, 2018). We used the AlexNet model in this study as it was shown achieving up to 97% accuracy in palm tree detection in high-resolution remote sensing images (Li et al., 2017), which made it a preferred option for our experiment over other deeper networks that required higher execution time and computation resources.

The AlexNet architecture consists of five convolutional layers (Conv), some of which are followed by max-pooling layers, and finally three fully-connected (FC) layers as presented in Figure 3.4 and Table 3.1. It is important to note that there was no fine-tuning of the AlexNet model being carried out by back-propagating at this stage; the pre-trained model was only used as a feature extractor. The required features were extracted from the fifth convolutional layer (Conv 5), first, and second FC layers (FC6 and FC7) for classification. The resulting feature vector was then fed as input to the proposed classifiers. We examined whether the features extracted from the different layers performed differently.

3.2.3 Classifier used in this work

After the features were extracted from the training images, they were fed into a classifier for training. There are a large number of classification algorithms (at least 179 classifiers from 17 families according to the study by

Table 3.1: Details of the AlexNet architecture

Layer	Activation
Image Input Layer	227 x 227 x 3
Convolution (COV1)	55 x 55 x 96
ReLU	55 x 55 x 96
Cross Channel Normalization	55 x 55 x 96
Max Pooling	27 x 27 x 96
Convolution (COV2)	27 x 27 x 256
ReLU	27 x 27 x 256
Cross Channel Normalization	27 x 27 x 256
Max Pooling	13 x 13 x 256
Convolution (COV3)	13 x 13 x 384
ReLU	13 x 13 x 384
Convolution (COV4)	13 x 13 x 384
ReLU	13 x 13 x 384
Convolution (COV5)	13 x 13 x 384
ReLU	13 x 13 x 384
Max Pooling	6 x 6 x 256
Fully Connected (FC6)	1 x 1 x 4096
Relu	1 x 1 x 4096
Dropout	1 x 1 x 4096
Fully Connected (FC7)	1 x 1 x 4096
ReLU	1 x 1 x 4096
Dropout	1 x 1 x 4096
Fully Connected (FC8	1 x 1 x 2
Softmax	1 x 1 x 2
Classification output	-

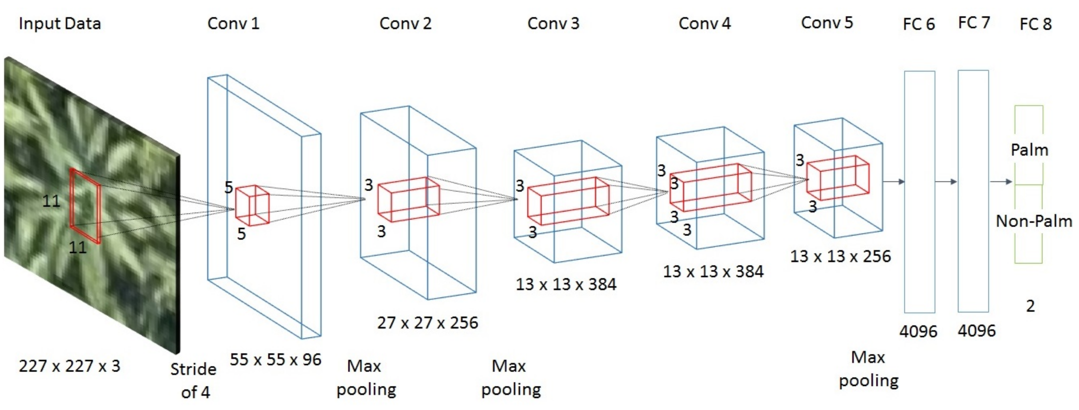


Figure 3.4: Architecture of AlexNet used in this study

Fernández-Delgado et al. (2014)) available for implementation, we selected Random Forest (RF) and SVM in this work for classification accuracy and computation runtime evaluations because they were the top two classifiers achieving high accuracy on the comparison over large data sets as reported in the same study.

Random Forests (RF)

Random forests (RF) was proposed by Breiman (2001), which combines a number of independent decision tree classifiers through bootstrap/bagging approach to make prediction more accurate. Each decision tree represents a subset of bootstrapped samples randomly drawn from the original training data set during training. Within a subset of randomly selected features at each node of the tree, the feature that best splits the data points is chosen for binary partitioning. The process is repeated recursively for each unsplit node until the stopping criterion is met. The training samples that are not used for building the trees are considered out-of-bag data and used to compute the prediction error rate. The prediction class of observation is determined by the majority vote of the decision trees for that observation (Equation 3.1). There are two parameters to be defined in RF: 1) the number of the decision tree, and 2) the number of features to be selected in

Table 3.2: Details of the feature extractors and classifiers tested in the study

Algorithm	Feature Extractor	Classifier	Feature Length
HOG (RF)	HOG	Random Forest	2916
LBP (RF)	LBP	Random Forest	944
SIFT (RF)	SIFT	Random Forest	2048
COV5(RF)	AlexNet COV 5	Random Forest	43264
FC6(RF)	AlexNet FC6	Random Forest	9216
FC7(RF)	AlexNet FC7	Random Forest	4096
HOG (SVML)	HOG	SVM (Linear Kernel)	2916
LBP (SVML)	LBP	SVM (Linear Kernel)	944
SIFT (SVML)	SIFT	SVM (Linear Kernel)	2048
COV5(SVML)	AlexNet COV 5	SVM (Linear Kernel)	43264
FC6(SVML)	AlexNet FC6	SVM (Linear Kernel)	9216
FC7(SVML)	AlexNet FC7	SVM (Linear Kernel)	4096
HOG (SVMRBF)	HOG	SVM (RBF Kernel)	2916
LBP (SVMRBF)	LBP	SVM (RBF Kernel)	944
SIFT (SVMRBF)	SIFT	SVM (RBF Kernel)	2048
COV5(SVMRBF)	AlexNet COV 5	SVM (RBF Kernel)	43264
FC6(SVMRBF)	AlexNet FC6	SVM (RBF Kernel)	9216
FC7(SVMRBF)	AlexNet FC7	SVM (RBF Kernel)	4096
VJ	HAAR	Adaboost	-
Alexnet	Alexnet	SoftMax	-

each split. We set the number of the decision tree as 100 because the out-of-bag data prediction error rate converged and stabilised at this level. The subset of features to be selected in each split was determined by the typical recommendation, which is $\sqrt{p}$, where p is the total number of features extracted by the feature extractor.

$$f(x) = \arg\max_{y} \sum_{i=1}^{N} I(h_i(x) = y) \tag{3.1}$$

where $h_i(x)$ is the prediction of the response variable x using ith tree. $I(h_i(x) = y) = 1$ if $h_i(x) = 1$ and 0 otherwise.

Support Vector Machine (SVM)

Cortes and Vapnik (1995) proposed support vector machine classifier (SVM), which determines the optimal hyperplane that best separates two classes.

For our case, the trained classifier predicts on which side of the hyperplane the new input image falls, which is to classify it to class "1" (palm tree) or "-1" (non-palm tree) following the distance function below:

$$f(x) = sgn(\sum_{i=1}^{M} \alpha_i y_i x^T x_i + b) \tag{3.2}$$

where (x_i, y_i) is the trained feature with its class, $y \in \{-1, 1\}$ and α and b are trained weights that were fined-tune using the default setting in Matlab. We tried to optimised the trained weights instead of using the default values but failed. The model performance deteriorated miserably despite the fact that the best weights were being used. This was probably due to over-fitting issues and will be addressed in the future. In addition to the linear kernel, a non-linear radial basis function (RBF) kernel was also used for the evaluation. RBF kernel maps the original features into higher dimensional space in which they are linearly separable through similarity comparisons without explicitly transforming the data. The RBF is expressed as:

$$K(x, x\prime) = \exp(-\gamma \|x - x\prime\|^2) \tag{3.3}$$

The solution can be computed by:

$$f(x) = sgn(\sum_{i=1}^{M} \alpha_i y_i K(x, x_i) + b) \tag{3.4}$$

The solution can be expressed as:

$$f(x) = sgn(\sum_{i=1}^{N} y_i \alpha_i K(x, x_i) + b) \tag{3.5}$$

3.2.4 Baselines For Comparison

We selected the off-the-shelf Viola-Jones framework and AlexNet CNN model available in Matlab machine learning packages as baselines for comparison. These object detection algorithms integrate feature extraction and classification algorithms in the same model. The components in the models have been fine-tuned by default achieving state-of-the-art performance.

Viola-Jones Algorithm (VJ)

Viola and Jones (2001) proposed to use Haar-like features that are represented by a set of rectangles to compute the intensity difference between two regions in an image. For example, the intensity difference value is obtained by subtracting the sum of pixel values in the black region from the sum of pixel values in the white region, as shown in Figure 3.5. The raw image is transformed into an integral image to speed up the processing time. The VJ framework uses AdaBoost to combine a set of weak classifiers to become final strong classifier, as shown in Equation 3.6. Each weak classifier consists of features that best segregate the dataset into two. A cascade architecture is also integrated into the framework where at each stage a set of classifiers are trained and built to detect almost all of the positive samples while rejecting a small fraction of non-positive samples. The samples that are not rejected by the stage classifiers will be processed by the next stage classifiers. Via this architecture, a lot of negative samples can be rejected at the early stages, therefore the detection speed can be increased by focusing on the promising regions in the images.

$$f(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t f_t(x) \geq \theta_t \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

where $f_t(x)$ is the weak classifier at t stage, α_t is the ensemble weight, and θ_t is the detection rate, both are obtained through the learning process in Matlab. We set 20 stages for training until the algorithm converged at the 13th stage. The threshold for allowable false alarm rate per stage was set at 0.5.

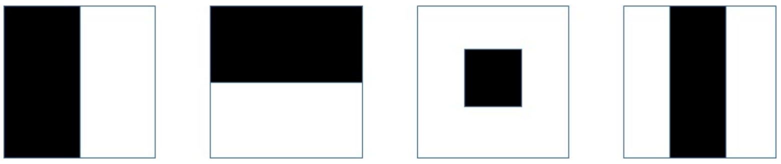


Figure 3.5: Examples of the Haar-like features used in the experiment

AlexNet model

For this approach, the fully-connected pre-trained AlexNet model was used. Since re-training the whole model from scratch requires a large number of training samples and time to search for the optimal parameters and settings, the transfer learning approach (Yosinski et al., 2014) was applied to fine-tune the pre-trained AlexNet model using our palm tree training samples for only focusing on palm tree classification task. Using the approach, we only optimised the weights of the last three FC layers while freezing the weights of other layers during re-training. The weights were updated by back-propagating from the SoftMax function using stochastic gradient descent (SGD) algorithm with the default cross-entropy log loss. Originally, the last FC layer was fed to a 1000-way SoftMax that produced classification results for 1000 classes, it was modified to only two in our study. The output SoftMax classification function can be formulated as

$$P(y=j|z_i) = \frac{e^{z_i}}{\sum_{j=0}^{N} e^{z_j}} \tag{3.7}$$

where $z = W^T X + b$, W and b correspond to trained weights and bias respectively, and X is the input feature vector of a sample. The function computes the probability that sample X belongs to class j given the weights and bias. The model ended up with 2 classification outputs, which was either palm tree or non-palm tree. The training accuracy reached almost 100% after the 5th epoch, as shown in Figure 3.6, with a learning rate set at 10^{-4}, and a mini-batch size set at 10.

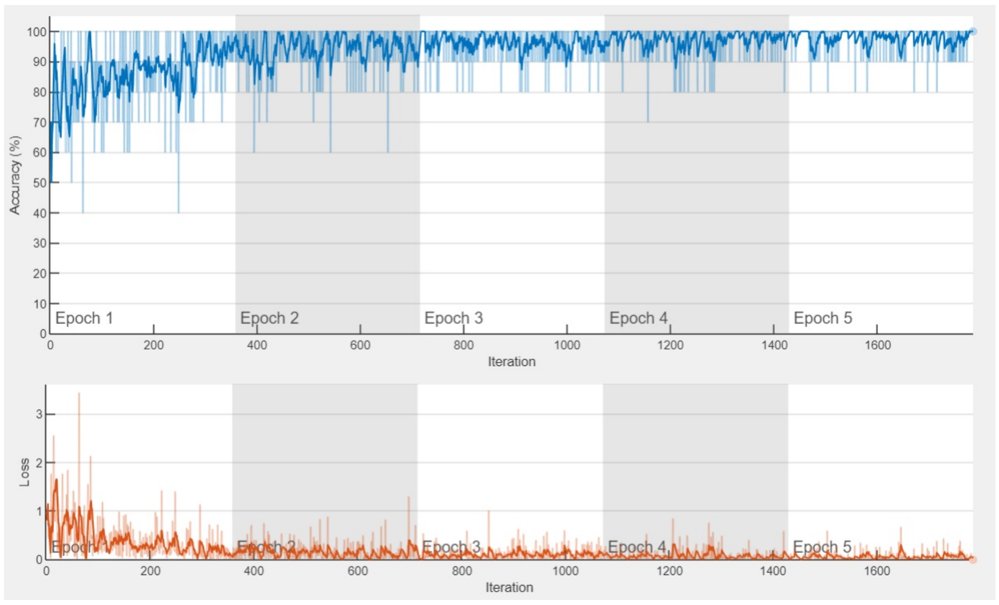
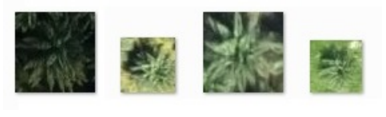


Figure 3.6: Training accuracy of the proposed AlexNet model

3.2.5 Implementation Details

Description of data

The UAV images used for this study were acquired using a DJI Phantom 4 quad-copter drone equipped with 12.4 megapixels RGB camera taken over a few plantations comprising flat to hilly terrain located in Malaysia. The images were taken between 9 a.m. to 5 p.m. at an altitude of 200 m above ground with 80% side and frontal overlaps. The aerial photos were post-processed using Pix4D Desktop Professional to produce 10 cm/pixel orthophoto, which was georectified to the correct location and scale. A total of 2294 positive and 2734 negative images inclusive of mature trees (>3 years old after planting), immature trees ($\leq$ 3 years old after planting), and other non-palm tree objects were manually cropped from the full-scale high-resolution UAV images. Samples with different illumination conditions were included to increase the diversity of data for training. Each training sample was resized to 60 $\times$ 60 pixels and converted to greyscale for feature extraction using human-engineered approaches. They were resized to 227 $\times$ 227 pixels for CNN approaches and 24 $\times$ 24 pixels for the VJ framework. The samples were split into 70% for training and 30% for validation. A few examples are shown in Figure 3.7.


(a) Positive samples.

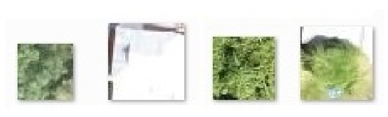

(b) Negative samples

Figure 3.7: Training samples used in the study.

The experiments were implemented in MATLAB R2020a by utilising the libraries as shown in Table 3.3 for feature extraction. The experiments were performed on a notebook computer with a Four-Core Intel i7 (TM) 10610U 1.80GHz CPU, 16 GB of DDR3 RAM, Intel UHD Graphics, and OS Windows 10.

Table 3.3: Sources of the library used for feature extraction

Feature extraction method	Library
HOG	MATLAB Computer Vision Toolbox
LBP	MATLAB Computer Vision Toolbox
SIFT	VLFeat (Vedaldi and Fulkerson, 2010)
CNN	MATLAB Deep Learning Toolbox
HAAR	MATLAB Computer Vision Toolbox

Evaluation Methods

The model performance was evaluated based on precision, recall, F1-score, success rate (SR), and runtime. Precision is defined as the proportion of correctly classified palm trees out of all detected palm trees (including the wrongly classified ones). The recall is defined as the proportion of palm trees correctly classified out of all known palm trees in the images. F1-score is the average of both measures. SR measures the percentage of successful detection (all positive and negative samples) out of all samples. If the detection is correct, it is coded as "1", otherwise "0". Runtime is the computation cost in milliseconds (ms) for an algorithm to perform feature extraction and classification in a single run.

$$Precision = \frac{TP}{TP + FP} \tag{3.8}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.9}$$

$$F1 - score = \frac{2.Precision.Recall}{Precision + Recall} \tag{3.10}$$

$$SR = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.11}$$

where TP stands for true positive, which is the image correctly classified as the palm tree. TN stands for true negative which is the image correctly classified as a non-palm tree. FP stands for false positive which is the image falsely classified as the palm tree, and FN stands for false negative, which is falsely classified as the non-palm tree.

3.3 Results and Discussion

Statistical analyses were performed using SPSS Statistics software to determine if there were any statistical difference between the tested models in terms of SR and runtime. Pearson's chi-square test was performed on SR results to compare the observed and expected frequencies. Since the runtime results were not normally distributed, the non parametric Kruskal-Wallis method with Dunn post-hoc pair-wise statistical test that does not assume normality was performed to compare the models. The results are shown in Table 3.4, where symbols + and − indicate the competitor is significantly better than or worse than AlexNet, respectively at $\alpha = 0.05$.

The table results show that the AlexNet model was the best performing model with a 96.45% F1-score and significantly higher SR at 96.76%, which shows its clear superiority in classifying most samples correctly with respect to the remaining models. On the other hand, the VJ algorithm achieved the poorest performance among all methods with a 66.97% F1-score and 75.12% SR. The reason for the poor accuracy is mainly attributed to the low recall rate at only 55.29%, where almost half of the palm tree samples were rejected as non-palm tree samples. The poor recall rate may be due to the high sensitivity of the algorithm to translation and shape changes. The images with palm trees slightly displaced from the image centre were falsely classified as negatives. In addition, many young palm trees with slightly different canopy structures from that of mature palm trees were also missed out. These findings suggests that two sets of VJ algorithms may be required to effectively differentiate mature and young palms. COV5(SVMRBF) was the second-best model, and its F1-Score and SR was only about 1%, slightly below the optimised AlexNet. This proves that the features extracted from the pre-trained COV5 layer, which describes lower level details of the input image, can be fed into SVM for palm

Table 3.4: The quantitative accuracy assessment results of different palm tree extraction approaches

Algorithm	**Recall (%)**	**Precision (%)**	**F1-Score (%)**	**Success Rate (%)**	**Runtime (ms)**
AlexNet	96.53	96.38	96.45	96.76	21.27
COV5(SVMRBF)	94.56	96.31	95.43	95.86 -	369.46 -
SIFT (SVMRBF)	95.62	93.92	94.76	95.18 -	20.69 +
FC6(SVMRBF)	93.50	95.82	94.65	95.18 -	60.08 -
COV5(SVML)	94.11	94.54	94.32	94.83 -	198.17 -
FC7(SVMRBF)	92.75	95.49	94.10	94.69 -	61.75 -
LBP (SVMRBF)	92.30	95.92	94.07	94.69 -	7.37 +
COV5(RF)	91.84	93.54	92.68	93.38 -	205.92 -
FC6(RF)	92.30	92.72	92.51	93.18 -	186.43 -
LBP (RF)	90.94	93.77	92.33	93.11 -	101.39 -
FC7(RF)	91.69	92.67	92.18	92.90 -	196.08 -
HOG (SVMRBF)	89.58	94.28	91.87	92.76 -	31.53 -
FC6(SVML)	91.54	91.96	91.75	92.49 -	38.50 -
FC7(SVML)	91.39	91.53	91.46	92.21 -	41.16 -
SIFT (RF)	90.33	92.43	91.37	92.21 -	95.09 -
SIFT (SVML)	90.18	90.18	90.18	91.04 -	14.37 +
LBP (SVML)	89.27	89.68	89.48	89.80 -	5.07 +
HOG (RF)	85.35	89.97	87.60	88.97 -	103.50 -
HOG (SVML)	85.35	85.61	85.48	86.77 -	15.82 +
VJ	55.29	84.92	66.97	75.12 -	0.58 +

Symbols + and − indicate that the competitor is respectively better than or worse than AlexNet according to the Pearson's chi-square test (SR) or Kruskal-Wallis with post-hoc Dunn tests (runtime) at $\alpha = 0.05$.

tree classification and obtain competitive result. SIFT(SVMRBF) was the third-best model at 95.18% F1-Score, which was also the best model to achieve the highest accuracy amongst all human-engineered approaches. Its accuracy was about 2% lower than that of AlexNet.

We summarised the performance of each feature extractor by averaging its F1-Score obtained from different classifiers and the results are presented in Figure 3.8. Among all feature extractors, COV5 yielded the best result. We observe that the accuracy of CNN-based features reduced gradually from COV5 to FC7, which concurs with the findings by Liu et al. (2018). We infer that the layers closer to the final classification layer (i.e. FC7) are more optimised to identify certain objects, i.e. the pre-defined 1000 classes. Therefore, the feature representation of the later layers (i.e. FC6 and FC7) are more specific to describe the original 1000 classes in the unoptimised model. Since none of the categories in the pre-trained AlexNet model is related to the palm tree, directly extracting the features from the deeper layers produced inferior results than the shallower one.

SIFT was the best feature extractor in human-engineered methods, followed by LBP and HOG. The accuracy of HOG was the lowest, it may be due to the fact that the extractor limited the information shared between image cells. For instance, each SIFT point described the gradient information from a subregion that contains 16 $\times$ 16 pixels. Since the subregions of the neighbouring SIFT points overlap, their information could be shared. Similarly, in the LBP feature extraction process, each image was divided into 3 $\times$ 3 pixels overlapping windows, and each center pixel in the window was compared with its 8 neighbours. However in HOG extractor, the image was already divided into non-overlapping cells since the beginning, limiting the information shared between the cells. This is one of the major disadvantages of human-engineered method where the user needs to carefully design the feature extractor to solve specific problem based on certain assumptions. The advantage of CNN is that it omits the feature designing steps.

We also summarised the performance of each classifier by averaging its F1-Score obtained from different feature extractors, and the results are presented in Figure 3.9. The best performing classifier was SVMRBF, followed by RF and SVML. The SVMRBF and RF performed better than SVML, suggesting that the palm tree and non-palm tree data are not linearly separable, which gave the SVM with non-linear kernel and random forest a leg-up over the linear SVM algorithm. The result also proves that SVM with RBF kernel was more suitable than a random forest to classify palm tree images in our study.

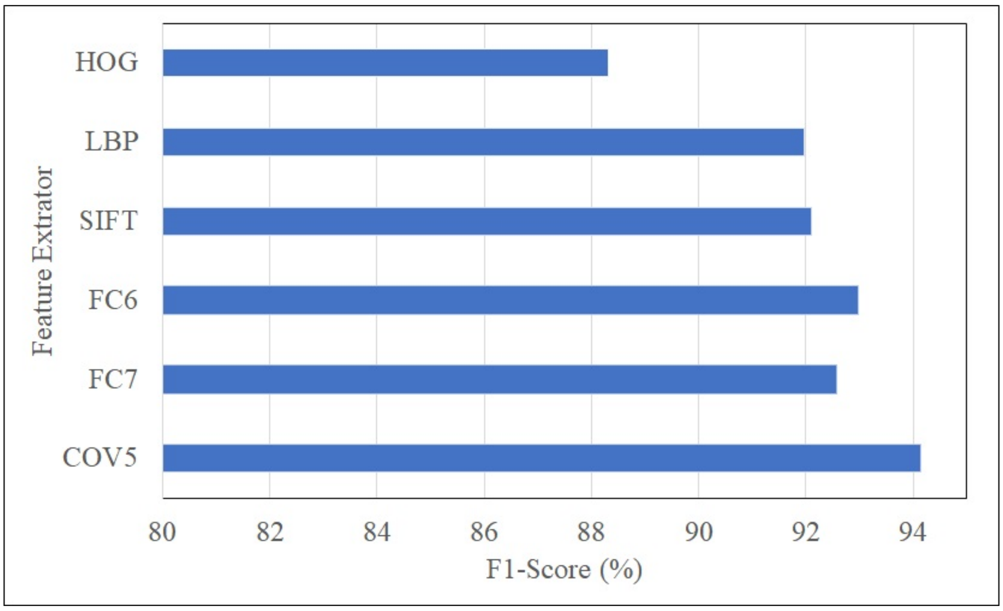


Figure 3.8: Comparison of performance between feature extractors, averaging over different classifiers

In terms of time efficiency, as expected, the VJ algorithm with HAAR-like features achieved the fastest computation speed due to the implementation of integral image and cascade algorithm, which cut down the computation efforts significantly. Even though COV5(SVMRBF) achieved the second-best result, its accuracy traded off with speed as it was the slowest algorithm because it contained the most features (43264) and required more time for computation. The accuracy of SIFT(SVMRBF) was slightly lower than COV5(SVMRBF), but its computation speed was almost 18 times faster than COV5(SVMRBF). However, it was still not the best option for palm tree detection as its computation speed was only marginally faster (although significant) than AlexNet that attained higher accuracy. LBP(SVMRBF) was the next best option for palm tree detection since it balanced accuracy and speed. Although LBP(SVMRBF)'s accuracy was about 3% lower than AlexNet, its computation speed was 3 times faster, significantly saving computation cost if applied in large-scale UAV images. Its training time was also significantly faster than AlexNet's training time (2 seconds VS 2 hours).

