Orchard Row Finding in Aerial Images

Leonard Chuang
Department of Computer Science
University of Cape Town
chnleo008@myuct.ac.za

Ryan McCarlie
Department of Computer Science
University of Cape Town
mccrya008@myuct.ac.za

Timothy Simons
Department of Computer Science
University of Cape Town
smntim001@myuct.ac.za

1. Project Description

An increase in crop yield of up to 70% is needed to meet the projected demand from an increasing human population by 2050 [1]. This increased crop yield will have to come from agricultural land as other land will be unavailable or occupied. Other hindrances to increasing the crop yield include diseases and pests which can result in losses of up to 40% of global crop yields [2]. The modern-day solution to this problem is the application of Precision Agriculture (PA) to existing farming techniques. PA is defined as applying geospatial information to site-specific farming with the intention of both increasing and monitoring crop yield while minimising resource expenditure. Good Sustainable farming practices such as biomass monitoring, pesticide allocation and resource allocation can be effortlessly achieved with the help of Information Technology (IT). Currently, Low Altitude Remote Sensing (LARS) systems, which includes Unmanned Aerial Vehicles (UAVs), are deployed to collect high-resolution images of agricultural plots for processing.

The main problem presented in this project is the identification of the underlying tree row structure in an orchard given the orchard data obtained from an Unmanned Aerial Vehicle (UAV) which contains the tree detections. A subproblem is identifying missing trees within the tree rows when traversing the rows. When identifying the rows, two issues arise which increase the complexity of the problem: (1) tree detections are poor, resulting in different sized trees and oddly positioned trees and (2) the orchard rows follow a combination of both straight and curved lines. Curved rows are particularly difficult with little research done in the area.

There are several challenges presented by the orchard aerial inputs. These challenges vary across the different orchard inputs. In some cases, there is dense tree growth with overlapping trees. This results in erroneous detections that detect multiple trees as a single tree. In other cases, tree growth is sparse. When there are too few trees in a row, it violates the assumption that inter-row distances are smaller than intra-row distances, thus making the detection of rows very difficult.

2. Problem Statement

The principle aim of the project is to evaluate the applicability of three separate algorithms to the orchard row finding problem. Each algorithm seeks to robustly identify coherent row structures of an orchard given the initial tree detections. Height data is often unavailable or ineffectual and thus each of these algorithms should achieve said goal without relying on height data.

2.1 Aims

Since each of our approaches is novel to this domain, the primary aim is to produce an accurate, in-depth analysis of how applicable each of these approaches are to orchard row detection. Auxiliary aims of each approach include the following:

- To produce accurate tree detections in accordance with the accuracy metric defined later on.
- To be robust to erroneous tree detections.
- To be able to identify both curved and straight rows.

2.2 Research Questions

The principle research question will be described below for each member of the project. Followed by how these research questions will be evaluated and tested.

Augmented polynomial fitting principle research question:
The primary research question is to identify the robustness of a combined algorithm of the Hough Transform and a modified least squares algorithm through its ability to map out the underlying row structure of an orchard given that the orchard data contains poor object detection and could include a curved row structure.

Multi-objective Depth-First Search principle research question
The main objective is to validate whether using a multi-objective Depth-First search algorithm is sufficient for detecting a combination of straight and curved rows whilst simultaneously being robust to poor tree detections.

Modified Ant Colony Optimization principle research question
The main research question is to determine the relevance and applicability of the Ant Colony Optimisation (ACO) algorithm in solving the orchard row detection problem. Secondly, it is to

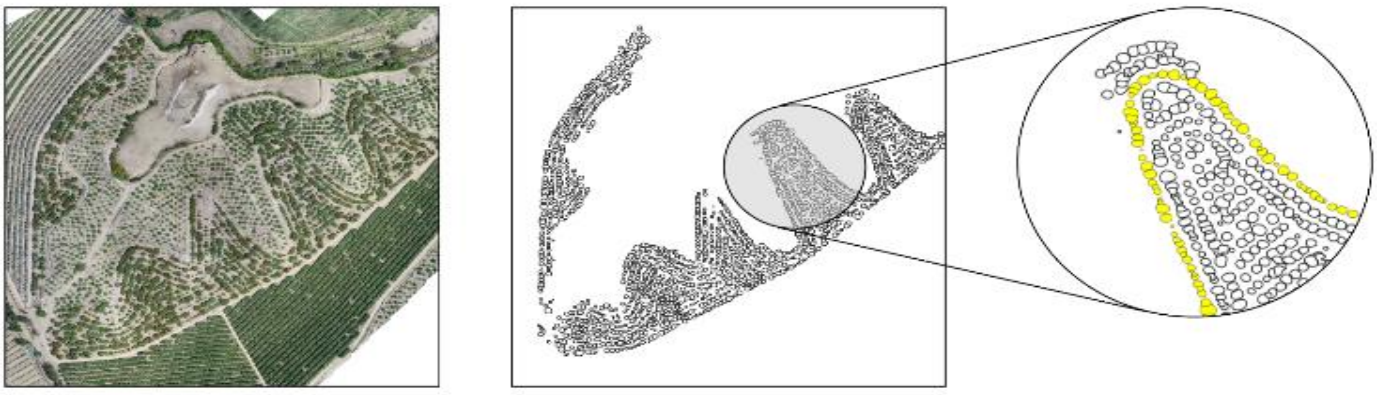


Figure 1: Example input data that shows the aerial image of the orchard and the trees identified from the image. The trees are recorded as points on a plane and no height data is available.

determine the success of a modified ACO implementation on a series of orchard inputs.

3. Related Work

3.1 Image Tiling and the Hough Transform

There is a large body of work that explores the application of the Hough Transform to orchard row detection. Unfortunately, most of this work focuses on ground images. Furthermore, the techniques explored are only able to detect straight lines. This is problematic as most orchard rows depict some degree of curvature. Soares et al. (2018) tries to solve this problem whilst exploring the application of the Hough Transform to images of coffee crop fields [3]. This approach takes advantage of the fact that a curved line can be approximated through a series of straight lines. Each aerial image of a coffee crop field is segmented into a series of tiles such that the rows in each tile are approximately straight. This allows the use of the Hough transform in each of those tiles. The detected lines are then combined to form the series of curved lines that represent the rows of the orchard. Experimental results of this method are not too promising. Detections of crop rows contain a high degree of both false positives and false negatives. This approach also relies on the assumption that rows are not too heavily curved. Rows that depict a high degree of curvature cannot use image tiling as there would be too few trees