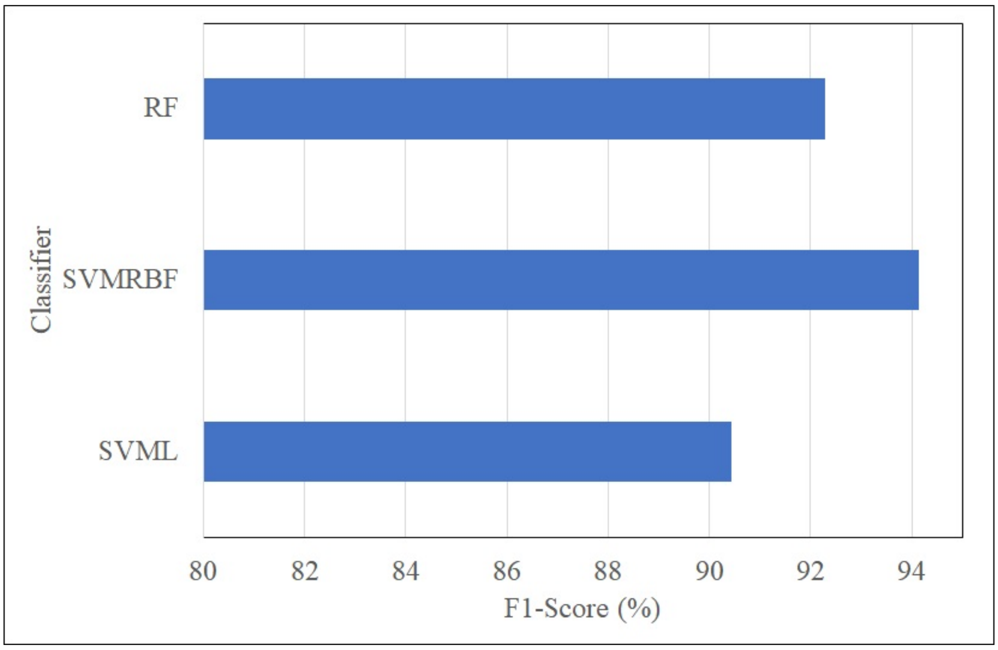


Figure 3.9: Comparison of performance between classifiers, averaging over different feature extractors

Likewise, we summarised the computation speed of feature extractors and presented them in Figure 3.11. COV5 was the slowest method as it contained the longest feature vector. Therefore, both SVM and random forest required higher computation costs to perform kernel calculations, and search through all nodes in the decision trees, respectively, to identify the correct class. LBP was the fastest as it had the shortest feature vector (944), yet achieving comparable results to the AlexNet methods. It proves that the discrimination power of the LBP extractor was sufficient to classify palm trees correctly despite having the shortest feature vector. Comparing the computation time of different classifiers, on average, RF performed the slowest while SVML performed the fastest. The computation complexity of RF is affected by the number of decision trees used in the ensemble model. Although we can reduce the number of decision trees in the model to speed up computation time, it may reduce accuracy. The SVML was the fastest algorithm because it did not need to compute the kernel function as in Equation 3.3.

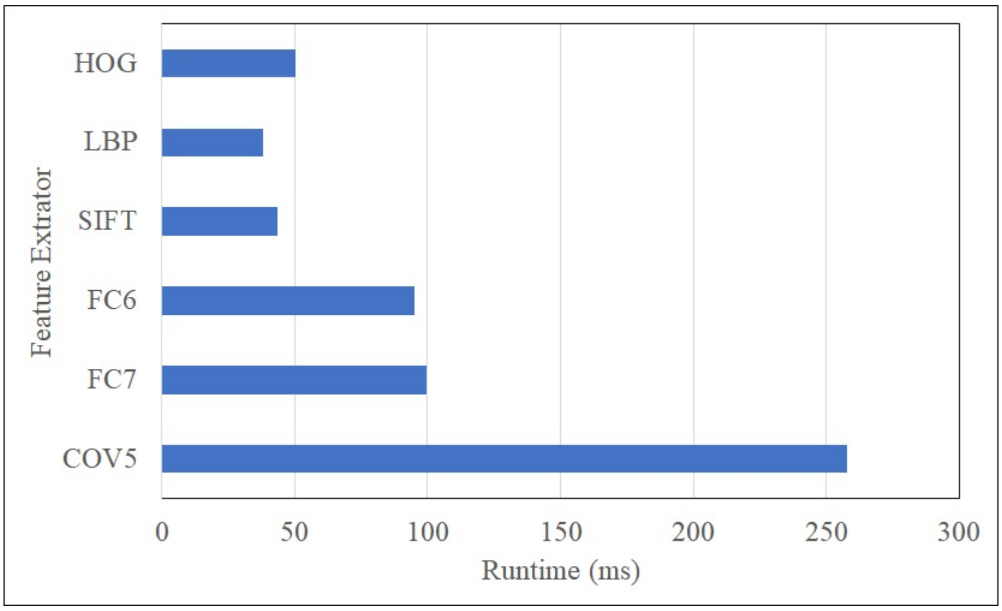


Figure 3.10: Comparison of computation time between feature extractors, averaging over different classifiers

3.4 Dimensionality reduction of LBP features

Based on the results in the previous sections, it was shown that the combination of LBP extractor and SVM (RBF kernel) achieved a balance between accuracy and computation speed to classify palm tree images. The next research question that we would like to ask is whether the performance of LBP can be further enhanced. This section describes an extended trial that utilises a feature selection approach to boost the LBP's performance by reducing its feature dimension. Feature selection algorithm reduces data dimensionality by skimming off the important and less redundant, which is useful to speed up classification algorithm and improve prediction accuracy (Huan Liu and Lei Yu, 2005). Feature selection can be generally categorised into three categories: filter, wrapper and hybrid methods. The filter methods select features based on the statistical scores for their correlation to the predictors without using any classifier. For example, *t*-test can be used to evaluate whether the values of a particular feature for class 1 is significantly different from the values of the same feature for class 2

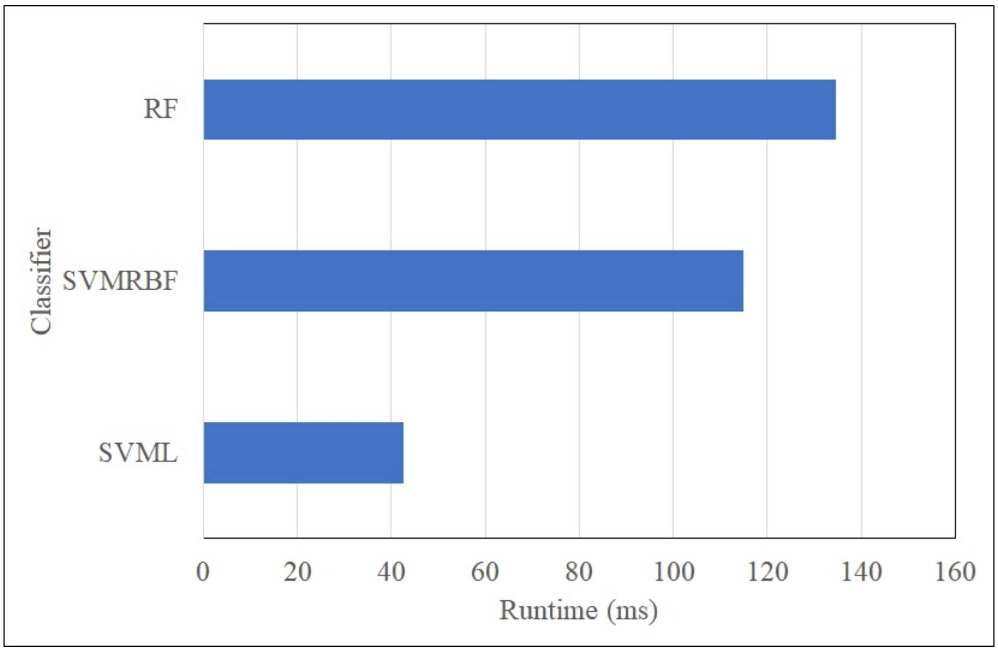


Figure 3.11: Comparison of computation time between classifiers, averaging over different feature extractors

(Wang et al., 2012a). If this hold, the feature is useful to differentiate our data. Filter methods do not consider the correlation between each feature and are usually fast to compute, which is useful in pre-processing stage to discard low-quality features.

The concept of wrapper methods is similar to filter methods. The main difference is that the selected features are fed into the machine learning classifier for training and cross-validated on test data. The process is repeated to test all possible combinations of subset, and the changes in the model performance for each combination are evaluated. The wrapper methods are usually computationally more expensive than filter methods. In the cases where the feature dimension is very large, filter methods are firstly applied to obtain a reduced number of features before passing them to the more sophisticated feature selection algorithms for processing. The hybrid methods combine the characteristics of both methods in a unified algorithm to perform a heuristic search in the space of all possible features.

3.4.1 Methods

In this study, we used the filter method to compute *t*-statistics for every feature. Then, the features were sorted based on each feature's p-value from the lowest (most significant) to highest (least significant), and added into a list S. After that, the best feature in S was removed from S and added into subset T one at a time. The features in T were passed to SVM (RBF) classifier for training. The cross-validation performance of the trained model for each addition of feature was evaluated. The process was repeated until all of the features in S were tested.

3.4.2 Results and discussion

Figure 3.12 illustrates the cross-validation error of the model against the number of features used for training. It shows that the cross-validation error stabilised when about 450 best features were used. This means that the top 450 features possess the equivalent description ability as the original 944 features. Adding more features to the data may cause redundancy and will reduce computation speed.

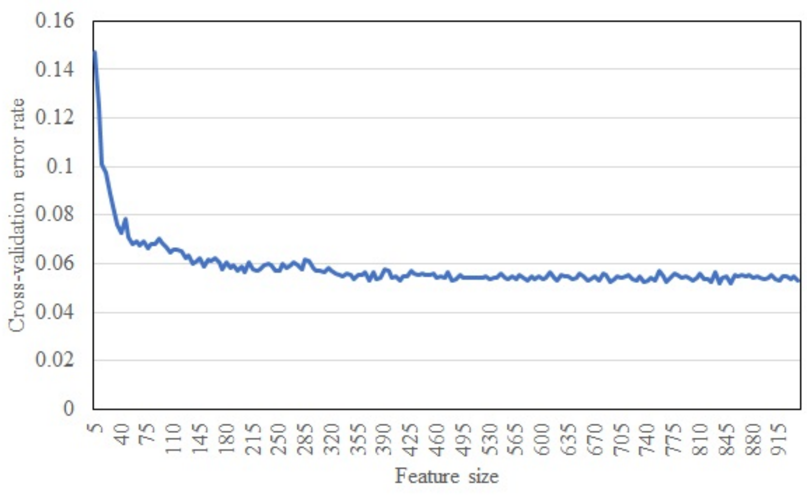


Figure 3.12: Cross-validation performance against the number of LBP features used for classification

Table 3.5: Comparison of accuracy and computation runtime of LBP features before and after dimensionality reduction.

Algorithm	Recall (%)	Precision (%)	F1-Score (%)	Success Rate (%)	Runtime (ms)
LBP(SVMRBF)-R	92.75	95.34	94.03	94.62	3.86*
LBP(SVMRBF)	92.30	95.92	94.07	94.69	7.37

LBP(SVMRBF)-R denotes LBP features after dimensionality reduction while LBP(SVMRBF) denotes the original LBP features. Symbols $*$ indicates that the result is significantly different based Kruskal-Wallis tests at $\alpha = 0.05$.

Table 3.5 shows the performance of LBP with reduced features (LBP(SVMRBF)-R) and the LBP with original features (LBP(SVMRBF)). The accuracy of both models was similar. However, LBP(SVMRBF)-R was significantly faster than LBP(SVMRBF), where it only required half of the computation cost of LBP(SVMRBF) to carry out the same classification task. This shows that our approach reduced the computation cost of LBP(SVMRBF) without compromising its accuracy.

3.5 Summary

This chapter evaluated the performance of six feature extraction methods in combination with three classifiers for palm tree classification. They were also compared with the optimised AlexNet model and Viola-Jones algorithm. The classification accuracy reached by the AlexNet model optimised with our palm tree images was significantly superior to that obtained by all other models. Nevertheless, we suggest adopting LBP with the SVM (RBF kernel) classifier for palm tree detection because its computation speed was 3 times faster than AlexNet, albeit with a 3% lower accuracy. For palm tree detection in large-scale and high-resolution, a balance between accuracy and computation speed is necessary to complete the task in time with minimum errors. LBP feature extractor with simple implementation also requires a lesser memory footprint than CNN models. We had also obtained other interesting findings: (1) both random forest and SVM (RBF kernel) obtained higher accuracy than SVM (linear kernel) indicating that palm tree features are non-linearly separable, (2) the weights of the deeper layers that are closer to the classification layer in CNN are optimised to detect target objects, they needed to be fine-tuned if the object of interest is not encompassed in the predefined categories , (3) HOG achieved the lowest accuracy in our study probably because the information is not effectively shared between subregions as opposed to LBP and SIFT, and (4) the pre-trained CNN model can be utilised as feature extractor without requiring the user to go through the complicated feature construction processes. However, if a domain expert properly design a feature extractor to solve a specific problem, it can achieve similar performance as the CNN model, but with faster training and computation speed.

We also presented an enhanced LBP(SVMRBF) model, achieved through dimensionality reduction using feature selection methods. The enhanced model achieved similar accuracy as the original one, but only required half of the computation cost. It was also seven times faster than the fully-connected AlexNet, which was a significant improvement for palm tree detection in large-scale UAV images. The number of features may further be reduced by filtering the correlated features and selecting the best combination using the hybrid feature selection approach, which is a more computationally expensive method.

At the time of writing the thesis, we observed that our results contradicted the latest results reported by Zheng et al. (2021). The authors claimed that SVM- and random forest- based classifiers were the worst performing algorithms, achieving less than 30% F1-score for multi-class classification and about 86% for overall palm tree detection. We postulated their dismal results were due to: (1) over-fitting of the model as they used more than 86,000 samples for training, whereas we only used about 3,500 samples, (2) limited number of sliding window sizes, and (3) unoptimised parameters for SVM and RF. The authors did not report the feature extractor that they used in conjunction with the SVM and RF classifiers. If the raw image data were used, the classifiers would probably return unsatisfactory results. In the future, we plan to test our algorithms on the datasets provided by the authors, which are freely available for public use.

Chapter 4

Improved Multimodal Particle Swarm Optimisation Algorithm

4.1 Introduction

As our aim of this thesis is to develop an efficient algorithm for palm tree detection in UAV images, we leverage the species-based multimodal PSO (SPSO) as the main framework for our study because PSO was proven to be faster and simpler, yet achieving similar accuracy as the other bio-inspired algorithms. In many MMO algorithms including SPSO, repetitive distance evaluations to search nearest neighbours to find multiple optima in the search space are unavoidable, which is time consuming. The overhead increases exponentially if the population size is large. In this chapter, we propose to speed up the nearest neighbour search calculation by introducing a special tree-based structure, called kd-tree, into SPSO to improve the efficiency of the nearest neighbour search.

First, we discuss the details of the basic unimodal and multimodal PSO algorithms. Then, we describe the limitation in the algorithm's search process and introduce the k-d tree search mechanism. We apply the improved multimodal PSO, called KDT-SPSO, to multimodal benchmark functions to demonstrate the effectiveness of our proposed method in solving general multimodal optimisation tasks.

4.2 Background

4.2.1 Particle Swarm Optimisation (PSO)

In this section, we present in detail the particle swarm optimisation (PSO) algorithm. It is important to include this section in this chapter as it forms the basis of our work for this thesis. PSO algorithm, originally proposed by Kennedy and Eberhart (1995), has been found robust and fast in solving non-linear and non-differentiable problems. It was inspired by the social and cooperative behaviour of the swarms in nature, such as birds and fish. The PSO algorithm maintains a population of particles (swarm), where each particle represents a candidate solution in a multidimensional search space. The particles start at random initial positions and explore through the search space with variable velocity to find the minimum or maximum of a given objective function. Each particle has memory to keep track of its current position, previous best position and velocity. The whole swarm shares the overall best position found among all the particles, known as global best. The steps of the PSO algorithm are summarized as follows and its pseudocode is presented in Algorithm 1:

1. **Initialisation**: Generate a population of N particles with random positions and velocities in the search space.

2. **Evaluation and best positions update**: The fitness value of all particles in the population are computed based on the given objective function. A particle's personal best position is replaced if its current fitness value is better than the last one. The population's best position is also replaced if the current fitness value found so far by any of the particles is better than the last one.

3. **Velocity and position updates**: In each iteration, each particle's velocity and position are updated according to the following formula, respectively:

$$V_i(t+1) = \gamma V_i(t) + c_1\beta_1(P_i(t) - X_i(t)) + c_2\beta_2(P_g(t) - X_i(t)) \quad (4.1)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (4.2)$$

where $V_i(t)$ denotes the rate of position change; $X_i(t)$ is the position of particle i at iteration t, where $X_i(t)$ has K dimensions. The velocity vector encodes the amount of change in position and direction while the position vector shows the location of a particle; γ represents the inertial weight introduced by (Shi and Eberhart, 1998) into the original PSO to balance the global and local search; c_1 and c_2 stand for cognitive and social coefficients to balance the exploration and exploitation; β_1 and β_2 are uniformly generated random numbers in the range [0,1]; $P_i(t)$ refers to the position with the best value recorded so far by the particle i (*pbest*); P_g denotes the position of the global best particle, which is the particle with the best value found so far in the entire swarm and will be shared by all the particles (*gbest*). The new position of particle i is calculated using Equation 4.2.

4. **Termination**: Steps 2 and 3 are repeated until a stopping criterion is reached.

Algorithm 1 Algorithm of a standard PSO

```
1: Randomly initialise $X_i(t)$ for all $i$, where $t = 0$;
2: Evaluate $f(X_i(t))$;
3: Assign $P_i = X_i(t)$;
4: Assign $P_g = P_j$, where $j = \arg\max_i \{f(P_i)\}$
5: while termination criteria is not met do
6:     for each particle $i$ in the population do
7:         Equation 4.1;
8:         Equation 4.2;
9:         Evaluate $f(X_i(t))$;
10:        if $f(X_i(t)) > f(P_i)$ then
11:            Assign $P_i = X_i(t)$;
12:        end if
13:    end for
14:    $j = \arg\max_i \{f(P_i)\}$;
15:    if $f(P_g) < f(P_j)$ then
16:        Assign $P_g = P_j$;
17:    end if
18: end while
```

4.2.2 PSO for Multimodal Optimisation

The classical unimodal PSO is not ideal for solving multimodal problems because it is likely to guide all particles to converge to one of the optima (*gbest*). It can be extended to deal with multimodal problems using a niching technique that allows each particle to be influenced by the best-fit particle selected from its neighbours. Using this mechanism, the particles are able to search for different regions simultaneously and converge to multiple optima in the search space. Niching is one of the earlier methods developed for GA to keep the diversity of populations(Goldberg and Richardson, 1987). Niches represent multiple global bests/peaks across the search space while niching is used to group similar particles into subswarms and maintain them around each peak until the end of the search (Li et al., 2018a). Numerous niching methods have been proposed, such as mutation, crowding, fitness sharing, clustering, speciation, etc. (see (Barrera and Coello, 2009) and (Li et al., 2018a) for a complete discussion). Among those, speciation has been commonly used in the literature due to its effectiveness and efficiency for exploring multiple solutions compared to other techniques (Parrott and Xiaodong Li, 2006; Luo et al., 2016; Huang et al., 2017).

Parrott and Xiaodong Li (2004) created the first species-based PSO (SPSO). A species is a group of particles in a population with similar properties, and the particle with the highest fitness value in the group is chosen as the species seed. SPSO aims to identify multiple species within a population; the *pbest* position of a species seed is adopted as the *gbest* position by all other particles within the species. The similarity metric, on the other hand, can be measured by Euclidean distance; the smaller the Euclidean distance between two particles, the more similar they are:

$$d(X_i, X_j) = \sqrt{\sum_{k=1}^{K} (X_i^k - X_j^k)^2} \tag{4.3}$$

where $d(X_i, X_j)$ is the distance between two particles. A particle X_i is grouped into a species if its distance is within a user specified radius r from the species seed. The value of r is critical to the performance of SPSO. If it is too large, it encourages particles to converge to only a few global optima. On the contrary, if it is too small, it will introduce the possibility of premature convergence as each particle may become a seed. The framework of the SPSO is summarised as follows and its pseudocode is presented in Algorithm 2:

1. **Initialisation**: A population of N particles is initialised with random velocity and position.

2. **Evaluation and determine species seeds**: The species seed set S is initially empty. All particles are evaluated, sorted in decreasing order of fitness and added to a list X_{sort}. All particles in X_{sort} are evaluated one after another (from best to least-fit) against all species seeds found so far in S. If S does not contain any seed that is closer than a radius r to the particle considered, the particle will be selected as new seed and added to S. Otherwise, it will become a member belonging to one of the seeds. Since the species seeds in S are sorted in the order of decreasing fitness, the first identified seed will dominate over other seeds identified later. For example, particles will be allocated to the more highly fit seeds first before the less fit ones.

3. **Velocity and position update**: Each particle's velocity and position are updated according to Equations 4.1 and 4.2. Instead of following $gbest$ (P_g), each particle is guided by its local best particle ($lbest$), the species seed.

4. **Termination**: Steps 2 and 3 are repeated until a stop criterion is met.

4.3 Optimising SPSO using k-d tree structure

The SPSO framework proposed by Parrott and Xiaodong Li (2004) can adapt better for multimodal optimisation problems than the global ones. However, it is still a challenging task to determine species members because the Euclidean distance calculations of all particles pairs have to be performed at every iteration. The complexity of the calculations is affected by the species radius r (Al-Ruzouq et al., 2018). Assuming there are N

Algorithm 2 Algorithm for determining species seeds and members in a population

```
1: $S = \emptyset$;
2: Sort $X$ in decreasing fitness values, where $X = \{X_1, X_2, ..., X_N\}$, obtaining $X_{sort}$;
3: while (there are unprocessed particle in $X_{sort}$) do
4:     Search for the best unprocessed $X_i \in X_{sort}$;
5:     $found \leftarrow false$;
6:     for each seed $s$ in $S$ do
7:         if $d(X_i, s) \leq r$ then
8:             Assign $s$ as $lbest$ for $X_i$
9:             $found \leftarrow true$;
10:            Break;
11:        end if
12:    end for
13:    if $found = false$ then
14:        $S \leftarrow S \cup \{X_i\}$;
15:    end if
16: end while
```

sorted particles in X_{sort}, the **while** loop in Algorithm 2 is executed N times to check if each particle is within the radius r of the seeds in S. If S currently consists of i species seeds, the **for** loop will be performed at best 1 and at worst i times. The former will happen when the particle considered is within r of the first seed. In contrast, the latter will happen if the particle considered falls outside of r of all seeds in S. This shows that the time complexity of the distance evaluations is between $O(N)$ and $O(N^2)$.

Some studies have suggested choosing neighbourhood members based on particle indices without involving real distance calculations (Li, 2010; Wang et al., 2012b). These indices merely represent the ordering of particles in a population list, and they are not necessarily closely related. For example, in (Wang et al., 2012b) a "ring" structure was used where every particle was labelled with permanent index i, and each particle i was connected with its two immediate particles $i-1$ and $i+1$. The strategy is computationally inexpensive because the Euclidean distance calculation is not required and the population indices are fixed throughout the entire run. However, the spatial distance between two connected particles could be very large which would inevitably increase the difficulty for the algorithm to converge and locate the optima effectively (Qu et al., 2013).

We propose to build the relationship between each particle using k-d tree data structure and utilise the structure to search for seeds' nearest neighbours. We name this improved SPSO as KDT-SPSO. In the best-case scenario, the **while** loop in Algorithm 5 is only performed once when all unprocessed particles in X_{sort} are within r of the first seed. In such a case, KDT-SPSO takes $O(N)$ times to report all nodes fall within the search region. In the worst-case scenario, the **while** loop steps through X_{sort} N times when all seeds do not have their radii overlapped. The number of evaluations performed in each of the loops follows a complexity of $O(\sqrt{N} + m)$, where $O(\sqrt{N})$ is the total number of regions intersected by the boundary of the search region in the k-d tree; m is the number of points falling inside the search region, which is equal to 1 if the seed does not have any neighbour. If the k-d tree building procedure is taken into account, KDT-SPSO's complexity would be $O(N \cdot \log^2 N)$ in the best-case scenario and $O(N(\sqrt{N}))$ in the worst-case scenario. It can be seen that k-d tree search outperforms exhaustive nearest neighbour search in the worst-case scenario, particularly if N is sufficiently large, which is typically required to solve multimodal problems. The proposed strategy is not limited to the SPSO used in our experiment, but can also be extended to other evolutionary and swarm algorithms that require distance calculation. The details of k-d tree building and searching procedures are discussed below.

4.3.1 Building k-d tree structure

The procedure of identifying species members can be formulated as nearest neighbour search problem (NNS) to find all points within a given range from a specified query point. However, naive NNS calculates the distance of each query point (seed) for every reference point (particle). Instead of using the brute-force approach, we propose to build a relationship between particles using a k-d tree data structure to enhance the efficiency of NNS. In the process of finding the nearest neighbours to the query points, we traverse down the tree to the regions that contain most of the nearest neighbours. The implementations of the k-d tree in evolutionary algorithms have been reported recently. Nguyen et al. (2015) used a k-d tree structure to divide the search space of GA into explored and unexplored regions so that the algorithm could focus more on searching the unexplored areas. Instead of selecting random parent solutions, Lacerda and Batista (2019) utilised the k-d tree structure in DE to group the parent solutions with similar characteristics under the same subtree, which were then used to generate new candidate solutions to improve convergence. To the best of our knowledge, ours is the first work using a k-d tree in SPSO to address the issue of distance evaluation in identifying species members.

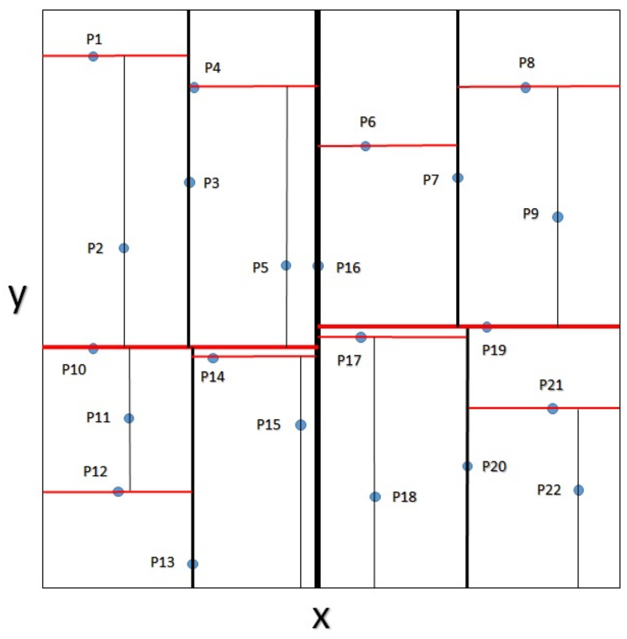
The k-d tree data structure proposed by Bentley (1975) was an extended version of the binary tree for organising data points in a space with k dimensions to assure fast searching. If the points comprise (x, y) dimensions, firstly they are sorted based on values of the first dimension, i.e., x. Then, the median point is selected as a splitting (parent) node that partitions all points into two parts with respect to their coordinates in the x axis. The left subspace holds the points with x value smaller than the parent node while the right subspace holds the points with x value at least as large as the parent node. The splitting procedure is called recursively in the left and right subspaces to create another two new subspaces along alternate dimensions. For example, at the first level, x dimension is utilised to split the whole space into two subspaces, and at the second level, y dimension is utilised to split each of the two subspaces into another two subspaces and so on. The splitting process continues until all points are subdivided. We illustrate the splitting process in Figure 4.1 for a clearer understanding. The steps of building a k-d tree are given in Algorithm 3. The accumulated cost of sorting and recursively building two child nodes amounting to $O(N \cdot \log^2 N)$, which is also widely reported in the literature (Wald and Havran, 2006).

Algorithm 3 BuildKdTree (X, *depth*)

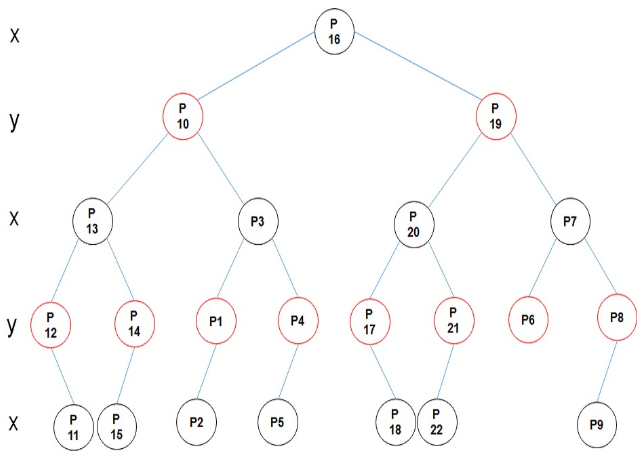
```
1: **Input:** A set of points $X$ and current depth *depth* in the tree
2: **Output:** The nodes of the $k$-d tree storing $X$
3: **if** $X \leq 0$ **then**
4:     **return** None;
5: **end if**
6: axis = *depth* mod $k$;
7: Sort and select median by axis from $X$;
8: $node.location \leftarrow median$;
9: $node.leftChild \leftarrow BuildKdTree$(sorted points in $X$ before median, $depth$+1);
10: $node.rightChild \leftarrow BuildKdTree$(sorted points in $X$ after median, $depth$+1) ;
11: **return** node;
```

4.3.2 Nearest neighbour search using a k-d tree

After building the tree structure, the nearest neighbour search of a query point can be done efficiently by selectively traversing down the tree. To find all points within a radial range r of the query seed, the algorithm starts at the root node and recursively searches both left and right child nodes. If the query circle is fully contained in one of the subspaces, there is no need to explore the other subspace. We illustrate an example of how the k-d tree is implemented to locate all nearest neighbours within a radial

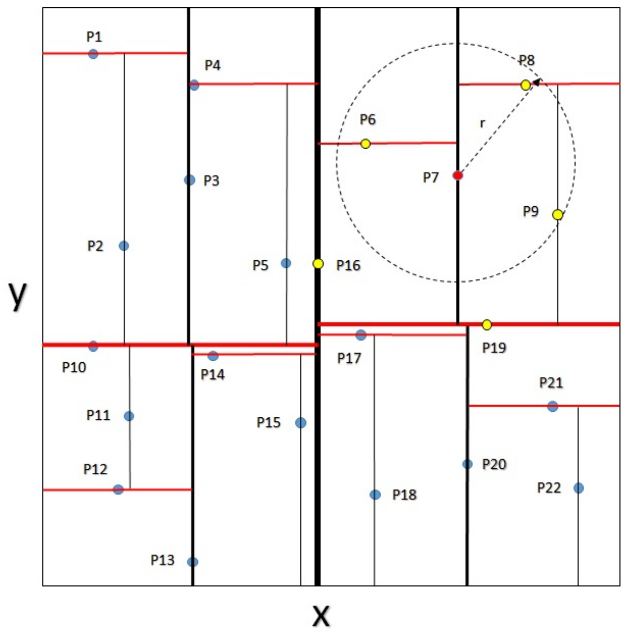


(a)

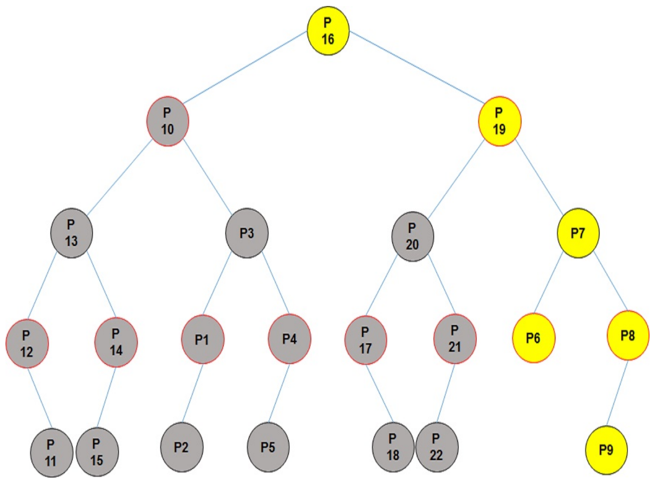


(b)

Figure 4.1: The chart on top (a) and the tree graph at the bottom (b) show how a two-dimensional space is structured using the k-d tree. The blue dots in the chart represent data points and also correspond to the tree nodes subdividing the space into two regions. The black vertical lines correspond to the splits in the x-axis while the red horizontal lines correspond to the splits in the y-axis. The space is split along a different dimension at each level of the tree. The points with lower values than the split node in the corresponding axis are partitioned into the left/bottom subspace, while the points with higher values will be partitioned into the right/upper subspace. The deeper the level of the splits, the thinner the line. The splitting procedure is repeated until each data point is transformed into a node.



(a)



(b)

Figure 4.2: The chart on top (a) and the tree graph at the bottom(b) show how nearest neighbour search is performed using the k-d tree. The blue dots in the chart represent data points, which also correspond to tree nodes subdividing the space into two regions. The black vertical lines correspond to splits in the x-axis while the red horizontal lines correspond to splits in the y-axis. To find the neighbours of P7 (red dot), the algorithm only needs to evaluate P16, P19, P7, P6, P8, and P9 (yellow dots). The yellow nodes in the tree graph indicate visited nodes while the grey nodes indicate eliminated nodes that are not visited.

r of point P7 in Figure 4.2. The search begins from the root node (P16). The distance between the node and the query point is computed. Since the distance is out of the query range, it is not added to the neighbourhood list. The algorithm continues examining the next two child nodes. If the search range is fully contained in only one of the subspaces, the search on the other side can be eliminated. If it crosses the other side, both subspaces have to be visited. As the splitting plane is axis-aligned, we can easily check whether the query circle crosses the splitting plane by calculating the absolute distance between the query point's x/y coordinate and split node's axis coordinate.

Since the algorithm begins at level 1, the discriminator at this level is the x-axis. It can be observed that the absolute distance is greater than r, it means that the nearest neighbours are all in the right side, and the left subspace can be pruned. We continue to look up points in the right subspace recursively. If the query circle crosses both half-spaces, the algorithm needs to unwind the recursion and walks back up to the current parent node after reaching a tree node to check the other side of the subspace. It can be observed that the algorithm only needs to visit 6 points out of 22 points to obtain the nearest neighbours which greatly reduces the computation cost as compared to naive search.

In general, the k-d tree radius search procedure has the complexity of $O(\sqrt{N} + m)$, where m is the number of a point falling inside the search region. The pseudocode of the k-d tree search is shown in Algorithm 4 and the steps of KDT-SPSO are summarised in Algorithm 5. Since the complexity of KDT-SPSO and SPSO may change in each iteration, we also included the actual computation time as one of the evaluation measures to gauge the real performance of both algorithms in several benchmark tests, and they are discussed in the next section.

Algorithm 4 SearchNN (*root*, *q*, *r*, *result*, *depth*)

```
1: $result = \emptyset$;
2: Input: The root/a subtree of a k-d tree , query point, radius, results
   and current depth
3: Output: Result of all nearest points within $r$ of $q$
4: $CurrNode = root$;
5: $distance \leftarrow ComputeDistance(CurrNode, q)$ ;
6: if $distance \leq r$ then
7:    $result \leftarrow result \cup \{CurrNode\}$;
8: end if
9: axis= $depth$ mod $k$;
10: if $CurrNode[axis] > (q[axis] - r)$ then
11:    if $CurrNode.leftChild$ is not Null then
12:       $SearchNN(Curr\_node.leftChild, q, r, result, depth + 1)$;
13:    end if
14: end if
15: if $CurrNode[axis] < (q[axis] + r)$ then
16:    if $CurrNode.rightChild$ is not Null then
17:       $SearchNN(CurrNode.rightchild, q, r, result, depth + 1)$;
18:    end if
19: end if
20: return $result$
```

Algorithm 5 Algorithm for determining species seeds and members using KDT-SPSO

```
1: Build k-d tree structure for all particles $X$ using Algorithm 3;
2: Sort all $X$ in decreasing fitness values, obtaining $X_{sort}$;
3: $S = \emptyset$;
4: while (there are unprocessed particle in $X_{sort}$) do
5:    Search for the best unprocessed $X_i \in X_{sort}$;
6:    $S \leftarrow S \cup \{X_i\}$;
7:    Search all $X$ within $r$ of $X_i$ using Algorithm 4;
8:    Mark all found $X$ as processed and assign $X_i$ as $lbest$ for $X$;
9: end while
```

4.4 Benchmark functions and experiment settings

Before applying KDT-SPSO to solve palm localisation problems, the proposed method was tested on six commonly used benchmark multimodal functions from the Congress on Evolutionary Computation (CEC)'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization test suite (Li et al., 2013) for performance evaluations. The performance of KDT-SPSO was also compared against another two algorithms, in particular r3PSO (Li, 2010) and SPSO (Parrott and Xiaodong Li, 2006). r3PSO is a ring topology based PSO, and each member interacts with its immediate member on its left and right. Both are representatives of niching PSO algorithm that use population indices and Euclidean distance measurements respectively to form their neighbourhood.

The characteristics of the six functions are presented in Table 4.1 and their optima are known in advance. The details of parameter settings are summarised in Table 4.2. Parameter r is not needed in r3PSO. The other parameters in the experiments were fixed as follows: The cognitive and social constants $c1$ and $c2$ were both set to 2 as proposed by Kennedy and Eberhart (1995) to average influences of both components to the overall performance. The initial weight γ was set to be the same as in the study of Shi and Eberhart (1999): 0.9 and linearly reduced to 0.4. Since there is no restarting mechanism in r3PSO, the mechanism was disabled in both KDT-SPSO and SPSO for fair comparisons. All test functions are considered as maximisation functions. The codes for r3PSO and SPSO were reproduced referring to the pseudocodes presented in their respective studies. Each benchmark function was run independently 50 times. In each run, the parameter settings and particle positions were randomly reinitialised.

Table 4.1: Test functions.

Function name	Function	Dimensions	Range	Number of global optima
F1: Five-uneven-peak trap	$F1(x) = \begin{cases} 80(2.5-x) & \text{for } 0 \leq x < 2.5, \\ 64(x-2.5) & \text{for } 2.5 \leq x < 5.0, \\ 64(7.5-x) & \text{for } 5.0 \leq x < 7.5, \\ 28(x-7.5) & \text{for } 7.5 \leq x < 12.5, \\ 28(17.5-x) & \text{for } 12.5 \leq x < 17.5, \\ 32(x-17.5) & \text{for } 17.5 \leq x < 22.5, \\ 32(27.5-x) & \text{for } 22.5 \leq x < 27.5, \\ 80(x-27.5) & \text{for } 27.5 \leq x \leq 30 \end{cases}$	1	$x \in [0, 30]$	2
F2: Equal Maxima	$F2(x) = \sin^6(5\pi x)$	1	$x \in [0, 1]$	5
F3: Uneven Decreasing Maxima	$F3(x) = \exp\left(-2\log(2) \times \left(\frac{x-0.08}{0.854}\right)^2\right) \times \sin^6(5\pi(x^{3/4} - 0.05))$	1	$x \in [0, 1]$	1
F4: Himmelblau's function	$F4(x, y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2$	2	$x, y \in [-6, 6]$	4
F5: Six-Hump Camel Back	$F5(x, y) = -4[(4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (4y^2 - 4)y^2]$	2	$x \in [-1.9, 1.9]$ $y \in [-1.1, 1.1]$	2
F6: Shubert	$F6(\vec{x}) = -\prod_{i=1}^{2} \sum_{j=1}^{5} (j \cos[(j+1)x_i + j]$	2	$x_1, x_2 \in [-10, 10]$	18

Table 4.2: Parameters and criteria for test functions.

Function	r	Maximum no. of evaluation	Population size	Accuracy level ϵ
F1	0.01	20000	100	1.00E-04
F2	0.01	20000	100	1.00E-04
F3	0.01	20000	100	1.00E-04
F4	0.05	20000	100	1.00E-04
F5	0.05	20000	100	1.00E-04
F6	0.50	200000	500	1.00E-04

4.4.1 Performance Evaluations

As the global optima of all test functions were known *a priori*, the following four measurements were used to evaluate the performance of the algorithms:

1. **Success Rate (SR):** SR measures the percentage of successful runs (all known global optima are successfully found) out of 50 independent runs. The algorithm is allowed to run for a maximum number of evaluations (MNE) before termination. If all known optima are found within a run, it is coded as "1", otherwise "0". Since our goal is to find all global optima, local optima are not considered in the measurement.

2. **Peak Ratio (PR):** PR measures the average number of known global optima found over the total number of known global optima in a run.

3. **Average number of evaluations (ANE):** ANE measures the average number of evaluations required to identify all known global optima, reflecting the convergence speed. If the algorithm cannot locate all global optima in the run, then the ANE equals the predefined MNE.

4. **Computation time (CT):** CT is the average computation cost in

second (s) for an algorithm to perform all required evaluations over a run.

A global optimum is found if the fitness value of a species seed is within the accuracy level ϵ, and within the radius r from the expected solution. The results were recorded for each run and the experimental results across the total 50 runs were averaged and reported.

4.4.2 Experiment results and analysis

Statistical analyses were performed to determine if there were statistically significant differences between the three algorithms in ANE, CT, SR and PR. Since the results were all left-skewed, non parametric Kruskal-Wallis statistical test that does not assume normality was selected to compare the algorithms. Pearson's chi-square test was performed on SR results, which were recorded in binary format. The results are shown in Tables 4.3 to 4.6, where symbols $+$, $-$ and $\approx$ indicate the competitor is better than, worse than and similar to KDT-SPSO, respectively, based on the Kruskal-Wallis method with Dunn post-hoc pair-wise test or Pearson Chi-Square test (for SR results) at $\alpha = 0.05$.

The results in Table 4.3 show that in general, both SPSO and KDT-SPSO required a similar number of evaluations while r3PSO required the highest number to identify all the optima within the required accuracy. r3PSO performed poorly because two connected particles based on population indices may be distant from each other in actual distance and require more evaluations to converge. It had reached the MNE before all the optima were found in 32% and 60% of the runs in F4 and F6, respectively. Both SPSO and KDT-SPSO performed equally well because their main structure are similar except the mechanism of nearest neighbour search is improved in KDT-SPSO.

Table 4.3: Comparison of average number of evaluations (ANE).

Function	r3PSO	SPSO	KDT-SPSO
F1	224 -	**200** $\approx$	206
F2	3556 $\approx$	**3134** $\approx$	3336
F3	1310 $\approx$	**1042** $\approx$	1386
F4	14178 $-$	**11308** $\approx$	11768
F5	6718 $\approx$	**5774** $\approx$	6400
F6	168800 $-$	145710 $\approx$	**143320**

Symbols $+$, $-$ and $\approx$ indicate that the competitor is statistically better than, worse than, and similar to KDT-SPSO by Kruskal-Wallis and post-hoc Dunn tests at $\alpha = 0.05$.

ANE is commonly used to reflect the convergence speed, but do not necessarily translate into actual computation runtime. This is because the latter depends on many factors such as software or hardware used, coding style, programming language, and complexity of the algorithm. However, some users may be interesting in knowing under the same environment, which algorithms use fewer computer resources, and whether the algorithms that require more computing time lead to better results. The CT results in Table 4.4 show that the r3PSO was the fastest in all evaluations. This is because the neighbourhood of particles in r3PSO is predefined and remains the same throughout the entire run. Moreover, the algorithm does not require Euclidean distance calculation. On the other hand, compared to KDT-SPSO, SPSO was faster in F1 and F2, similar in F3, but gradually became slower from F4 onwards. Since F1 to F3 are relatively simpler functions, so they converge more quickly than the other functions. When a function starts converging, the number of seeds reduces and stabilises towards the end of a run. This results in the gradual reduction in SPSO overhead because of the execution of **for** loop in Algorithm 2 reduces. In addition, as KDT-SPSO requires a construction phase in each iteration, the construction overhead can be significant for simple functions and small dataset. Due to these reasons, SPSO is more efficient than KDT-SPSO in

F1 to F3.

Another factor that affects the computation speed of both algorithms is the size of the dataset whereby KDT-SPSO performed significantly better than SPSO in F6 with a large population size. To show the advantage of KDT-SPSO over SPSO, a separate test with the population size varied from 50 to 1200 was performed on F4. The results are presented in Figure 4.3, and they clearly show that KDT-SPSO significantly outperformed SPSO when the population size increased. This is because the distance evaluation in SPSO increases exponentially with population size, which is the common issue encountered in traditional niching PSO algorithms. The k-d tree search in KDT-SPSO helped to prune most of the data points to be visited. Also, in a large dataset, the cost of building k-d tree structure becomes negligible. Since the palm tree detection problem is much more complex than F6 with higher dimension (3D) and more unevenly spaced optima, incorporating k-d tree search algorithm in traditional niching PSO algorithms is beneficial to improve the NNS efficiency.

Table 4.4: Comparison of computation time (CT) in seconds.

Function	r3PSO	SPSO	KDT-SPSO
F1	**1.19**+	1.38 +	1.59
F2	**1.44**+	1.63 +	1.71
F3	**1.42**+	1.98 ≈	1.97
F4	**3.64**+	6.30 -	5.00
F5	**1.38**+	2.31 -	1.74
F6	**32.40**+	100.94 -	68.34

Symbols +, $-$ and $\approx$ indicate that the competitor is statistically better than, worse than, and similar to KDT-SPSO by Kruskal-Wallis and post-hoc Dunn tests at $\alpha = 0.05$.

Interestingly, shorter computation time does not necessarily equate to lower accuracy. It can be observed from the SR results presented in Table 4.5 that despite KDT-SPSO being faster than SPSO, both performed equally well and KDT-SPSO was slightly better than SPSO in F4. It shows that the recall capability of both algorithms is high. On the other hand, although

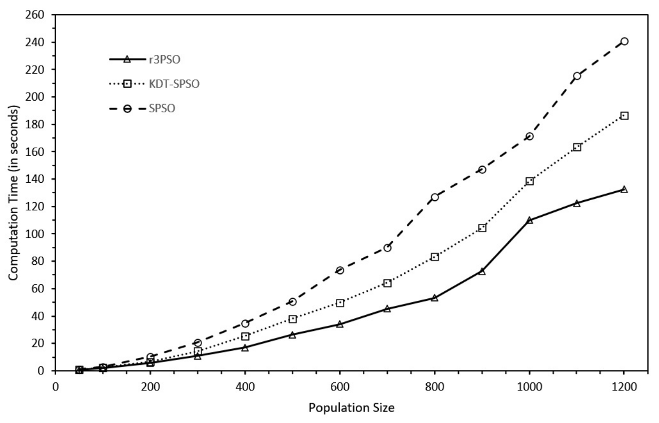


Figure 4.3: Comparison of computation time of three different algorithms.

the speed of r3PSO was the fastest, its SR was generally the lowest. It performed significantly worse than KDT-SPSO in F4 and F6. In F6, it successfully located all optima only in 40% of the runs compared to 74% achieved by SPSO and KDT-SPSO. We noticed that in these two functions, although some of the particles were already close to the nearest global optimum, they remained stagnant even before the MNE was reached. This happens because when population indices are used, two neighbouring particles may be located in different niches and they oscillate between the two niches. Due to this, particles' fitness values always fluctuate and are unable to reach an equilibrium state. As a result, the particles fail to locate the known optima within the acceptable accuracy level ϵ.

Since F6 is the most complex function among the others, none of the algorithms was able to identify all optima in all 50 runs. The results in Table 4.6 further elaborate the percentage of global optima found by each algorithm in each function. r3PSO recorded the lowest PR in F4 and F6, which were only 0.945 and 0.961, respectively. The poor performance of r3PSO was also widely reported in literature (Qu et al., 2013; Liu et al., 2020), implying that although r3PSO is fast, it is not suitable for complex functions requiring a high recall rate. Nonetheless, its overall performance in terms of speed and recall rate was still the best in relatively simple functions, for example, F1, F2, F3, and F5. In general, all three algorithms evaluated were not able to identify every optimum in F6 suggesting that larger population size is probably needed but will be commensurate with higher overhead. The other option is to integrate restarting mechanism in the algorithms to reposition selected particles to other parts of search spaces.

Table 4.5: Comparison of success rate (SR).

Function	r3PSO	SPSO	KDT-SPSO
F1	**1** $\approx$	**1** $\approx$	**1**
F2	**1** $\approx$	**1** $\approx$	**1**
F3	**1** $\approx$	**1** $\approx$	**1**
F4	0.78 $-$	0.98 $\approx$	**1**
F5	**1** $\approx$	**1** $\approx$	**1**
F6	0.4 $-$	**0.74** $\approx$	**0.74**

Symbols $+$, $-$ and $\approx$ indicate that the competitor is better than, worse than, and similar to KDT-SPSO according to Pearson's chi-square test at $\alpha = 0.05$.

Table 4.6: Comparison of peak ratio (PR).

Function	r3PSO	SPSO	KDT-SPSO
F1	**1** $\approx$	**1** $\approx$	**1**
F2	**1** $\approx$	**1** $\approx$	**1**
F3	**1** $\approx$	**1** $\approx$	**1**
F4	0.945 $-$	0.995 $\approx$	**1**
F5	**1** $\approx$	**1** $\approx$	**1**
F6	0.961 $-$	**0.986** $\approx$	0.984

Symbols $+$, $-$ and $\approx$ indicate that the competitor is better than, worse than, and similar to KDT-SPSO according to Kruskal-Wallis and post-hoc Dunn tests at $\alpha = 0.05$.

4.5 Summary

In this chapter, we discussed the implementation of a special type of binary search tree, called k-d tree, to speed up the multimodal PSO (SPSO) NNS during optimisation process. The use of k-d tree search involves two stages: the tree construction stage and tree searching stage. The tree construction stage aims to subdivide all points into a binary tree structure comprising of multiple nodes representing the data's subspaces. During the NNS process, the algorithm only needs to evaluate the points in the subspaces within the predefined distance from the query point, eliminating the need to evaluate all points exhaustively. A detailed comparison of six of the CEC 2013 multimodal benchmark functions between the proposed method and other techniques, namely r3SPSO and SPSO, shows that the proposed method was significantly faster than SPSO and successfully discovered more optima than r3SPSO. The proposed mechanisms can also be applied in other metaheuristics algorithms that require distance evaluation. The benefits of using the proposed method is more apparent in solving complex problems, e.g. palm tree detection, that require a large population size than the simple ones.

Chapter 5

Application of KDT-SPSO for Palm Tree Detection

5.1 Introduction

The previous chapter explains the details of KDT-SPSO and its performance in solving selected benchmark functions. This chapter describes the utilisation of KDT-SPSO for palm tree detection in UAV images. Figure 5.1 and Figure 5.2 present a "big picture" overview of how KDT-SPSO is incorporated into the palm tree detection framework. Our approach consists of three major steps. First, an input image randomly generates a large set of particles representing plausible palm tree candidates. Each particle contains a potential solution, i.e. x and y coordinates of the pixel centres and size of the tree. Then, fitness/classification score representing the detection's quality is computed for each candidate. This forms the objective function for KDT-SPSO, which is then optimised aiming to find the best sets of locations and scales that maximise the particles' fitness score. The higher the score, the more likely the subregion contains a palm tree. These processes are repeated in each iteration until the maximum number of evaluations is reached. At each of the candidate coordinates, a square image patch with the width and height of its proposed size is extracted and fed into the feature-extraction-based-classifier selected in Chapter 3 to produce a classification score.

5.2 Methods

5.2.1 Feature extraction and classification

The feature-extraction-based-classifier (LBP(SVMRBF)-R) selected in Section 3.4 was applied to produce the fitness score for the candidates. SVM normally gives categorical output for classification purposes, but continuous value is required in KDT-SPSO for meaningful comparisons among candidates. Therefore, the output of $f(x)$ was transformed into posterior probabilities based on the following sigmoid function as suggested in Platt (1999):

$$P(y = 1) \mid f(x)) = \frac{1}{1 + \exp(Af(x) + B)} \tag{5.1}$$

where parameters A and B were found by maximum likelihood estimation from the same training data used to fit the SVM. The posterior probabilities in the range of [0,1] were turned into fitness values used in KDT-SPSO and the particle with fitness value of more than 0.5 corresponds to a palm tree.

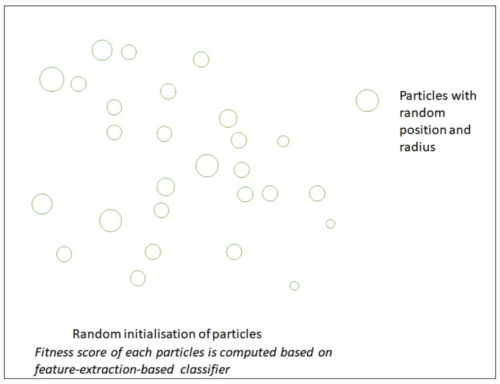
5.2.2 Reinitialisation of non-participating particles

As observed in the results of F4 and F6 in Section 4.4.2, some of the optima were not detected because of the imbalanced number of particles assigned to each species. In some cases, there may be a lack of particles exploring other potential regions if too many particles track the same peak, resulting in loss of diversity, leading to premature convergence. They are likely to move around and move closer to the same *lbest* and not able to generate new movements beyond the search region. One of the ways to address the issue is by increasing the number of particles, but it is impractical when the search space and amount of optima to be identified are large. Parrott and Xiaodong Li (2006) suggested integrating a reinitialisation mechanism into SPSO and triggering it when the number of particles inside a species exceeds a maximum population threshold (POP_{max}). This mechanism allows particles that have already converged to be reinitialised in other search regions.

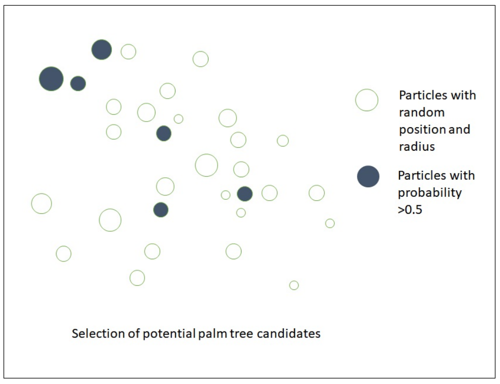
In the present study, we modified the original mechanism. The particles that were further away from the seed and caused the species to exceed the population capacity were reinitialised and relocated to a random position in each iteration. The POP_{max} was also set to limit the number of particles inside each species where only the closest POP_{max} particles remained as the members of a species and other particles that would cause the species population to exceed the threshold would be reinitialised. The mechanism was also extended to select the particles that were neither seed nor species members with fitness value $<= 0.5$ as they did not represent palm trees. We called these selected particles as non-participating particles (NPP) because they did not contribute to the improvement of convergence. By reinitialising them with new positions and velocities, the number of evaluations executed in the existing regions could be cut down and the selected particles could help to explore other areas. Nevertheless, the selected particles' current *pbest* were not reinitialised so that the new positions could only be accepted if they were fitter than *pbest*. This approach maintained a balance between exploration and exploitation. The rest of the particles would keep exploiting their respective species regions to look for better solutions. The summary of KDT-SPSO's procedures for palm tree detection is explained in Algorithm 6, the pictorial illustration is shown in Figure 5.1, and the flowchart is presented in Figure 5.2.

Algorithm 6 Algorithm of KDT-SPSO for palm tree detection

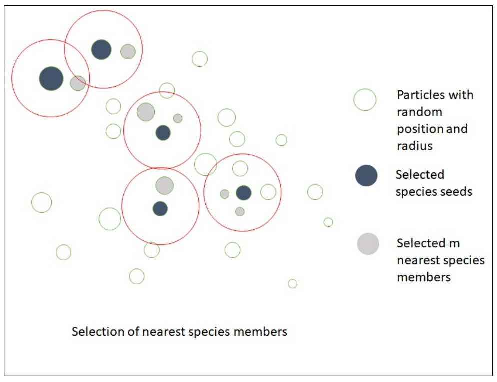
```
1: Randomly initialise $X_i(t)$ for all $i$ in image space, where $t = 0$;
2: Extract image patch based on $X_i(t)$'s coordinates and size;
3: Compute image patch's LBP features;
4: Evaluate $f(X_i(t))$'s fitness/probability value using trained SVM;
5: Assign $P_i = X_i(t)$;
6: while termination criteria is not met do
7:     $r = r_{max} \times t/t_{max}$;
8:     Select particles with probability $> 0.5$, obtaining $X_{palm}$
9:     Determine species seeds and species members in $X_{palm}$ following Algorithm 5;
10:    Update velocity and position of $X_i(t)$ following Equation 4.1 and Equation 4.2;
11:    Select NPP and reinitialise their position and velocity;
12:    Extract image patches based on $X_i(t)$'s updated coordinates and size;
13:    Repeat Step 3 and Step 4;
14:    if $f(X_i(t)) > f(P_i)$ then
15:        Assign $P_i = X_i(t)$;
16:    end if
17: end while
```


Particles with random position and radius
Random initialisation of particles
Fitness score of each particles is computed based on feature-extraction-based classifier

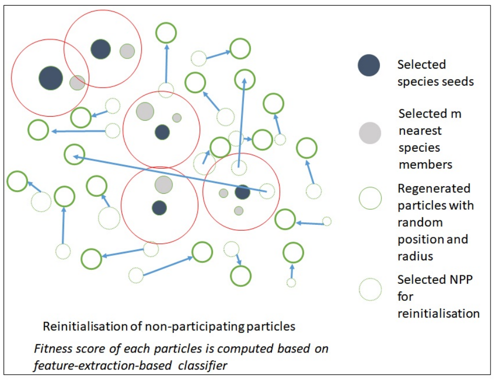
(a) Step 1


Particles with random position and radius
Particles with probability >0.5
Selection of potential palm tree candidates

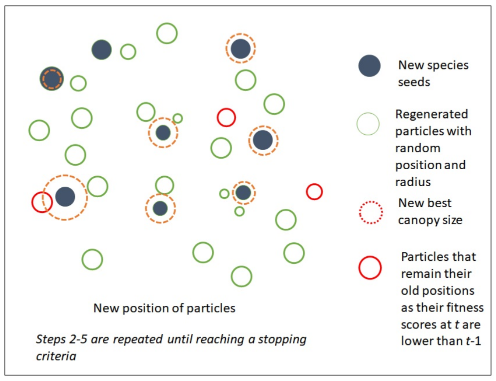
(b) Step 2


Particles with random position and radius
Selected species seeds
Selected m nearest species members
Selection of nearest species members

(c) Step 3


Selected species seeds
Selected m nearest species members
Regenerated particles with random position and radius
Selected NPP for reinitialisation
Reinitialisation of non-participating particles
Fitness score of each particles is computed based on feature-extraction-based classifier

(d) Step 4


New species seeds
Regenerated particles with random position and radius
New best canopy size
Particles that remain their old positions as their fitness scores at t are lower than t-1
New position of particles
Steps 2-5 are repeated until reaching a stopping criteria

(e) Step 5

Figure 5.1: Pictorial description of the KDT-SPSO algorithm for palm tree detection.

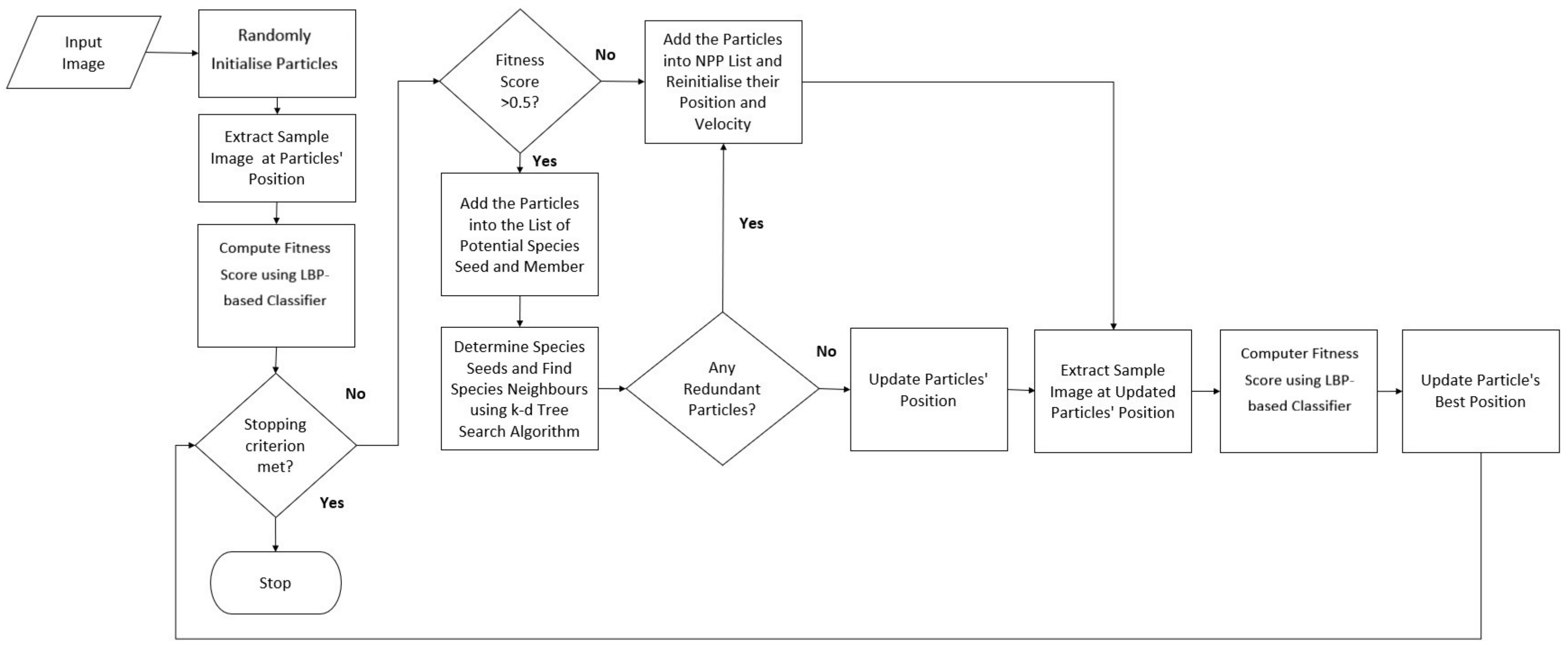

Input Image
Randomly Initialise Particles
Extract Sample Image at Particles' Position
Compute Fitness Score using LBP-based Classifier
Stopping criterion met?
No
Yes
Stop
Fitness Score >0.5?
No
Yes
Add the Particles into the List of Potential Species Seed and Member
Determine Species Seeds and Find Species Neighbours using k-d Tree Search Algorithm
Add the Particles into NPP List and Reinitialise their Position and Velocity
Yes
Any Redundant Particles?
No
Update Particles' Position
Extract Sample Image at Updated Particles' Position
Computer Fitness Score using LBP-based Classifier
Update Particle's Best Position

Figure 5.2: Flowchart of the proposed KDT-SPSO algorithm for palm tree detection.

5.3 Optimising KDT-SPSO's parameters for palm tree detection

In this section, the performances of the proposed method for palm tree detection are presented. It is well known that the parameters used in metaheuristics are problem dependent, the same set of parameters used for specific problems may not be suitable for another set of problems. To make the performance of the proposed KDT-SPSO more robust for palm tree detection, an empirical analysis was performed on a 400×400 pixels sample image of an oil palm field (a subset of Image 1) to find an appropriate parameter settings that produced optimal results. The effect of three levels of population size (POP_{size}) and maximum population threshold (POP_{max}) on the detection performances, namely SR, PR and ANE, were evaluated. Since the captured images were already georectified to the correct scale, the maximum number of palm trees in a given image could be estimated. The POP_{size} used in this study represents the multiplier of the expected number of palm trees that exist in an image, i.e. POP_{size} of 6 means *6 × expected number of palm trees.*

Other parameters were fixed: The maximum species radius r_{max} was set to 35 pixels, equivalent to 7 m on the ground and smaller than the average planting distance between palm trees of about 9 m. Referring to Brits et al. (2002), the species radius r started with a small value and gradually increased to r_{max}. Through this approach, more species could be formed and discovered at the beginning of a run, and they were slowly absorbed into fitter species at the end of the run. The cognitive and social coefficients $c1$ and $c2$ were both set to 2 as suggested in Kennedy and Eberhart (1995) to average the influences of both components to the overall performance. The initial weight γ was set to 0.9 and linearly reduced to 0.4 following

Shi and Eberhart (1999). The velocity V (Equation 4.1) was constricted within the range $[V_{min}, V_{max}]$, which was $[-15, 15]$ for both position and tree size to prevent particles from moving too far away from the existing search region. A particle was reinitialised to a new position if its updated position exceeded the image domain.

5.3.1 Results and discussion of parameter optimisation experiments

The results of the empirical study summarised in Table 5.1 clearly show that the combination of $POP_{size} = 10$ and $POP_{max} = 2$ achieved the highest SR and PR, where all palm trees were successfully identified in each of the 50 independent runs. It used the second least number of evaluations after $POP_{size} = 6$ and $POP_{max} = 2$ pair to complete the search. The poorest performer was $POP_{size} = 6$ and $POP_{max} = 4$ pair where it used the most number of evaluations to complete the search, yet achieving the lowest SR and PR. The mean effect of POP_{size} and POP_{max} on SR and ANE is demonstrated in Figure 5.3. Figure 5.3a shows that SR was positively correlated to population size. A larger population size provides better search capability to detect all palm trees. Nevertheless, ANE was not influenced by population size, as shown in Figure 5.3b. This implies that more iterations are still required to identify all palm trees if low population size is used.

To illustrate the parameter selection of KDT-SPSO, four sets of graphs were plotted (Figure 5.3) to observe the influence of POP_{size} and POP_{max} on the SR and ANE. Figure 5.3c shows that higher POP_{max} threshold has an adverse effect on SR. The recall capability of KDT-SPSO was highest when only 2 particles were allowed to stay in the same species. This is because when a higher threshold value is used, more particles are trapped in the same species and unable to identify more palm trees outside the species. The result also shows that it is necessary to trigger reinitialisation in KDT-SPSO. On the other hand, lower POP_{max} also improved convergence speed, as shown in Figure 5.3d. This happens because when more particles are reinitialised, the chances of finding new optima are higher leading to faster convergence.

Table 5.1: Experimental results with respect to the performance of KDT-SPSO with different levels of POP_{size} and POP_{max} on the test image based on 50 independent runs.

POP_{size}	POP_{max}	SR (%)	PR	ANE
6	2	88.00	0.994	**1339.20**
6	4	60.00	0.976	1949.76
6	6	62.00	0.975	1920.96
8	2	98.00	0.999	1512.96
8	4	92.00	0.996	1893.12
8	6	90.00	0.995	1578.24
10	2	**100.00**	**1**	1473.60
10	4	94.00	0.997	1843.20
10	6	90.00	0.995	1852.80

5.4 Evaluation of KDT-SPSO's performance on test images

After obtaining the optima POP_{size} and POP_{max} settings from the previous section, the KDT-SPSO was performed on 25 sets of new images as shown in Figure 5.6. The size of each test image was 500 $\times$ 500 pixels containing 5-33 palm trees. The algorithm was compared with the perfor-

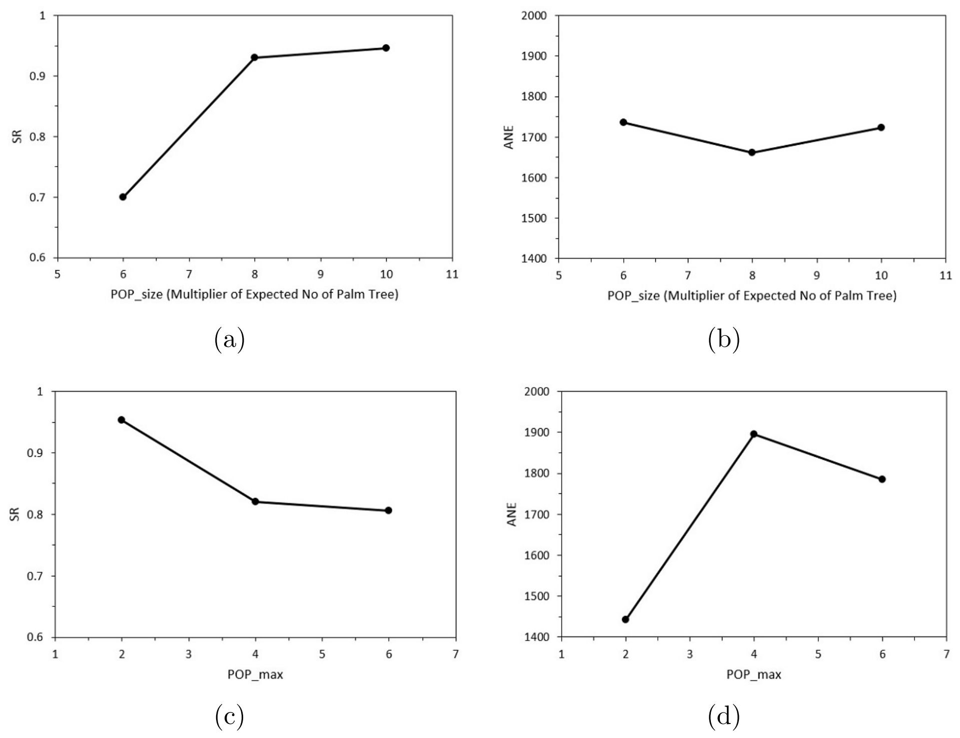


Figure 5.3: Effect of POP_{size} and POP_{max} on SR and ANE.

mance of exhaustive sliding window approach and subsampling approach as explained in Section 2.2.2. The description of the methods compared in the experiment is presented in Table 5.2.

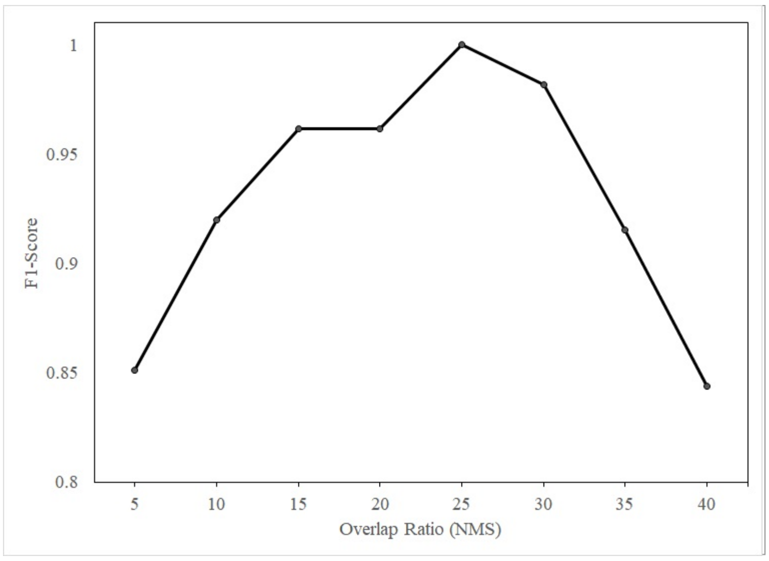
5.4.1 Exhaustive sliding window approach

To detect palm trees of various sizes using sliding window technique, the original image was firstly downscaled to 6 different sizes at the scale of 0.9^x to the original image following the image pyramid approach. Then, each of the resized images (7 including the original image) was traversed by a 60 $\times$ 60 pixels square window to extract image patches as object proposals for palm tree classification. Since the same location might be repeatedly detected in different image scales, non-maximal suppression (NMS) is applied to select the proposals with the highest SVM fitness value. The final selection of NMS is dependent on the object's fitness score and overlap threshold that we have defined. That is, to find the set of proposals that have the largest scores but simultaneously do not exceed the overlap threshold:

$$IOU(r_i, r_j) = \frac{Area(r_i \cap r_j)}{Area(r_i \cup r_j)} \tag{5.2}$$

where $IOU(r_i, r_j)$ refers to the ratio of intersection area of object proposals r_i and r_j to the total area covered by r_i and r_j. NMS starts with a list of object proposals with scores sorted from the highest to the lowest. In the beginning, the first proposal in the list is selected, and the overlap ratio between itself and the remaining proposals is computed as in Equation 5.2. If the overlap ratio $IOU(r_i, r_j)$ exceeds the predefined threshold, the proposal with the lower score will be eliminated from the list. After all comparisons are made, the selected proposal is moved to the list of final detections. The process is repeated for the remaining proposals in the list. We set the overlap threshold at 0.25 as it was the best value achieving the highest F1-Score in our optimisation test, as shown in Figure 5.4.

Figure 5.4: F1-score results against different levels of NMS's overlap ratio.



5.4.2 Subsampling approach

In contrast to the exhaustive sliding window approach, which systematically samples a large number of object proposals at various scales and locations in the input image, the subsampling approach samples a subset of plausible proposals at random scales and locations. Then, NMS is applied to select a subset for the proposals with the best scores. The object proposal selection problem can also be formulated as a binary optimisation problem as reported by (Rujikietgumjorn and Collins, 2013; Pham et al., 2016), which is to select the best set of proposals that minimises the following quadratic objective function:

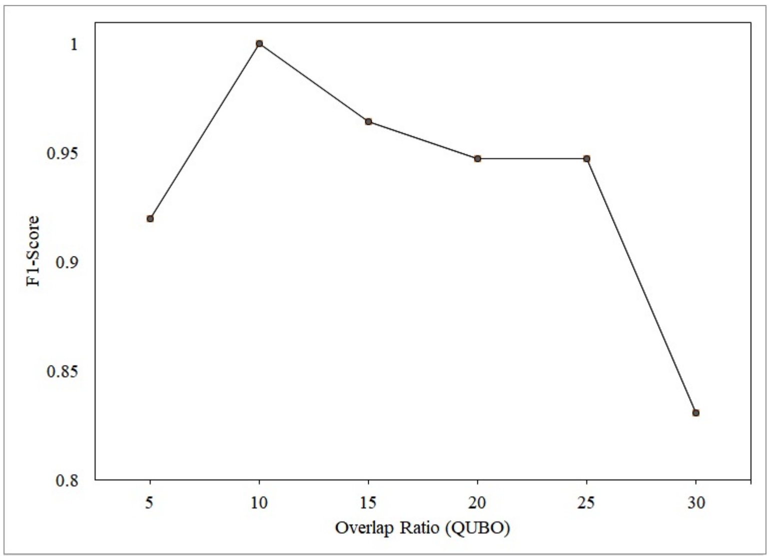
$$f(x) = \sum_{i=1}^{n} c_i x_j + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_i x_j \tag{5.3}$$

where the first term c_i represents the unary energy, which is the classification score of the proposal i. The second term c_{ij} is defined as the pairwise energy, which measures the score when a pair of candidates x_i, x_j overlaps based on the overlap ratio, computed as the intersection area between two overlapping proposals, divided by the area of the smaller proposal. If the overlap ratio is high, a large penalty will be given so that only the fitter one will be chosen. $x_i \in \{0, 1\}$ is the binary variables that need to be solved, an optimisation algorithm is applied to search for the best assignment of 0 and 1 that maximises the objective function value.

As the studies claimed that QUBO could achieve better results than NMS in selecting the best candidates, we included the QUBO approach for comparison. We leveraged the QUBO source code in Matlab provided by Pham et al. (2016) for our study. We set the overlapping threshold at 0.1 as it was the optimum value achieving the highest F1-Score as shown in Figure

5.5, a large penalty (i.e. -100000) was given to the overlapping pairs that exceed the threshold.

Figure 5.5: F1-score results against different levels of QUBO's overlap ratio.



In addition to the overlapping threshold, the initial number of sub-samples to be generated also needs to be defined. If the number of sub-samples is too low, it may miss out a lot of objects. If the number is too high, there will be no significant advantage in terms of computation cost over the sliding window approach. We investigated the effect of two different numbers of initial object proposals, which were 7500 and 3250 on the detection accuracy. 7500 was equivalent to the total number of evaluations spent in KDT-SPSO approach (375 particles $\times$ 20 iterations). This was to compare whether both algorithms could perform similarly if only a limited number of evaluations was allowed. Then, the number of proposals was halved (3250) to evaluate whether a smaller proposal size was sufficient to detect all palm trees in the images. In addition, NMS was also tested to compare its performance against QUBO.

Table 5.2: Description of the methods tested in our palm tree detection experiment.

Method	Description
SW	Sliding window approach and final selection using NMS
S-NMS1	Generating 3250 random object proposals and final selection using NMS
S-QUBO1	Generating 3250 random object proposals and final selection using QUBO
S-NMS2	Generating 7500 random object proposals and final selection using NMS
S-QUBO2	Generating 7500 random object proposals and final selection using QUBO
KDT-SPSO	Our proposed method

5.4.3 Evaluation methods

Precision, recall, F1-score and runtime (s) as described in Section 3.2.5 were used as the evaluation measures. The non-parametric Kruskal-Wallis was also applied to test the results statistically for significant difference at $\alpha = 0.05$.

5.5 Results and discussion

The results of the experiment are shown in Tables 5.3 - 5.6 and the snapshots of the KDT-SPSO optimisation process are presented in Figure 5.7. Results show that the average recall rate of KDT-SPSO was 97.86%, which was slightly lower than that of the sliding window technique, because some of the palms were missed in 15 out of 25 test images. The poorest results are observed in Image 25 mainly because many palm canopies were out of shape. As a result, the classifier was not able to recognise them as palm trees. This situation normally happens when a palm tree is infected with disease or smothered by overgrown creepers plants. The second reason was because the distances between the palm trees in the images were inconsis-

Table 5.3: Comparison of recall rate obtained by different methods on UAV images (%).

Image	Number of palms	Recall rate (%)					
		SW	S-NMS1	S-QUBO1	S-NMS2	S-QUBO2	KDT-SPSO
1	33	100.00	100.00	100.00	100.00	100.00	99.58
2	31	100.00	100.00	96.77	100.00	100.00	99.47
3	27	100.00	92.59	96.30	100.00	100.00	97.26
4	17	100.00	94.12	94.12	100.00	100.00	97.41
5	32	100.00	96.88	96.88	100.00	100.00	99.87
6	25	100.00	76.00	72.00	100.00	100.00	97.36
7	21	90.48	76.19	76.19	80.95	80.95	90.48
8	30	100.00	90.00	90.00	93.33	93.33	99.93
9	21	100.00	85.71	85.71	90.48	90.48	100.00
10	30	100.00	96.67	96.67	96.67	96.67	100.00
11	22	100.00	95.45	90.91	95.45	95.45	100.00
12	31	100.00	93.55	93.55	96.77	96.77	100.00
13	23	91.30	56.52	56.52	82.61	82.61	89.48
14	25	100.00	80.00	80.00	88.00	88.00	97.08
15	26	100.00	84.62	84.62	88.46	88.46	99.92
16	20	95.00	75.00	75.00	75.00	75.00	94.80
17	30	100.00	100.00	100.00	100.00	100.00	99.93
18	18	100.00	94.44	94.44	100.00	100.00	100.00
19	7	100.00	100.00	100.00	100.00	100.00	100.00
20	16	100.00	100.00	100.00	100.00	100.00	100.00
21	21	100.00	100.00	100.00	100.00	100.00	100.00
22	30	100.00	96.67	96.67	100.00	100.00	99.07
23	5	100.00	100.00	100.00	100.00	100.00	100.00
24	27	100.00	100.00	100.00	100.00	100.00	100.00
25	16	75.00	62.50	62.50	75.00	75.00	84.75
	Mean	**98.07**$\approx$	89.88$\approx$	89.55$\approx$	94.51$\approx$	94.51$\approx$	97.86

Symbols +, $-$ and $\approx$ indicate that the competitor is statistically better than, worse than, and similar to KDT-SPSO by Kruskal-Wallis and post-hoc Dunn tests at $\alpha = 0.05$.

tent. Some of them were less than the predefined species radius, thus might have caused two very close species to merge into one. Since the species radius r needs to be known *a priori* in KDT-SPSO or many other SPSO variants, it becomes a limitation when the distances between optima are inconsistent. In such cases, an algorithm that can adaptively change the optimum species radius will be advantageous to eliminate the requirement of predefining r.

Notably, the recall rates of all subsampling approaches were lower than that of the KDT-SPSO and sliding window technique. This shows the weakness of the subsampling approach where the randomly generated object proposals may not be sufficient to encompass all palm trees comprising of various sizes and locations in the input images. Their recall rate deteriorated significantly when the proposal size was reduced to half. S-QUBO2 and S-NMS2, which had the equivalent number of proposals needed to be evaluated as KDT-SPSO, could only yield a lower recall rate than or at best similar to KDT-SPSO's. The main factor is contributed to the updating mechanism in KDT-SPSO, which had helped the algorithm to search in other regions for palm trees, whereas in subsampling approach, the location and sizes of the proposals were fixed since the beginning. Interestingly, in our experiment, QUBO did not show significant improvement in recall rate over NMS approach, which contradicts the results reported in (Rujikietgumjorn and Collins, 2013; Pham et al., 2016). In fact, it was slightly poorer than NMS in the 3250-sample test (89.55% vs 89.88%), and same as the NMS in the 7500-sample test (94.51% vs 94.51%). This shows that the efficiency of classifier is the foremost factor affecting the recall rate rather than the performance of the optimisation/ subset selection methods. NMS can perform as good as the QUBO algorithm without going through the complicated optimisation process.

Despite the lower recall rate, KDT-SPSO attained a higher precision rate (88.56%) than the sliding window technique's (80.00%). Its precision rate was higher than that of the sliding window technique in most of the test images, except Image 2 and Image 24. This shows that the optimisation approach implemented in KDT-SPSO had guided the algorithm in selecting palm tree candidates with fitter scores. As demonstrated in Figure 5.7, the particles (red asterisks) were moving towards the locations with better fitness, which were the palm tree centres. Also, the particles might be trapped in only a few palm tree centres if the restart mechanism was not integrated resulting in high false negative rate. However, both of the algorithms performed poorly in complex oil palm environments with mixture of forest trees and bushes, such as Image 4, Image 20, and Image 23. In Image

Table 5.4: Comparison of precision rate obtained by different methods on UAV images (%).

Image	Number of palms	Precision rate (%)					
		SW	S-NMS1	S-QUBO1	S-NMS2	S-QUBO2	KDT-SPSO
1	33	100.00	94.29	94.29	100.00	100.00	100.00
2	31	100.00	100.00	96.77	96.88	100.00	99.93
3	27	92.86	92.59	92.86	96.43	96.43	98.01
4	17	41.46	84.21	80.00	89.47	89.47	80.84
5	32	94.12	93.94	93.94	94.12	94.12	99.82
6	25	92.59	100.00	100.00	96.15	96.15	96.63
7	21	90.47	94.12	100.00	94.44	94.44	92.22
8	30	90.91	100.00	100.00	87.50	87.50	98.20
9	21	61.76	85.71	90.00	82.61	82.61	79.40
10	30	88.24	100.00	100.00	96.67	96.67	96.33
11	22	75.86	91.30	90.91	80.77	80.77	84.14
12	31	91.18	96.67	90.63	90.91	90.91	97.11
13	23	87.50	92.86	92.86	90.48	90.48	99.18
14	25	88.89	90.91	90.91	95.65	95.65	95.77
15	26	92.86	100.00	100.00	100.00	100.00	99.34
16	20	82.61	93.75	93.75	83.33	83.33	86.29
17	30	96.77	100.00	100.00	100.00	100.00	99.87
18	18	62.07	89.47	89.47	90.00	90.00	78.53
19	7	43.75	77.78	77.78	77.78	77.78	72.43
20	16	43.24	59.26	57.14	51.61	51.61	53.24
21	21	84.00	95.45	95.45	91.30	87.50	88.68
22	30	88.24	90.63	90.63	93.75	90.91	94.08
23	5	20.00	55.56	55.56	33.33	33.33	30.17
24	27	100.00	100.00	100.00	100.00	96.43	97.45
25	16	92.31	100.00	90.91	100.00	100.00	96.35
	Mean	80.07$\approx$	**91.14**$\approx$	90.55$\approx$	88.53$\approx$	88.24$\approx$	88.56$\approx$

Symbols $+$, $-$ and $\approx$ indicate that the competitor is statistically better than, worse than, and similar to KDT-SPSO by Kruskal-Wallis and post-hoc Dunn tests at $\alpha = 0.05$.

23, about 80% of the objects were falsely detected as palm trees by sliding window technique resulting in the lowest F1-Score of 33.33% compared to 46.18% achieved by KDT-SPSO. This shows that greedy search adversely affects the detection accuracy because it increases the chances of misclassification of non-palm tree objects. In contrast to recall rate, S-NMS1 and S-QUBO1 approaches based on 3250 samples produced the top two precision rates among all tested approaches. This shows that introducing unnecessary object proposals will increase the chance of false detections.

Subsequently, the F1-Score results that averaged both recall and precision rates show that KDT-SPSO was the best while the sliding window approach was the worst performing algorithm. Again, the performances of

Table 5.5: Comparison of F1-Score obtained by different methods on UAV images (%).

Image	Number of palms	F1-Score (%) (%)					
		SW	S-NMS1	S-QUBO1	S-NMS2	S-QUBO2	KDT-SPSO
1	33	100.00	97.06	97.06	100.00	100.00	99.78
2	31	100.00	100.00	96.77	98.41	100.00	99.69
3	27	94.54	92.59	94.55	98.18	98.18	97.61
4	17	58.62	88.89	86.49	94.44	94.44	88.31
5	32	96.97	95.38	95.38	96.97	96.97	99.84
6	25	96.15	86.36	83.72	98.04	98.04	96.78
7	21	90.48	84.21	86.49	87.18	87.18	91.32
8	30	95.24	94.74	94.74	90.32	90.32	99.05
9	21	76.36	85.71	87.80	86.36	86.36	88.46
10	30	93.75	98.31	98.31	96.67	96.67	98.12
11	22	86.27	93.33	90.91	87.50	87.50	91.31
12	31	95.38	95.08	92.06	93.75	93.75	98.52
13	23	89.36	70.27	70.27	86.36	86.36	93.98
14	25	94.12	85.11	85.11	91.67	91.67	96.34
15	26	96.30	91.67	91.67	93.88	93.88	99.62
16	20	88.37	83.33	83.33	78.95	78.95	90.16
17	30	98.36	100.00	100.00	100.00	100.00	99.90
18	18	76.60	91.89	91.89	94.74	94.74	87.88
19	7	60.87	87.50	87.50	87.50	87.50	83.77
20	16	60.38	74.42	72.73	68.09	68.09	69.45
21	21	91.34	97.67	97.67	95.45	93.33	93.94
22	30	93.75	93.55	93.55	96.77	95.24	96.48
23	5	33.33	71.43	71.43	50.00	50.00	46.18
24	27	100.00	100.00	100.00	100.00	98.18	98.70
25	16	82.76	76.92	74.07	85.71	85.71	89.93
	Mean	85.97≈	89.42≈	88.94≈	90.28≈	90.12≈	**91.80**

Symbols +, − and ≈ indicate that the competitor is statistically better than, worse than, and similar to KDT-SPSO by Kruskal-Wallis and post-hoc Dunn tests at $\alpha = 0.05$.

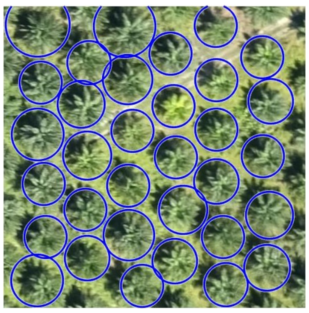
NMS and QUBO in selecting the fittest proposals were very similar. Although all methods performed equally well in detecting young palm trees and simple environment, the low precision rate achieved by all methods in a complex environment indicates that a more robust classification model, e.g. convolutional neural networks (CNN), is needed to improve the accuracy, at the expense of additional computation cost. Alternatively, the results can possibly be improved by eliminating the outliers or anomalies using post-processing techniques such as isolation forest and local outlier factor, which can be considered for further study.

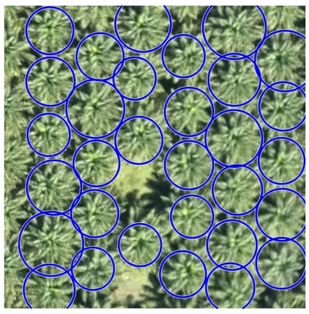
Finally, the runtime results show that KDT-SPSO required significantly less time than the sliding window technique for detecting all the palm trees in each test image without comprising the overall accuracy, proven by the higher F1-Score of 91.80% as opposed to 85.97% achieved by sliding window technique. This could be achieved because KDT-SPSO optimises palm tree locations and sizes simultaneously, whereas sliding window technique uses a brute-force approach to identify both of the parameters. As expected, S-NMS1 and S-QUBO1 were the two fastest methods as the proposals evaluated by them were far less than the rest of the methods, with some trade-off in accuracy. NMS was marginally faster than QUBO, suggesting no significant advantage of using QUBO in palm tree detection. S-NMS2 and S-QUBO2 evaluated the same amount of proposals as in KDT-SPSO; although not significant, their speed were slightly faster than KDT-SPSO. The speed of KDT-SPSO can be further improved by executing the method parallelly on a multi-core CPU, which is one of the advantages our approach has over the subsampling approach.

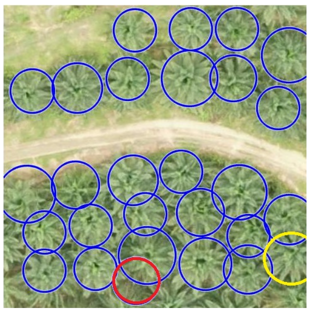
Table 5.6: Comparison of runtime (s)

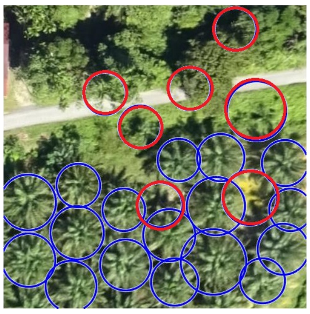
Image	Number of palms	Runtime (s)					
		SW	S-NMS1	S-QUBO1	S-NMS2	S-QUBO2	KDT-SPSO
1	33	495.45	28.29	30.84	72.64	73.72	80.48
2	31	434.80	29.21	30.34	72.03	73.17	77.88
3	27	428.98	29.25	30.60	72.18	74.61	75.65
4	17	421.92	30.03	29.48	73.24	75.00	80.77
5	32	470.27	28.01	29.43	72.18	73.4	71.08
6	25	428.47	28.97	30.19	71.72	73.85	70.14
7	21	425.56	28.40	30.09	71.47	73.04	68.61
8	30	428.00	30.30	30.95	70.54	71.67	72.83
9	21	427.19	28.72	30.54	74.38	75.85	72.66
10	30	428.34	30.17	30.20	71.50	74.04	73.05
11	22	429.97	29.99	30.21	71.85	73.39	67.82
12	31	428.17	29.80	30.36	71.84	73.68	82.08
13	23	427.44	30.27	30.95	71.98	74.08	77.35
14	25	434.98	29.83	29.58	74.05	75.47	77.56
15	26	422.56	29.82	30.58	71.69	72.91	83.06
16	20	432.57	29.83	29.32	74.56	75.84	81.89
17	30	442.47	29.12	29.74	74.28	75.71	78.94
18	18	428.09	29.54	30.20	74.10	75.26	82.31
19	7	437.33	29.56	30.14	73.46	75.98	83.57
20	16	379.38	29.08	32.43	74.93	76.24	78.40
21	21	432.19	28.71	29.55	74.70	75.78	74.56
22	30	429.07	29.11	29.94	74.89	76.10	75.80
23	5	428.00	28.92	30.84	73.05	75.64	72.69
24	27	434.08	28.29	30.15	73.68	76.30	80.22
25	16	451.24	29.29	29.37	74.03	75.82	80.19
	Mean	433.06−	**29.30** +	30.24 +	73.00≈	74.66≈	76.78

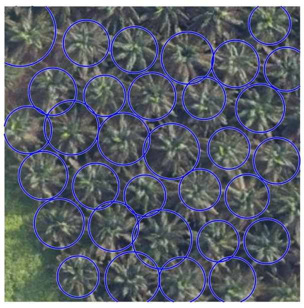
Symbols +, − and ≈ indicate that the competitor is statistically better than, worse than, and similar to KDT-SPSO by means of Kruskal-Wallis and post-hoc Dunn tests at $\alpha = 0.05$.

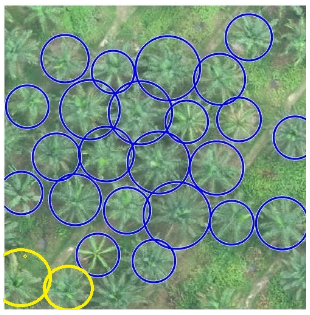

(a) Image 1

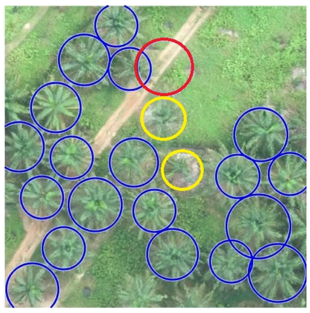

(b) Image 2

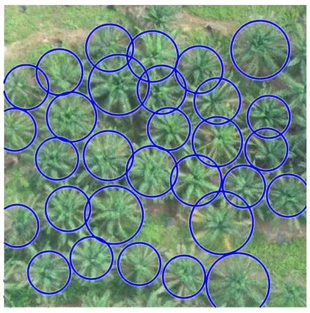

(c) Image 3

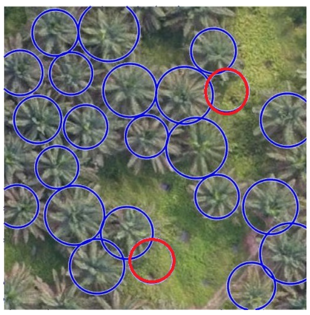

(d) Image 4

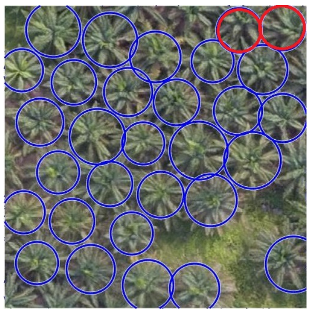

(e) Image 5

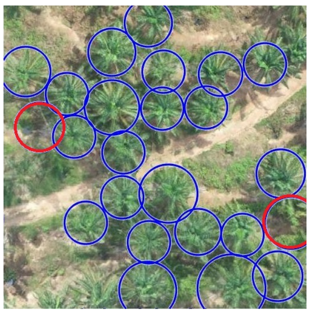

(f) Image 6

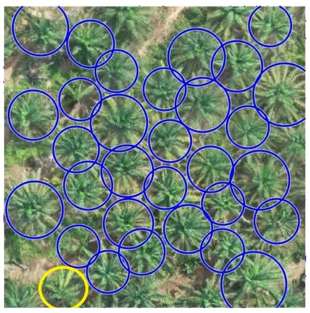

(g) Image 7

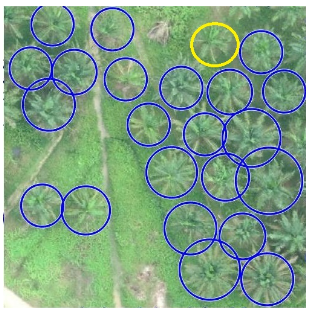

(h) Image 8

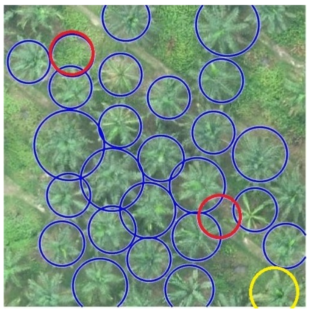

(i) Image 9

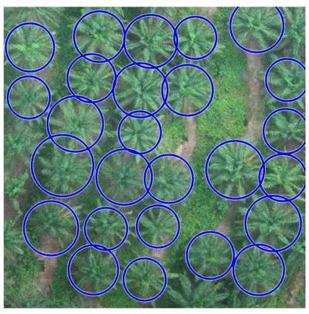

(j) Image 10

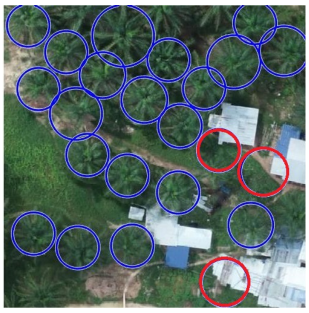

(k) Image 11

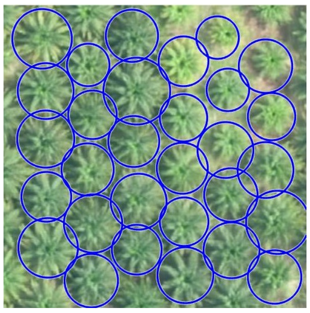

(l) Image 12

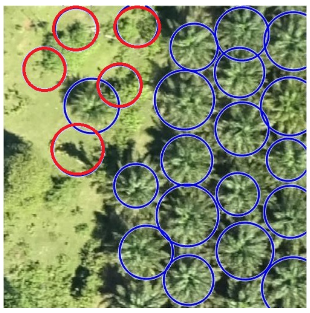

(m) Image 13

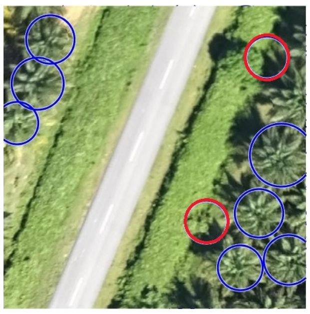

(n) Image 14

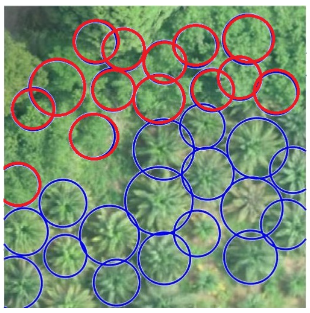

(o) Image 15

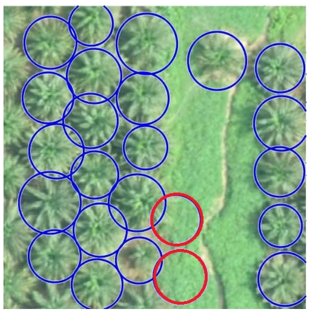

(q) Image 16

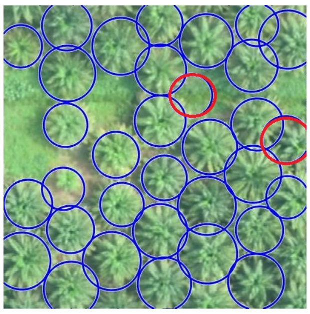

(r) Image 17

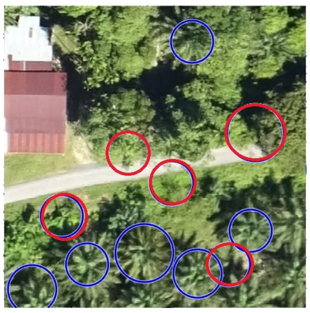

(s) Image 18

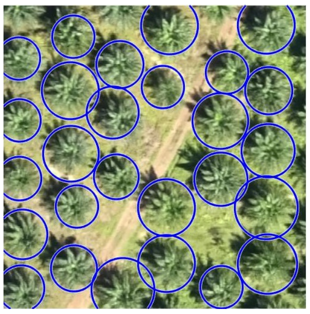

(t) Image 19


(u) Image 20


(v) Image 21


(w) Image 22


(x) Image 23


(y) Image 24

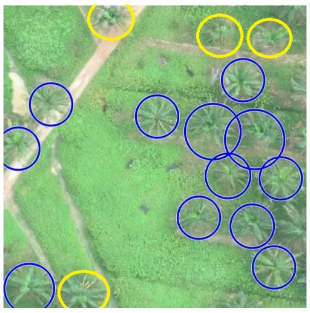
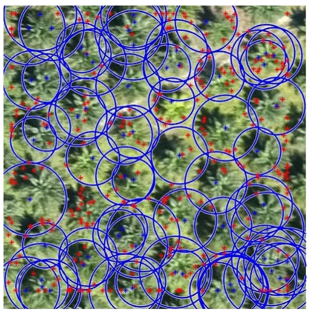
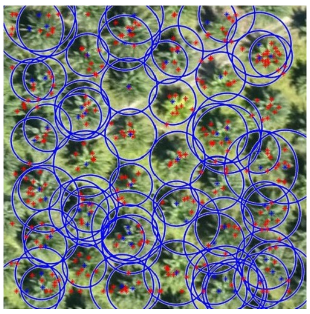
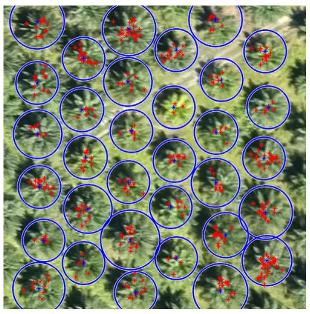
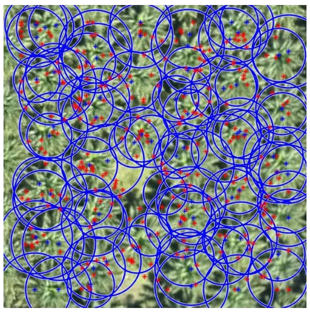

(z) Image 25

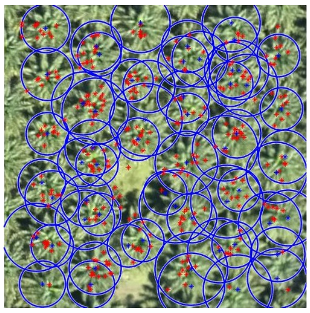
Figure 5.6: Test images showing results of KDT-SPSO detection. The blue circles indicate detected palm trees. The red circles indicate false positives while the yellow circles indicate false negatives. The images are best viewed in colour.

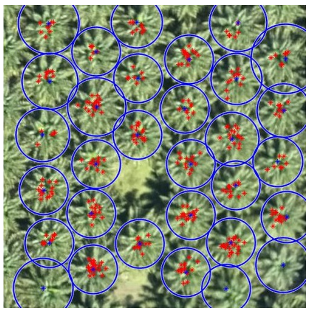

(a) Image 1, 1st iteration samples

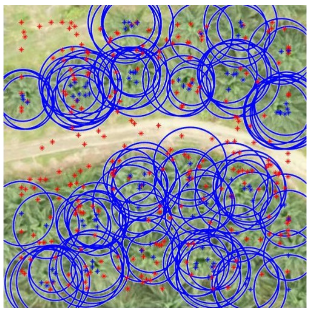

(b) Image 1, 5th iteration

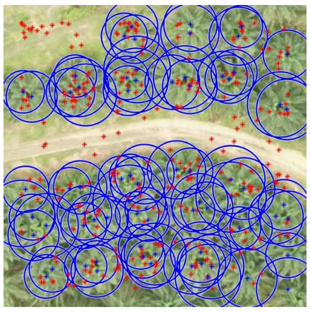

(c) Image 1, 20th iteration

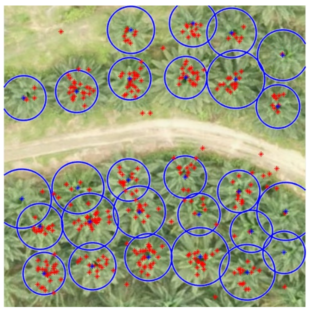

(d) Image 2, 1st iteration

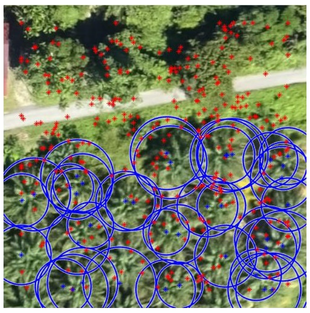

(e) Image 2, 5th iteration

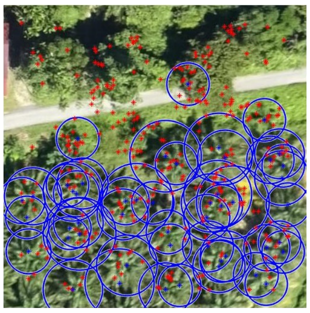

(f) Image 2, 20th iteration


(g) Image 3, 1st iteration


(h) Image 3, 5th iteration


(i) Image 3, 20th iteration


(j) Image 4, 1st iteration


(k) Image 4, 5th iteration

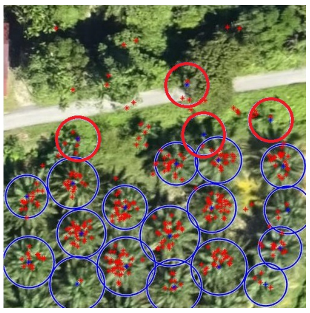

(l) Image 4, 20th iteration

Figure 5.7: Snapshots of the KDT-SPSO's searching process at 1st, 5th and 20th iteration, respectively in Image 1 (a-c), Image 2(d-f), Image 3 (g-i), and Image 4 (j-l). The red asterisks indicate particles, the blue asterisks indicate species seeds while the blue circles indicate detected canopy sizes. The canopy diameter corresponds to the width of the image patch. The red circles in Image 4 indicate false positives. The images are best viewed in colour.

5.6 Summary

In this chapter, we present how KDT-SPSO was applied in palm tree detection. The objective function of KDT-SPSO was represented by the classification score, which was computed using the LBP(SVMRBF)-R model selected in Chapter 3. Prior to the real tests, we investigated the optimal parameter settings of KDT-SPSO for palm tree detection through a set of factorial experiments. The optimised KDT-SPSO was then tested on 25 new images and compared with the greedy sliding window and subsampling approaches. Our approach successfully achieved 91.80% (F1-Score), which was the highest amongst all competitors. When we went along with the experiments, we found that the QUBO approach, which was applied to select the best subset from a large number of proposals in subsampling approach, performed marginally poorer than the NMS approach, which is out of our expectation as many of the available literatures concluded that QUBO was significantly better than NMS. We think that the benefit of QUBO is probably apparent in solving more complicated problems, such as multi-class object detections rather than the two-class palm tree detection problem. Despite the best performance, KDT-SPSO still takes a significant amount of time for fitness evaluations in large-scale UAV images where a large population size is required. We propose to incorporate additional information, which is the digital surface model (DSM), into KDT-SPSO to generate lesser but higher confidence proposals to reduce the number of evaluations.

Chapter 6

Integration of Digital Surface Model (DSM) into Palm Tree Detection Framework

6.1 Introduction

The previous chapter has detailed the general workflow of KDT-SPSO in palm tree detection. However, it is still computationally expensive to run the approach on a large-scale UAV image. The major cost is attributed to the large number of particles that need to be evaluated at each iteration. The possible improvements include reducing the number of populations and/or adjusting the stopping criterion, which require extensive experiments to obtain the optimal parameters. We present a novel approach that generates a small set of high confidence palm tree proposals from structure-from-motion (SfM)-derived digital surface model (DSM), then pass these proposals to the KDT-SPSO algorithm for evaluation. Our proposed method does not require additional training and is computation-

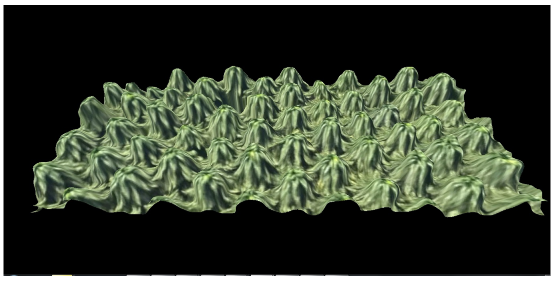


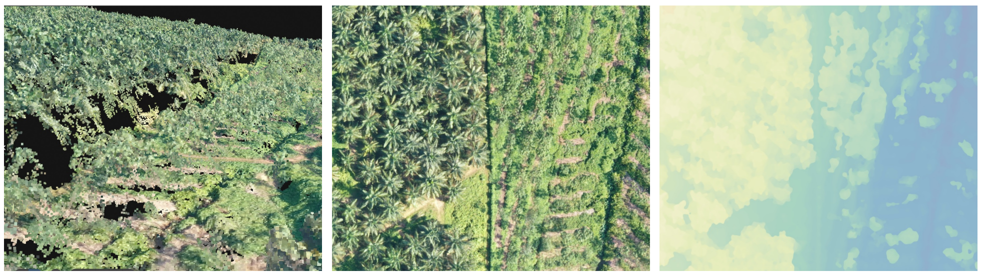
Figure 6.1: 3D representation of palm trees

ally efficient since we only need to apply local maximum (LM) operator to extract potential palm tree proposals from DSM. This is inspired by the appearance of a palm tree in 3D space that resembles a hill (Figure 6.1), and the tree centre can be identified as the hill top. In the second part of this chapter, we present a proof-of-concept study that incorporates DSM into the existing Fast R-CNN framework to increase detection accuracy and speed. The findings from this section can provide users with better computing facilities an alternative to detecting palm trees in UAV images.

6.2 Background

Three-dimensional (3D) remote sensing data have been widely utilised in mapping tree location and measuring tree height and canopy size (Korpela, 2004). Over the last two decades, the use of airborne laser scanning (ALS) in forest monitoring and mapping have become the internationally established method due to its high accuracy and quick acquisition (Næsset, 2007). However, the acquisition of ALS data requires tedious planning and high investment cost, which is less viable for relatively small scale plantations. The acquisition of overlapping images from UAVs allows the generation of 3D photogrammetric point clouds offers a flexible and cost-effective way for obtaining 3D information (Westoby et al., 2012). Stucture-from-Motion (SfM), an automated photogrammetric technique based on computer vision approach is used to generate orthomosaic and 3D point clouds, and subsequently derive a digital surface model (DSM) representing the scene's terrain from sequences of overlapping images. The SfM-derived DSM offered comparable accuracy as the ALS-derived DSM, where the SfM-derived DSM resulted in a mean overestimation of 0.18 m (SD = 0.30 m). In comparison, the ALS-derived DSM resulted in a mean overestimation of 0.28 m (SD = 0.16 m) when compared to the GPS-surveyed ground control points (Guerra-Hernández et al., 2018). Unlike ALS, SfM-derived DSM can be obtained without additional cost during the orthophoto generation (Stone et al., 2016). Figure 6.2 illustrates the outcome of the SfM photogrammetric process.

The 3D data from DSM alone can be utilised to identify tree locations quickly using LM filtering (Fawcett et al., 2019). The common assumption is that the pixel of treetops has higher elevation than its neighbouring pixels, thus being assigned as tree center. The computation of LM is faster and easier than that of machine learning approaches and does not require training. However, directly applying LM filters on DSM for tree detection without further classification using a machine learning approach can generate erroneous results because it will also include other non-tree objects that cannot be differentiated using DSM alone, particularly when large local topographic variation exists in the study area. In order to reduce the errors, some studies introduced the canopy height model (CHM) (Lisein et al., 2013; Mohan et al., 2017), which is computed by subtracting digital terrain model (DTM) from DSM. DTM is a model representing bare ground elevation, which is an estimated model derived from DSM. The operation aims to segregate trees from the ground for more accurate tree location estimation. However, it is non-trivial to derive an accurate DTM when overlapped tree canopies fully block the ground.



(a) SfM-derived 3D point cloud (b) The resultant 2D ortho-mosaic (c) The resultant DSM

Figure 6.2: The deliverables of the SfM photogrammetric process from sequences of overlapping UAV images. The blue colour in the resultant DSM represents a lower elevation value

The advantage of our proposed approach is that it does not require the derivation of DTM and CHM, yet can generate highly accurate results because the detected LM proposals (either palm or non-palm trees) will be further classified using machine learning approach. To the best of our knowledge, this is the first work that combines SfM-derived DSM and 2D visual images for tree detection using machine learning techniques, particularly palm tree. The closest study was carried out by Rizeei et al. (2018) that used image segmentation techniques to group pixels from 2D visual images with homogenous properties as an individual object (palm tree), and used ALS data to estimate the palm height. However, the approach was parametric and required heavy user input.

6.3 Integration of Digital Surface Model (DSM) into KDT-SPSO

We aim to extract as many high confidence palm tree proposals as possible from DSM to serve as the input for KDT-SPSO. The proposals still need to be evaluated by the feature-extraction-based classifier because they can be either palm trees or random objects that just protrude from the ground

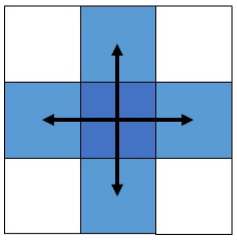
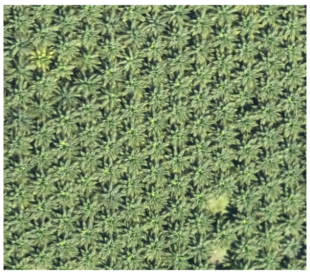


Figure 6.3: The 4-connected regional maxima operator that is used to identify potential palm tree proposals in the experiment

surface. We utilised the regional maxima operator available in Matlab for the processing. The operator slid a 3×3 pixels moving window over all pixels in the DSM to evaluate if the centre pixel's value was higher than its 4 neighbours' values (Figure 6.3). The selected locations were included in the proposal list and passed to KDT-SPSO for initialisation. Since the proposals only contained x and y coordinates, KDT-SPSO randomly assigned radius values to each of them. After that, the process of optimisation followed the flowchart as presented in Chapter 5. Firstly, we tested our approach on three large-scale UAV test images (Figure 6.4) and their corresponding DSMs to investigate the performance of KDT-SPSO on large areas. Image A represents a young palm area where the canopies have not overlapped. Image B represents a mature palm area where the canopies are heavily overlapped and only a small part is bare ground, which is more challenging than the first case. Image C is the most challenging as it is mixed with other crops and buildings. We compared the performance of the original KDT-SPSO with the enhanced KDT-SPSO integrated with DSM (named as KDT-SPSO+D) in terms of recall rate, precision rate, F1-score and computation runtime (s).

Subsequently, we repeated the same experiment on an additional 25 test images of 500 × 500 pixels (Figure 6.5) and their corresponding DSMs for statistical analyses.

(a) Image A: Immature area (1234 × 1056 pixels)


(b) Image B: Mature area (1079 × 977 pixels)

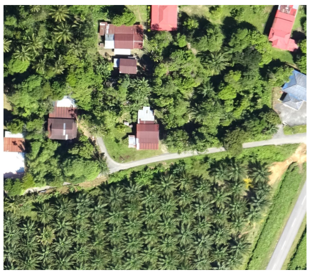

(c) Image C: Mixed vegetation area (1344 × 1261 pixels)

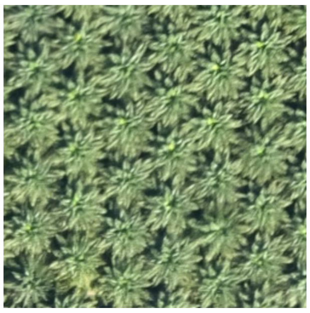
Figure 6.4: The three large-scale UAV images used in testing the effect of integrating DSM into palm tree detection pipeline.

(a) Image 1

(b) Image 2

(c) Image 3

(d) Image 4


(e) Image 5

(f) Image 6

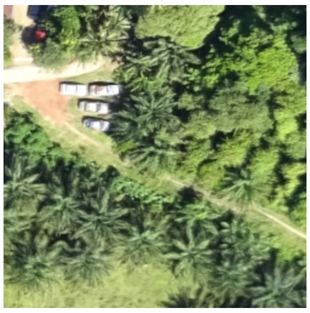
(g) Image 7

(h) Image 8

(i) Image 9

(j) Image 10

(k) Image 11


(l) Image 12

(m) Image 13

(n) Image 14

(o) Image 15

(p) Image 16

(q) Image 17

(r) Image 18

(s) Image 19

(t) Image 20

(u) Image 21

(v) Image 22

(w) Image 23

(x) Image 24

(y) Image 25

Figure 6.5: The 25 additional small-scale UAV images of 500 $\times$ 500 pixels each used in testing the effect of integrating DSM into palm tree detection pipeline

6.3.1 Results and discussion

The comparison results of the first 3 large-scale images are shown in Table 6.1, and the output are demonstrated in Figure 6.6. The results clearly show the superiority of KDT-SPSO+D over KDT-SPSO in terms of computation time and accuracy. KDT-SPSO+D was almost 10 times faster than KDT-SPSO because the number of proposals generated using DSM was 6 to 8 times less than that of the original KDT-SPSO. In KDT-SPSO, the initial number of particles was decided based on the parameters obtained from the preliminary test. The ratio was fixed across all different images and had to be larger than the number of objects possibly present in the images to avoid missed detections. The population size can be extremely large if the algorithm is applied on large-scale high-resolution UAV images. On the other hand, the number of proposals generated using DSM

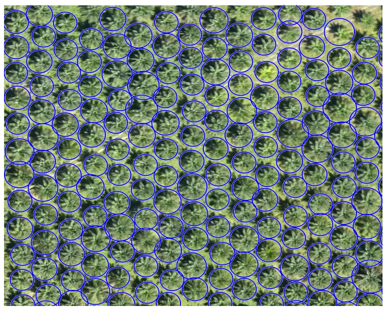
Table 6.1: Comparison of performance between KDT-SPSO and KDT-SPSO+D on the first 3 set of large-scale images

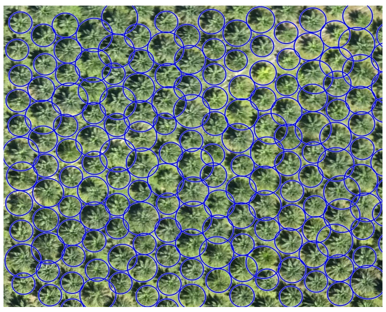
Image	Number of palms	Number of Proposals	KDT-SPSO/KDT-SPSO+D			
			Recall (%)	Precision (%)	F1-Score (%)	Runtime(s)
A	137	1955/**310**	**100/100**	**100/100**	**100/100**	438.04/**58.24**
B	116	1581/**238**	**100**/98.28	**100/100**	**100**/99.13	316.54/**45.94**
C	77	2541/**332**	**100**/98.68	51.68/**75.00**	68.14/**85.23**	529.42/**53.61**

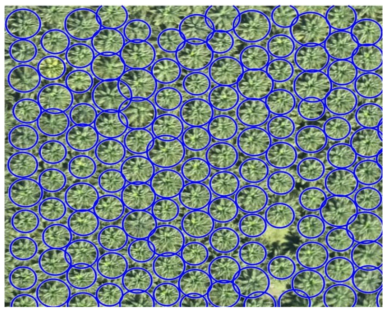
is dynamic depending on the amount of "hill" like objects present in the image.

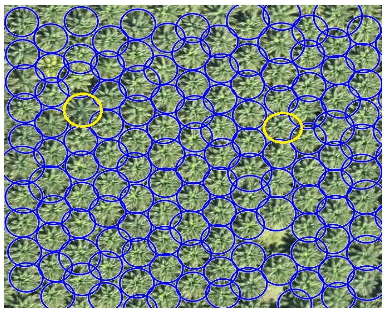
The object proposals extracted from DSM are of high confidence to be palm trees, therefore reducing the risk of false detections (i.e. Image C). This is the major reason why KDT-SPSO+D's accuracy, particularly the precision rate was higher than that of KDT-SPSO despite generating fewer proposals. The initial object proposals produced by both algorithms are shown in Figure 6.7. As observed from the figures, instead of the random positions generated by KDT-SPSO, the proposals generated by KDT-SPSO+D are mainly concentrated in areas containing palm trees. However, as DSM generally generates fewer proposals, KDT-SPSO+D may not have enough particles to explore other potential regions, increasing the chance of missed detections (Image 2). The other reason that possibly causes the lower recall rate of KDT-SPSO+D is the artefacts introduced during DSM production. The elevation of some areas may be underestimated due to error and noise (Sai et al., 2019), therefore affecting the quality of local maximum/proposal extraction.

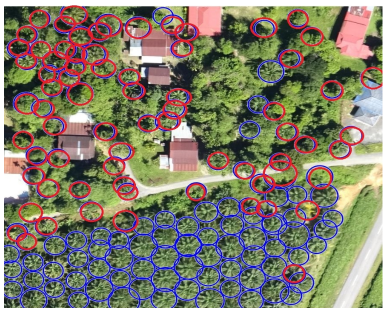
Likewise, the comparison of the additional 25 images (Table 6.2) shows that KDT-SPSO+D performed significantly faster than KDT-SPSO with no significant difference in accuracy. KDT-SPSO+D also obtained a higher average F1-Score than that of KDT-SPSO. However, in some areas with hilly and uneven topography, i.e. Image 4, Image 9, Image 13, Image 19, Image 22, and Image 23, KDT-SPSO+D performed only marginally faster


(a) Image A: KDT-SPSO


(b) Image A: KDT-SPSO +DSM


(c) Image B: KDT-SPSO


(d) Image B: KDT-SPSO + DSM


(e) Image C: KDT-SPSO

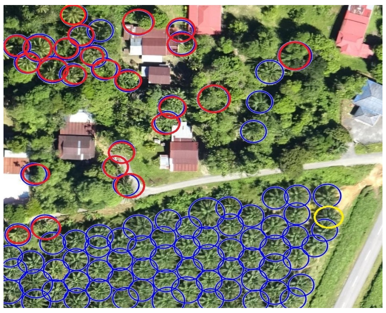
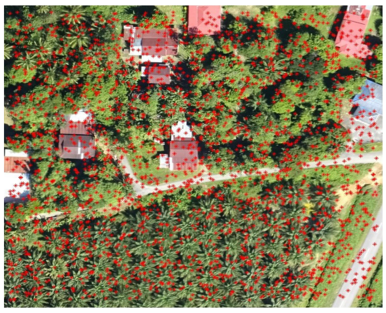

(f) Image C: KDT-SPSO + DSM

Figure 6.6: Test images showing results of KDT-SPSO (left column) and KDT-SPSO+DSM (right column) detections. The blue circles indicate detected palm trees. The red circles indicate false positives while the yellow circles indicate false negatives. The images are best viewed in colour.


(a) KDT-SPSO

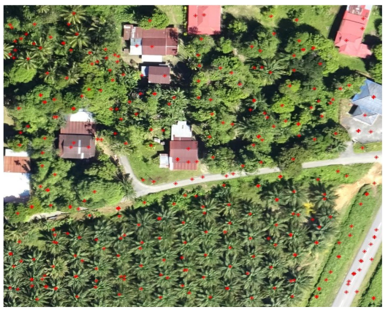

(b) KDT-SPSO+D

Figure 6.7: Comparison of the initial proposals (red asterisk) generated by KDT-SPSO(a) and KDT-SPSO+D(b).

Table 6.2: Comparison of performance between KDT-SPSO and KDT-SPSO+D on additional 25 images of 500 × 500 pixels.

		KDT-SPSO/ KDT-SPSO+D			
Image	Number of Palms	Recall (%)	Precision (%)	F1-Score (%)	Runtime(s)
1	26	92.71/**93.46**	96.49/**97.74**	94.50/**95.54**	21.46/**6.37**
2	28	85.56/**92.85**	**100**/97.95	92.08/**95.18**	18.69/**7.97**
3	30	95.06/**100**	**100**/92.04	**97.44**/95.85	19.85/**7.39**
4	27	90.27/**100**	**99.03**/94.47	94.24/**97.14**	21.69/**18.56**
5	30	**99.23**/98.18	**100/100**	**99.61**/99.04	22.15/**7.02**
6	27	**86.41**/85.18	**99.53**/96.91	**92.08**/90.52	21.33/**5.71**
7	28	95.01/**98.57**	98.24/**98.57**	96.54/**98.57**	21.12/**6.77**
8	23	85.68/**98.26**	**99.12**/87.29	91.80/**92.33**	21.00/**14.18**
9	24	93.22/**100**	97.30/**99.20**	95.01/**99.59**	**20.92**/20.96
10	27	80.63/**100**	**99.91**/86.58	89.05/**92.79**	21.53/**6.33**
11	14	**95.98**/95.71	**87.29**/72.81	**91.13**/82.49	23.50/**6.23**
12	15	83.33/**90.66**	**78.86**/68.75	**80.46**/78.17	21.43/**5.78**
13	20	**94.79**/92.00	93.27/**100**	93.56/**95.00**	21.06/**18.14**
14	30	91.18/**92.00**	**99.36**/99.31	95.00/**95.44**	20.66/**6.33**
15	29	79.88/**87.58**	**100**/97.67	88.75/**92.26**	20.95/**6.25**
16	23	93.20/**99.13**	**99.61**/94.30	96.15/**96.63**	21.27/**6.23**
17	28	**95.61**/95.26	99.40/**100**	97.43/**97.56**	21.01/**6.59**
18	16	78.12/**93.31**	**100**/95.42	87.39/**94.15**	21.67/**5.98**
19	28	91.14/**100**	**100/100**	95.23/**100**	20.80/**18.22**
20	24	66.66/**80.00**	**100/100**	80.00/**88.74**	20.77/**15.57**
21	35	**99.42**/98.57	**100/100**	**99.71**/99.26	21.69/**7.07**
22	25	**100/100**	78.12/**86.25**	87.71/**92.60**	21.67/**18.37**
23	26	**100/100**	**85.13**/83.39	**91.92**/90.89	21.13/**19.90**
24	27	**100**/93.51	93.86/**100**	**96.80**/96.60	21.95/**6.17**
25	34	**99.41**/90.44	96.16/**99.19**	**97.72**/94.60	21.37/**6.29**
Mean		90.90/**94.98**	**96.03**/93.91	92.85/**94.04**	21.23/**10.18***

Symbol $*$ indicates a difference in significance between two algorithms by means of Kruskal-Wallis test at $\alpha = 0.05$. The better performer is indicated in bold.

than KDT-SPSO. This is probably because more proposals are extracted from DSM with hilly and uneven surfaces compared to the flat ones, thus increasing the processing time. In the future, we will explore the possibility of smoothening the DSM before LM operation so that only salient points are extracted.

6.4 Integration of DSM into region-based CNN (R-CNN)

Due to the outstanding performance of incorporating DSM into KDT-SPSO, we studied whether the same approach can be applied in the state-of-the-art region-based CNN, namely Fast R-CNN (Girshick, 2015) for palm tree detection. This can reaffirm the applicability of DSM in other object detection frameworks. We can also benchmark our hybrid KDT-SPSO+D algorithm against R-CNNs that were reported achieving state-of-the-art accuracy in palm tree detection in UAV images, but requiring better computing facilities. We introduce a novel technique that uses Fast R-CNN as base network and DSM derived from SfM 3D point cloud to guide the region of interests (RoIs) generation. The architecture is summarised in Figure 6.8. Different from one-stage region-based CNNs such as Faster R-CNN, YOLO and SSD, Fast R-CNN has separate region proposal and CNN networks, thus allowing more room for improvement, i.e. using additional data and works on any dimension of input images.

6.4.1 Methods

Our proposed approach comprises two modules. The first module uses the same method as described in Section 6.3 that takes a DSM as input and generates potential palm tree locations as outputs using the LM operator. The LM filtering can be considered as a spatial constraint to guide the CNN model in searching for objects and making it faster for prediction. In order to not miss out on any potential palm trees, the smallest kernel size (3×3) was used to search for local maxima. Unlike Selective Search (SS) proposed by Uijlings et al. (2013), and Region Proposal Network (RPN) in Faster R-CNN that extract many ROIs with or without objects, LM filtering provides higher confidence ROIs that are difficult to be obtained from 2D images. The idea of the improved technique shares similar idea as in Taewan Kim and Ghosh (2016), i.e., using 3D LiDAR depth map and RGB image to increase detection accuracy. However, Taewan Kim and Ghosh (2016) extracted region proposals from both 3D and 2D data using SS that required higher computation costs.

This study used the concept of anchor boxes as implemented in RPN to account for palm trees of varying sizes. Three anchor boxes of size 40×40, 80×80 and 120×120 pixels that were close to the minimum, average and maximum canopy sizes were introduced at each potential palm tree location. The second module, which is the Fast R-CNN network first convolved input RGB image with a series of a number of convolutional and max pooling layers to a feature map. Then, the three anchor boxes were projected onto the corresponding feature map for RoI extraction. Since there are three RoIs of different sizes, a RoI pooling layer was used to extract a fixed-length feature vector. Each feature vector was subsequently fed into a sequence of fully connected layers and finally two sibling convolution layers: a box-regression layer that outputs bounding box related values and a classification layer that outputs the SoftMax probability of palm tree and "background" classes.

The VGG16 model (Simonyan and Zisserman, 2014) pre-trained on ImageNet was employed as the base network of Fast R-CNN as suggested in the original paper. As this is only a proof-of-concept study that demonstrates the viability of incorporating DSM into R-CNN framework, we did not amend the structure of the base network. Whereas we fine-tuned it for palm tree detection. To fine-tune the model, 94 images and their corresponding DSM of size 400×400 pixels that were cropped outside of the test areas were selected for training. Each image contained 10-20 palm trees and the bounding box of each palm tree was manually annotated as ground-truth. This will bias towards positive samples as they were dominating. Thus, additional 100 locations were randomly sampled in each image to increase the negative sample size. The training sample was considered as positive if its overlap ratio with the ground-truth is greater than 0.6. If the overlap was less than 0.1, it was labelled as negative. The rest of the training samples with an overlap between 0.1 to 0.6 were discarded. The losses of box-classification and box-regression layers were computed separately and combined as final loss:

$$L(p, p^*, t, t*) = L_{cls}(p, u) + \lambda[u \geq 1]L_{reg}(t, t^*) \tag{6.1}$$

in which, $L_{cls} = -\log p_u$ is the classic cross-entropy log loss of true class u. The second task loss, L_{reg} is defined as:

$$L_{reg}(t, t^*) = \sum_{i \in x,y,w,h} smooth_{L1}(t_i - t_i^*) \tag{6.2}$$

in which

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & \text{If } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \tag{6.3}$$

t and t^* are vectors representing four parametrised coordinates of the predicted bounding box and the ground-truth box associated with a positive anchor respectively where:

$$t_x = \frac{(x - x_a)}{w_a}, t_y = \frac{(y - y_a)}{h_a} \tag{6.4}$$

$$t_w = \log(\frac{w}{w_a}), t_h = \log(\frac{h}{h_a}) \tag{6.5}$$

$$t_x^* = \frac{(x^* - x_a)}{w_a}, t_y^* = \frac{(y^* - y_a)}{h_a} \tag{6.6}$$

$$t_w^* = \log(\frac{w^*}{w_a}), t_h^* = \log(\frac{h^*}{h_a}) \tag{6.7}$$

Variables x, y, w, and h denote the box's centre coordinates and its width and height while x_a, y_a, w_a and h_a correspond to the four parametrised coordinates of anchor box. This can be interpreted as regressing a bounding box from an anchor box to a nearby ground-truth box. The Inverson bracket indicator function $[u \geq 1]$ evaluates to 1 if the anchor is positive (palm), and is 0 if the anchor is negative (background). L_{reg} is ignored for background ROIs since the ground-truth bounding box is not labelled. λ controls the weights of both losses, and it was set to 1.

In this study, one image per mini-batch was randomly sampled for training. Stochastic gradient decent (SGD) solver with based learning rate of 0.0001 was run for 10 epochs to minimise the objective function. Data augmentation was ignored since there was little visual difference between palm tree objects. The mean mini-batch accuracy obtained after learning for 14 hours was 96.88%.

6.4.2 Implementation and evaluation details

To evaluate the performance of the proposed method, it was compared with the original Faster R-CNN and YOLO V2 (both were based on VGG16).

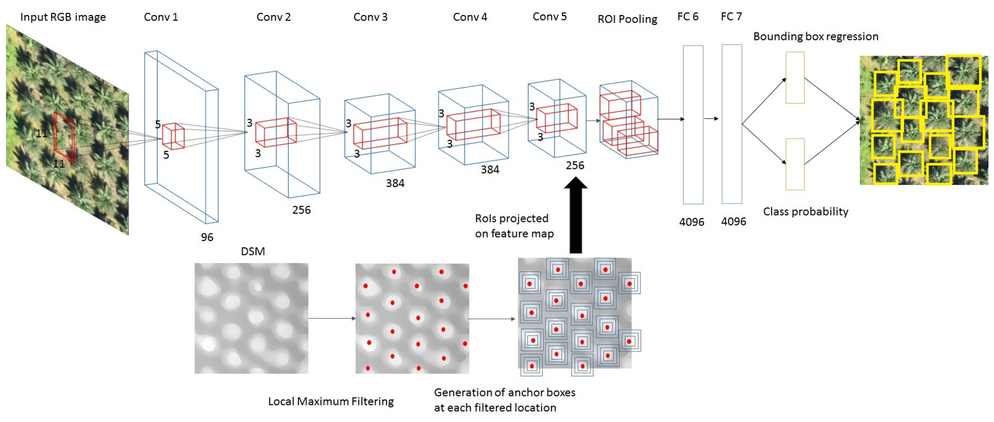


Figure 6.8: Architecture of the proposed method used in this experiment.

Due to the limitation of our computing facilities, the experiments were performed in Matlab Online, a cloud-based version of Matlab.

At test time, the 3 large-scale test images were partitioned into multiple sub-images with the size of 400 × 400 pixels via a sliding window, with stride=150 pixels and ran through the trained models. The stride ensures all regions, especially the image edges were analysed, but this will result in overlapping detections on the edges. Such a problem was alleviated using NMS.

The resolution of the palm tree is reduced in a deep convolutional feature map caused by repeated down-sampling of R-CNN, which is usually too small to contain discriminative information for reliable classification. Therefore, we cropped an image of size 2000 × 2000 pixels near Image A and partitioned it into 10 sub-images of different sizes, ranging from 200 × 200 pixels up to 2000 × 2000 pixels to test the effect of input image size on the detection accuracy and time of Faster R-CNN, YOLO, and our approach using VGG16 as a base network.

The evaluation of tree detection was based on the precision rate, recall rate, F1-Score and runtime as described in Chapter 3.

6.4.3 Results and discussion

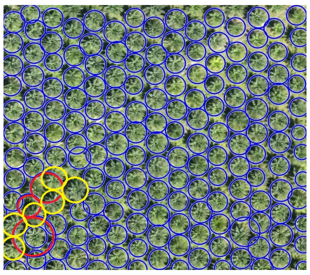
Performance comparison with other methods

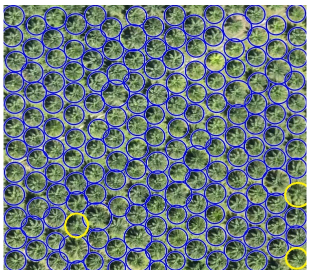
Table 6.3 shows the detection accuracy in terms of precision, recall and F1-Score of the three methods in three study areas. Results show that the proposed method achieved F1-score of 99.29% in Image A, 98.99% in Image B, and 88.75% Image C. The precision in Site 3 was low because all of the coconut trees were mistakenly identified as palm trees. The pictorial descriptions of the results are shown in Figure 6.9. In the future, we will provide more coconut tree samples for training to decrease the number of false positives. The proposed method outperformed Faster R-CNN in Image A and Image B, but slightly poorer in Image C. There were more trees not being detected by our approach than that of Faster R-CNN in Image C, and we observed that none of the local maxima was found on those false negatives, probably due to the issue of over-smoothing during the production of DSM. On the other hand, YOLO performed the best in all three images. DSM can probably be integrated into the YOLO model for further performance enhancement.

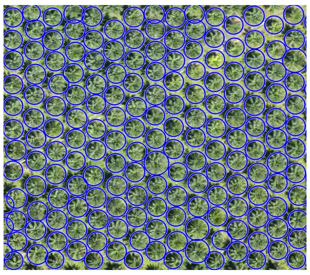
Our approach ran almost 50% faster than Faster R-CNN. The outstanding improvement in speed is that the proposed method only needs to evaluate RoIs at selected locations identified by LM filter, thus reducing the need to process a large number of background proposals. Also, LM filtering successfully reduced the number of false positives. RPN in Faster R-CNN exhaustively analyses all locations in the feature map, thus increasing the possibility of false detections. YOLO achieved the fastest detection speed, mainly because it resized the input image to 25×25 pixels, greatly reducing the convolutional layers' processing time. However, this may cause small objects not to be detected in high-resolution images.

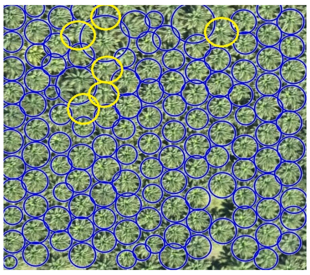
Table 6.3: Comparative results between Faster R-CNN, our approach, and Yolo in respective to recall ,precision, F1-score, and detection time in three study areas

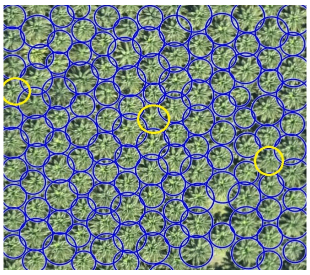
Image	Methods	Recall (%)	Precision (%)	F1-Score (%)	Runtime (s)
	Faster R-CNN	95.62	97.76	96.68	35.64
A	Ours	97.81	100	98.89	33.55
	Yolo	**100**	**100**	**100**	**27.3**
	Faster R-CNN	94.82	100	97.34	42.00
B	Ours	97.43	100	98.69	28.65
	Yolo	**100**	**100**	**100**	**22.73**
	Faster R-CNN	98.72	81.72	89.41	61.01
C	Ours	92.21	85.54	88.75	48.76
	Yolo	**100**	**87.5**	**93.33**	**36.72**

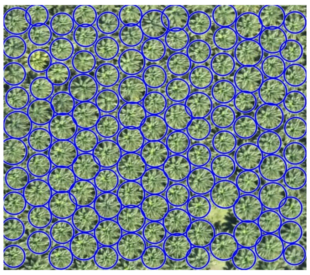

(a) Image A: Faster R-CNN

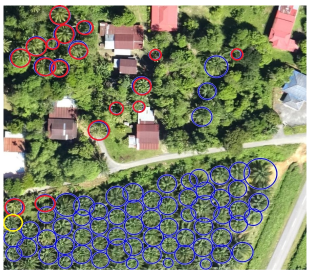

(b) Image A: Our approach

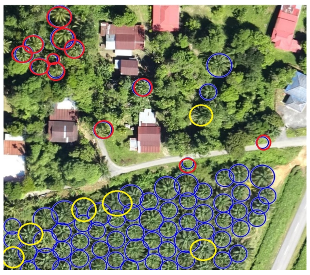

(c) Image A: YOLO


(d) Image B: Faster R-CNN


(e) Image B: Our approach


(f) Image B: YOLO


(g) Image C: Faster R-CNN


(h) Image C: Our approach

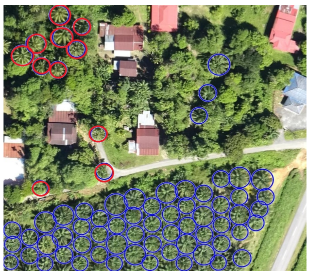

(i) Image C: YOLO

Figure 6.9: Test images showing results of Faster R-CNN (first column), our approach (second column) and YOLO (third column). The blue circles indicate detected palm trees. The red circles indicate false positives while the yellow circles indicate false negatives. The images are best viewed in colour.

Performance comparison based on AlexNet backbone

The backbone network of the R-CNNs used in this experiment was based on VGG16 model as suggested in the original study. We did an additional experiment by switching the base network to the AlexNet model to compare its performance with our KDT-SPSO approaches, and the results are demonstrated in Table 6.4. Since AlexNet is a shallower network than VGG16, its performance was poorer than VGG16-based R-CNNs as presented in Table 6.3 despite the faster detection speed. Nevertheless, our approach using DSM managed to improve the accuracy and computation time of the original Faster R-CNN (AlexNet) as expected. Interesting, our hybrid KDT-SPSO+D approach's accuracy (Table 6.1) was higher than that of Faster R-CNN (both VGG16 and AlexNet) and very close to YOLO, proving that our proposed method was as robust as the state-of-the-art R-CNN model.

Table 6.4: Comparative results between Faster R-CNN and our approach based on AlexNet backbone.

	Methods	Recall (%)	Precision (%)	F1-Score (%)	Runtime (s)
Image A	Faster R-CNN (AlexNet)	**100.00**	91.95	95.80	13.98
	Ours (AlexNet)	95.62	**99.23**	**97.40**	**10.69**
Image B	Faster R-CNN (AlexNet)	95.68	98.23	96.94	8.44
	Ours (AlexNet)	**95.69**	**99.11**	**97.37**	**6.17**
Image C	Faster R-CNN	**93.51**	40.91	56.92	17.94
	Ours (AlexNet)	**93.51**	**70.59**	**80.45**	**13.99**

Effect of input image size on accuracy and detection time

Results in Figure 6.10 show that the accuracy of our proposed method was not affected by the image size whereas the accuracy of Faster R-CNN reduced gradually when the image size was larger than 1000 × 1000 pixels due to the increase in false positive rate. On the other hand, YOLO experienced sudden decline in accuracy and resulted in 100% false negative rate when the image size was larger than 800 × 800 pixels. YOLO

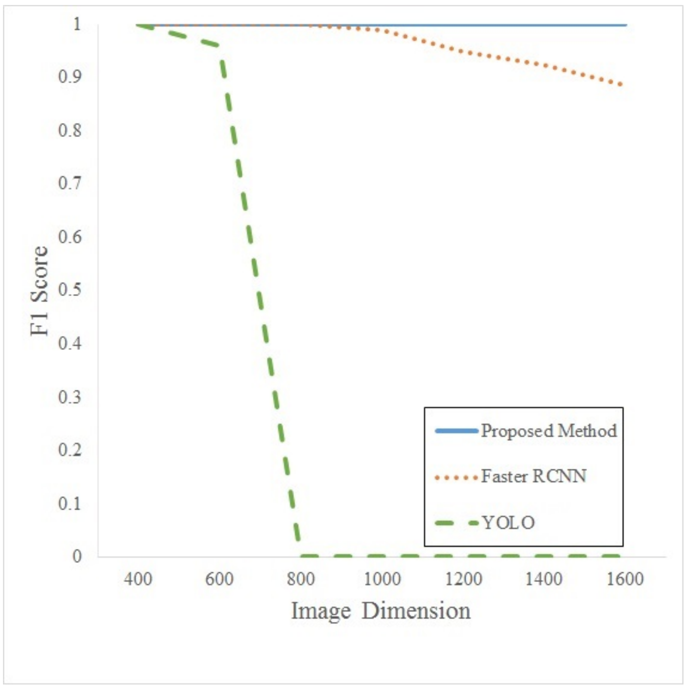


Figure 6.10: Comparison of palm detection F1 score with respect to different image dimensions.

resized all input images to the training sample resolution, i.e. 400 × 400 pixels, to maintain fast detection speed. As a result, the resolution of a single palm tree had become very low and eventually smaller than the cell size of the feature map. Therefore, it was hard to distinguish palm trees from generic clutter in the background. Although this can be improved by training the network using higher resolution images, it will increase the computation time and memory consumption quadratically. In contrast, both Faster R-CNN and our proposed method did not resize input images to a fixed resolution, thus providing greater flexibility in passing images of various sizes to the networks.

Considering that using large-scale UAV images directly will result in high computation time and memory consumption for training and testing, the large images are usually cropped on a much smaller scale with some overlaps between sub-images and processing the sub-images asynchronously. However, results in Figure 6.11 show that too many sub-images will increase computation time. This is because each sub-image has to be analysed in-

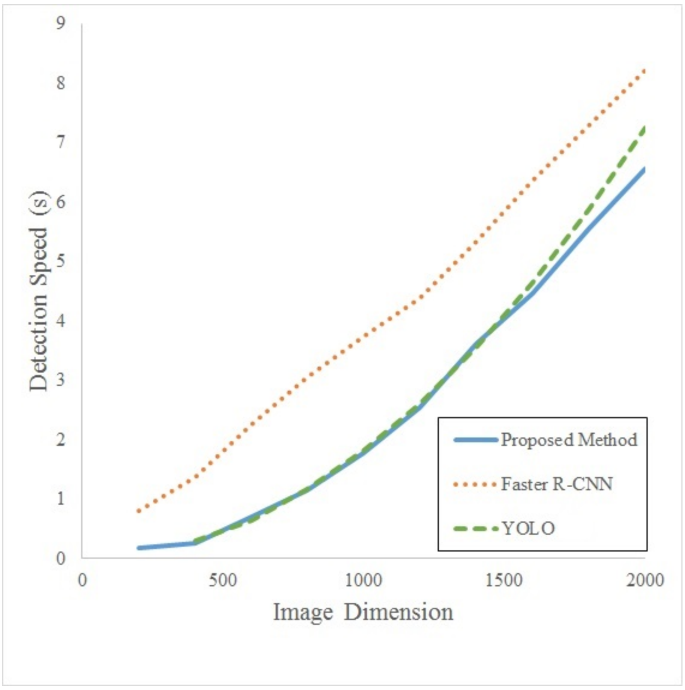


Figure 6.11: Comparison of palm detection speed with respect to different image dimensions.

dependently and the network has to process redundant information due to the overlapping regions between the sub-images.

Since the proposed method and Faster R-CNN could handle large-scale images without cropping, their rate of increase in detection time was similar to the rate of increase in image size. In contrast, multiple sub-images of size 400 × 400 pixels have to be cropped from the whole images for YOLO detection to maintain high accuracy. Consequently, the rate of increase in detection time was faster than the rate of increase in the image size. Despite YOLO had the fastest speed in detecting a single 400 × 400 pixels sub-image, its speed gradually deteriorated with the increase of input image size and eventually outperformed by the proposed method with a wide margin, i.e. in 2000 × 2000 pixels image. This implies that the proposed method is still the best option to detect palm trees in large-scale UAV images.

6.5 Summary

The first section of this chapter demonstrates the advantages of incorporating the DSM into KDT-SPSO's algorithm to improve the overall detection accuracy and computation efficiency. As the computation cost of KDT-SPSO is largely affected by the population size, the utilisation of DSM for proposal generation using local maximum filtering approach showed a significant improvement over the original KDT-SPSO. Proposal identification using LM operator is simple, fast and does not require any training.

The second section of this chapter presents a proof-of-concept study that incorporated DSM data into the existing Fast R-CNN framework to increase its detection accuracy and speed. Results show that the proposed method was up to 50% faster than Faster R-CNN and successfully achieved 93.33 to 100% average accuracy in three study areas (VGG16 backbone). Its detection speed and accuracy were higher than the state-of-the-art one-stage YOLO if they were tested on the large-scale UAV image. An advantage of our proposed method resides in the fact that it can be applied to images of any size and maintain the same accuracy. The class-agnostic proposals generated from DSM can also be utilized to detect other objects such as vehicles and buildings.

Out of expectation, the hybrid KDT-SPSO+D algorithm attained higher accuracy and yet achieved similar processing speed as the deeper VGG16-based Faster R-CNN model. In fact, it was faster than Faster R-CNN in Image 3. Although it may not be a fair apples-to-apples comparison as both algorithms were tested on different hardware (personal laptop vs cloud server) and trained using different training data (R-CNNs require image patches that consist of multiple objects for training), the results show that with little computation cost, our hybrid approach could attain state-of-the-art accuracy and perform palm tree detection task within reasonable time. We attribute the outstanding performance of KDT-SPSO+D to several reasons: 1) The carefully trained and improved feature-extraction-based classifier LBP(SVMRBF)-R is able to classify palm tree instances as accurate as CNNs, but with significant reduction in processing time. 2) KDT-SPSO offers an efficient searching strategy to locate all palm tree instances in images, and 3) DSM is used to generate high confidence proposals that likely contain palm trees for KDT-SPSO, thus reducing the time of searching other non-palm tree areas. The combination of these features makes KDT-SPSO+D a successful approach.

Chapter 7

Summary and Future Work

This thesis proposes a hybrid multimodal particle swarm optimisation algorithm that leverages feature-extraction-based classifier for objective function, a k-D tree-based speeded up mechanism for search process, and a digital surface model- (DSM) based method for high confidence proposal generation for palm tree detection in UAV images. This chapter summarises our main findings, highlights the limitations of the proposed methods, and gives directions for future work.

7.1 Evaluation of feature extraction methods for classification of palm trees

Chapter 3 presented the evaluation of six feature extraction methods and three types of classifiers for palm tree classification. The feature extractors, which were originally designed for visual images, were able to extract useful information from palm tree data and discriminate them from non-palm tree objects. The state-of-the-art fully-connected CNN algorithm that trained and classified objects from end-to-end, as well as the classic Viola-

Jones algorithm, were included for comparative study. The fully-connected CNN was found to yield the best accuracy with trade-off in computational time. The two-stage approach utilised a light-weight LBP feature extractor, and the RBF-kernel SVM classifier achieved slightly lower accuracy but three times faster than the fully-connected CNN. It was considered the best combination because it did not require long training time and specialised computer hardware to run the model. The Viola-Jones algorithm achieved the lowest accuracy despite its fastest speed due to its high sensitivity to lighting conditions and shape changes. We suggest that two types of Viola-Jones classifiers are needed to detect mature and young trees separately if the algorithm is considered.

We further enhanced the performance of LBP by means of dimensionality reduction. The feature selection process was performed to choose the most important features from the original data. The selection results show that the top 450 features in LBP were sufficient to discriminate palm tree and non-palm tree images as opposed to the original 944 features. After the reduction, it was 50% faster than the original LBP and seven times faster than the fully-connected CNN, which was a significant improvement. The selected features may not be the best combination as they were selected through a sequential forward selection approach, which added features one at a time without exploring all possible combinations. We plan to apply a more sophisticated, but computationally expensive hybrid feature selection to select the best subset of features that are more efficient and robust than the current combination. In future, we will also include the classification of various palm tree categories, e.g. diseased palms, nutrient deficient palms, pest-infested palms, etc. in our algorithm to provide more useful insights to estate managers about their plantations.

7.2 Improved multimodal particle swarm optimisation algorithm

Chapter 4 presented the main framework for the palm tree detection task focused in this thesis. One of the limitations of the existing multimodal PSO algorithm was the exhaustive nearest neighbour search (NNS) mechanism, which scaled up quadratically when the number of particles increased. We introduced a k-d tree structure to speed up the search process. The k-d tree structure reduced the computation complexity by only visiting the subtrees that most likely contained the neighbours. We statistically tested the improved algorithm, named KDT-SPSO, on a number of benchmark functions, and the results showed that it was up to 48% faster than the original SPSO. The proposed improvement can also be applied in other bioinspired algorithms or metaheuristics algorithms that require distance evaluations. The main limitation of the k-d tree framework is that its performance may deteriorate in high-dimensional space attributed to the curse of dimensionality where the algorithm needs to visit more nodes to find the nearest neighbours (Andoni and Indyk, 2017). Although our problem only involved three dimensions, it may be worthwhile to extend the trials to more complex problems with much higher dimensions to test the applicability of our algorithm and suggest possible improvements, such as using approximate nearest neighbour search to address the issue.

7.3 Application of KDT-SPSO for palm tree detection

In Chapter 5, we applied the improved multimodal optimisation algorithm (KDT-SPSO) developed in Chapter 4 for real palm tree detection problem. The enhanced LBP-based classifier was integrated into KDT-SPSO as the objective function to be maximised. Prior to implementing the algorithm in test images, we fine-tuned the parameters of KDT-SPSO. The results show that KDT-SPSO yielded the best accuracy among its competitors, the greedy search and subsampling search approaches. KDT-SPSO's computation speed was between its competitors. As some of the parameters in KDT-SPSO such as the cognitive and social coefficients and initial weight were fixed in our study according to the recommended values, we plan to explore the effect of varying the parameters on the model's performance in the future.

There are also many areas for potential research to enhance the performance of KDT-SPSO further. In the future, post-processing techniques such as isolation forest and local outlier factor can be applied to eliminate the outliers or anomalies found in the final output to reduce its error rate. Secondly, designing a self-adaptive mechanism to adjust species radius for different conditions and stopping criteria ill be interesting. Thirdly, we will explore the feasibility of extracting LBP features on the entire image once and converting the feature map to the integral image to reduce the overhead of the algorithm. Finally, optimisation of the code, parallelising of the algorithm on multi-core CPU, and implementing of the code on GPU will also be considered in the upcoming research to speed up processing. In future, we also plan to test our algorithm on the dataset published by Zheng et al. (2021), which was not available at the time of thesis writing.

7.4 Integration of digital surface model (DSM) into palm tree detection framework

In Chapter 6, we proposed a novel approach that incorporated 3D digital surface model (DSM) into the KDT-SPSO's initialisation process that significantly improved the computational speed. The proposed methodology extracted high confidence palm tree proposals from DSM using local maximum filtering approach and passed to KDT-SPSO for optimisation. Due to the use of DSM as prior information about the number and location of palm trees, the detection time was 10 times faster and the accuracy was slightly higher than the original KDT-SPSO.

We performed an additional experiment by incorporating DSM into the Fast R-CNN model (a region-based CNN) to test the applicability of DSM in R-CNN and to confirm whether KDT-SPSO was comparable to the state-of-the-art R-CNN. We found that the Fast R-CNN with DSM was 50% faster and slightly more accurate than Faster R-CNN. It helped reduce the number of false positive target windows than those based on image information alone, as is the case in KDT-SPSO with DSM. Although the proposed approach performed poorer than YOLO, the accuracy and speed of YOLO were easily affected by the input image size.

Out of expectation, our hybrid palm tree detection approach, KDT-SPSO+D achieved similar computational speed and higher accuracy than that of Faster R-CNN. The results suggest that our approach could complete palm tree detection task within reasonable time and attain state-of-the-art accuracy with lesser computational cost. Since applying R-CNNs in high-resolution aerial images requires long training hours and high performing computer, our proposed method that requires only mid- to low- range computers is a preferable alternative to R-CNN approaches. The main limitation of our approach is that it can only be applied in UAV images with DSM data, which is usually not available in satellite images that can cover bigger areas. Although DSM can be generated from high-resolution multi-view stereo satellite images, e.g. Sentinel-2, WorldView-3/4, GeoEye-1, etc., processing them into an accurate model remains a challenge as they are usually collected on different dates. Therefore, the resultant DSM may be erroneous due to different illumination conditions, geometric configurations, and topography changes (Gong and Fritsch, 2019). In addition, the resolution of satellite-derived DSM is generally lower than UAV's (e.g. 1 m/pixel vs 0.1 m/pixel), its efficiency and accuracy in extracting palm tree proposals still need to be investigated in future. As over-smoothing may be an issue of concern during the production of DSM, we plan to investigate the parameters involved in the DSM production to generate good-quality DSM for inclusion into our method. We also propose to fuse the DSM data into KDT-SPSO's objective function so that it is optimised simultaneously with the classification score to make the algorithm more robust and accurate.

Bibliography

Ahonen, T., Hadid, A., and Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041.

Al-Ruzouq, R., Shanableh, A., Barakat A. Gibril, M., and AL-Mansoori, S. (2018). Image segmentation parameter selection and ant colony optimization for date palm tree detection and mapping from very-high-spatial-resolution aerial imagery. *Remote Sensing*, 10(9).

Ammar, A., Koubaa, A., and Benjdira, B. (2021). Deep-learning-based automated palm tree counting and geolocation in large farms from aerial geotagged images. *Agronomy*, 11(8).

Andoni, A. and Indyk, P. (2017). Nearest neighbors in high-dimensional spaces. In *Handbook of Discrete and Computational Geometry*, pages 1135–1155. Chapman and Hall/CRC.

Banharnsakun, A. (2018). Multiple traffic sign detection based on the artificial bee colony method. *Evolving Systems*, 9(3):255–264.

Barrera, J. and Coello, C. A. C. (2009). *A Review of Particle Swarm Optimization Methods Used for Multimodal Optimization*, pages 9–37. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bazi, Y., Malek, S., Alajlan, N., and AlHichri, H. (2014). An automatic approach for palm tree counting in uav images. In *2014 IEEE Geoscience and Remote Sensing Symposium*, pages 537–540.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517.

Binitha, S., Sathya, S. S., et al. (2012). A survey of bio inspired optimization algorithms. *International journal of soft computing and engineering*, 2(2):137–151.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Briechle, K. and Hanebeck, U. D. (2001). Template matching using fast normalized cross correlation. In *Optical Pattern Recognition XII*, volume 4387, pages 95–102. International Society for Optics and Photonics.

Brits, R., Engelbrecht, A. P., and Van den Bergh, F. (2002). A niching particle swarm optimizer. In *Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning*, volume 2, pages 692–696. Singapore: Orchid Country Club.

Cheang, E. K., Cheang, T. K., and Tay, Y. H. (2017). Using convolutional neural networks to count palm trees in satellite images. *arXiv preprint arXiv:1701.06462*.

Chemura, A., van Duren, I., and van Leeuwen, L. M. (2015). Determination of the age of oil palm from crown projection area detected from worldview-2 multispectral remote sensing data: The case of ejisu-juaben district, ghana. *ISPRS Journal of Photogrammetry and Remote Sensing*, 100:118–127.

Chen, D., Liu, S., Kingsbury, P., Sohn, S., Storlie, C. B., Habermann, E. B., Naessens, J. M., Larson, D. W., and Liu, H. (2019). Deep learning and alternative learning strategies for retrospective real-world clinical data. *NPJ digital medicine*, 2(1):1–5.

Chen, Z. Y. and Liao, I. Y. (2019). Evaluation of feature extraction methods for classification of palm trees in uav images. In *2019 International Conference on Computer and Drone Applications (IConDA)*, pages 13–18. IEEE.

Chen, Z. Y. and Liao, I. Y. (2020). Improved fast r-cnn with fusion of optical and 3d data for robust palm tree detection in high resolution uav images. *International Journal of Machine Learning and Computing*, 10(1):122–127.

Chen, Z. Y., Liao, I. Y., and Ahmed, A. (2021). Kdt-spso: A multimodal particle swarm optimisation algorithm based on kd trees for palm tree detection. *Applied Soft Computing*, 103:107156.

Chouhan, S. S., Kaul, A., and Singh, U. (2019). Image segmentation using computational intelligence techniques: Review. *Archives of Computational Methods in Engineering*, 26:533–596.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Cuevas, E., Fausto, F., and González, A. (2020). *An Introduction to Nature-Inspired Metaheuristics and Swarm Methods*, pages 1–41. Springer International Publishing, Cham.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1.

Daliman, S., Abu-Bakar, S., and Azam, S. M. N. (2016). Development of young oil palm tree recognition using haar-based rectangular windows. In *IOP Conference Series: Earth and Environmental Science*, volume 37, page 012041. IOP Publishing.

Daliman, S., Rahman, S. A., Bakar, S. A., and Busu, I. (2014). Segmentation of oil palm area based on glcm-svm and ndvi. In *Region 10 Symposium, 2014 IEEE*, pages 645–650. IEEE.

Dong, N., Wu, C.-H., Ip, W.-H., Chen, Z.-Q., Chan, C.-Y., and Yung, K.-L. (2011). An improved species based genetic algorithm and its application in multiple template matching for embroidered pattern inspection. *Expert Systems with Applications*, 38(12):15172 – 15182.

Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39.

Fawcett, D., Azlan, B., Hill, T. C., Kho, L. K., Bennie, J., and Anderson, K. (2019). Unmanned aerial vehicle (uav) derived structure-from-motion photogrammetry point clouds for oil palm (elaeis guineensis) canopy segmentation and height estimation. *International Journal of Remote Sensing*, 0(0):1–23.

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15(90):3133–3181.

Ge, W. and Collins, R. T. (2009). Marked point processes for crowd counting. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2913–2920.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Goh, K., Teo, C., Chew, P., and Chiu, S. (1999). Fertiliser management in oil palm-agronomic principles and field practices. *Fert Manage Oil Palm Plant*, 20:21.

Goldberg, D. E. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, page 41–49, USA. L. Erlbaum Associates Inc.

Gong, K. and Fritsch, D. (2019). Dsm generation from high resolution multi-view stereo satellite imagery. *Photogrammetric Engineering & Remote Sensing*, 85(5):379–387.

Guerra-Hernández, J., Cosenza, D. N., Rodriguez, L. C. E., Silva, M., Tomé, M., Díaz-Varela, R. A., and González-Ferreiro, E. (2018). Comparison of als- and uav(sfm)-derived high-density point clouds for individual tree detection in eucalyptus plantations. *International Journal of Remote Sensing*, 39(15-16):5211–5235.

Holland, J. H. (1992). Genetic algorithms. *Scientific American*, 267(1):66–73.

Hossain, M. D. and Chen, D. (2019). Segmentation for object-based image analysis (obia): A review of algorithms and challenges from remote sensing perspective. *ISPRS Journal of Photogrammetry and Remote Sensing*, 150:115–134.

Huan Liu and Lei Yu (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491–502.

Huang, L., Ng, C.-T., Sheikh, A. H., and Griffith, M. C. (2017). Niching particle swarm optimization techniques for multimodal buckling maximization of composite laminates. *Applied Soft Computing*, 57:495 – 503.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Kong, T., Yao, A., Chen, Y., and Sun, F. (2016). Hypernet: Towards accurate region proposal generation and joint object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 845–853.

Korpela, I. (2004). Individual tree measurements by means of digital aerial photogrammetry. *Silva Fennica. Monographs*, 3.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

Lacerda, A. S. and Batista, L. S. (2019). Kdt-moea: A multiobjective optimization framework based on k-d trees. *Information Sciences*, 503:200 – 218.

Lampert, C. H., Blaschko, M. B., and Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Li, J. and Ghosh, S. (2020). Quantum-soft qubo suppression for accurate object detection. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, pages 158–173, Cham. Springer International Publishing.

Li, J., Yuan, S., Li, Q. S., and Li, B. (2018a). *A review of particle swarm optimization for multimodal problems*, pages 443–473. Institution of Engineering and Technology.

Li, W., Dong, R., Fu, H., and Yu, L. (2019). Large-scale oil palm tree detection from high-resolution satellite images using two-stage convolutional neural networks. *Remote Sensing*, 11(1).

Li, W., Fu, H., and Yu, L. (2017). Deep convolutional neural network based large-scale oil palm tree detection for high-resolution remote sensing images. In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 846–849.

Li, W., Fu, H., Yu, L., and Cracknell, A. (2017). Deep learning based oil palm tree detection and counting for high-resolution remote sensing images. *Remote Sensing*, 9(1):22.

Li, W., Ma, W., Quan, B., Pei, M., and Feng, X. (2016). A pedestrian detection method based on pso and multimodal function. In *2016 Chinese Control and Decision Conference (CCDC)*, pages 6054–6058.

Li, W., Xia, M., Fu, H., and Yu, L. (2018b). A deep learning based end-to-end oil palm tree detection approach using large-scale high-resolution uav images. In *AGU Fall Meeting Abstracts*, volume 2018, pages GC51H–0883.

Li, X. (2010). Niching without niching parameters: Particle swarm optimization using a ring topology. *IEEE Transactions on Evolutionary Computation*, 14(1):150–169.

Li, X., Engelbrecht, A., and Epitropakis, M. G. (2013). Benchmark functions for cec'2013 special session and competition on niching methods for multimodal function optimization. *RMIT University, Evolutionary Computation and Machine Learning Group, Australia, Tech. Rep.*

Lisein, J., Pierrot-Deseilligny, M., Bonnet, S., and Lejeune, P. (2013). A photogrammetric workflow for the creation of a forest canopy height model from small unmanned aerial system imagery. *Forests*, 4(4):922–944.

Liu, Q., Du, S., van Wyk, B. J., and Sun, Y. (2020). Niching particle swarm optimization based on euclidean distance and hierarchical clustering for multimodal optimization. *Nonlinear Dynamics*, 99(3):2459–2477.

Liu, T., Ye, X., and Sun, B. (2018). Combining convolutional neural network and support vector machine for gait-based gender recognition. In *2018 Chinese Automation Congress (CAC)*, pages 3477–3481.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., and Berg, A. C. (2015). SSD: single shot multibox detector. *CoRR*, abs/1512.02325.

Liu, X., Li, D., Dong, N., Ip, W. H., and Yung, K. L. (2019). Noncooperative target detection of spacecraft objects based on artificial bee colony algorithm. *IEEE Intelligent Systems*, 34(4):3–15.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2.

Luo, W., Sun, J., Bu, C., and Liang, H. (2016). Species-based particle swarm optimizer enhanced by memory for dynamic optimization. *Applied Soft Computing*, 47:130 – 140.

Malek, S., Bazi, Y., Alajlan, N., AlHichri, H., and Melgani, F. (2014). Efficient framework for palm tree detection in uav images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(12):4692–4703.

Manandhar, A., Hoegner, L., and Stilla, U. (2016). Palm tree detection using circular autocorrelation of polar shape matrix. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3:465.

Mansoori, S. A., Kunhu, A., and Ahmad, H. A. (2018). Automatic palm trees detection from multispectral UAV data using normalized difference vegetation index and circular Hough transform. In Huang, B., López, S., and Wu, Z., editors, *High-Performance Computing in Geoscience and Remote Sensing VIII*, volume 10792, pages 11 – 19. International Society for Optics and Photonics, SPIE.

Mohan, M., Silva, C. A., Klauberg, C., Jat, P., Catts, G., Cardil, A., Hudak, A. T., and Dia, M. (2017). Individual tree detection from unmanned aerial vehicle (uav) derived canopy height model in an open canopy mixed conifer forest. *Forests*, 8(9):340.

Mohan, S., Bhattacharya, S., Kaluri, R., Feng, G., Tariq, U., et al. (2020). Multi-modal prediction of breast cancer using particle swarm optimization with non-dominating sorting. *International Journal of Distributed Sensor Networks*, 16(11).

Mubin, N. A., Nadarajoo, E., Shafri, H. Z. M., and Hamedianfar, A. (2019). Young and mature oil palm tree detection and counting using convolutional neural network deep learning method. *International Journal of Remote Sensing*, 40(19):7500–7515.

Nguyen, D. T., Li, W., and Ogunbona, P. O. (2016). Human detection from images and videos: A survey. *Pattern Recognition*, 51:148–175.

Nguyen, T. T., Jenkinson, I., and Yang, Z. (2015). An experimental study of combining evolutionary algorithms with kd-tree to solving dynamic optimisation problems. In Mora, A. M. and Squillero, G., editors, *Applications of Evolutionary Computation*, pages 857–868, Cham. Springer International Publishing.

Næsset, E. (2007). Airborne laser scanning as a method in operational forest inventory: Status of accuracy assessments accomplished in scandinavia. *Scandinavian Journal of Forest Research*, 22(5):433–442.

Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987.

Özgenel, Ç. F. and Sorguç, A. G. (2018). Performance comparison of pretrained convolutional neural networks on crack detection in buildings. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 35, pages 1–8. IAARC Publications.

Parrott, D. and Xiaodong Li (2004). A particle swarm model for tracking multiple peaks in a dynamic environment using speciation. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 1, pages 98–103 Vol.1.

Parrott, D. and Xiaodong Li (2006). Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computation*, 10(4):440–458.

Pham, T. T., Rezatofighi, S. H., Reid, I., and Chin, T. (2016). Efficient point process inference for large-scale object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2837–2845.

Piotrowski, A. P., Napiorkowski, M. J., Napiorkowski, J. J., and Rowinski, P. M. (2017). Swarm intelligence and evolutionary algorithms: Performance versus speed. *Information Sciences*, 384:34 – 85.

Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press.

Qi, X., Wang, T., and Liu, J. (2017). Comparison of support vector machine and softmax classifiers in computer vision. In *2017 Second International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pages 151–155.

Qu, B. Y., Suganthan, P. N., and Das, S. (2013). A distance-based locally informed particle swarm model for multimodal optimization. *IEEE Transactions on Evolutionary Computation*, 17(3):387–402.

Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2015). You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Ren, Y., Zhu, C., and Xiao, S. (2018). Small object detection in optical remote sensing images via modified faster r-cnn. *Applied Sciences*, 8(5):813.

Rizeei, H. M., Shafri, H. Z., Mohamoud, M. A., Pradhan, B., and Kalantar, B. (2018). Oil palm counting and age estimation from worldview-3 imagery and lidar data using an integrated obia height model and regression analysis. *Journal of Sensors*, 2018.

Rokhmana, C. A. (2015). The potential of uav-based remote sensing for supporting precision agriculture in indonesia. *Procedia Environmental Sciences*, 24:245–253.

Rueda, C., Miserque, J., and Laverde, R. (2016). Validation of an oil-palm detection system based on a logistic regression model. In *2016 IEEE ANDESCON*, pages 1–4.

Rujikietgumjorn, S. and Collins, R. T. (2013). Optimized pedestrian detection for multiple and occluded people. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3690–3697.

Sai, S. S., Tjahjadi, M. E., and Rokhmana, C. A. (2019). Geometric accuracy assessments of orthophoto production from uav aerial images. *KnE Engineering*, pages 333–344.

Santoso, H., Tani, H., and Wang, X. (2016). A simple method for detection and counting of oil palm trees using high-resolution multispectral satellite imagery. *International Journal of Remote Sensing*, 37(21):5122–5134.

Shafri, H. Z., Hamdan, N., and Saripan, M. I. (2011). Semi-automatic detection and counting of oil palm trees from high spatial resolution airborne imagery. *International Journal of Remote Sensing*, 32(8):2095–2115.

Shi, Y. and Eberhart, R. (1998). A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pages 69–73.

Shi, Y. and Eberhart, R. C. (1999). Empirical study of particle swarm optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pages 1945–1950 Vol. 3.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Soltani-Mohammadi, S., Safa, M., and Mokhtari, H. (2016). Comparison of particle swarm optimization and simulated annealing for locating additional boreholes considering combined variance minimization. *Computers & Geosciences*, 95:146 – 155.

Stone, C., Webster, M., Osborn, J., and Iqbal, I. A. (2016). Alternatives to lidar-derived canopy height models for softwood plantations: a review and example using photogrammetry. *Australian Forestry*, 79:271 – 282.

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

Taewan Kim and Ghosh, J. (2016). Robust detection of non-motorized road users using deep learning on optical and lidar data. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 271–276.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2):154–171.

USDA-FAS (2020). Oilseeds: world markets and trade. *https://www.fas.usda.gov/psdonline/circulars/oilseeds.pdf*.

Vedaldi, A. and Fulkerson, B. (2010). Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1469–1472.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I.

Wald, I. and Havran, V. (2006). On building fast kd-trees for ray tracing, and on doing that in o(n log n). In *2006 IEEE Symposium on Interactive Ray Tracing*, pages 61–69.

Wang, D., Zhang, H., Liu, R., and Lv, W. (2012a). Feature selection based on term frequency and t-test for text categorization. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, page 1482–1486, New York, NY, USA. Association for Computing Machinery.

Wang, H., Moon, I., Yang, S., and Wang, D. (2012b). A memetic particle swarm optimization algorithm for multimodal optimization problems. *Information Sciences*, 197:38 – 52.

Wang, Y., Zhu, X., and Wu, B. (2019). Automatic detection of individual oil palm trees from uav images using hog features and an svm classifier. *International Journal of Remote Sensing*, 40(19):7356–7370.

Westoby, M., Brasington, J., Glasser, N., Hambrey, M., and Reynolds, J. (2012). 'structure-from-motion' photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology*, 179:300–314.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*.

Zaitoun, N. M. and Aqel, M. J. (2015). Survey on image segmentation techniques. *Procedia Computer Science*, 65:797–806.

Zhang, J., Yeung, S., Shu, Y., He, B., and Wang, W. (2019). Efficient memory management for gpu-based deep learning systems. *CoRR*, abs/1903.06631.

Zheng, J., Fu, H., Li, W., Wu, W., Yu, L., Yuan, S., Tao, W. Y. W., Pang, T. K., and Kanniah, K. D. (2021). Growing status observation for oil palm trees using unmanned aerial vehicle (uav) images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 173:95–121.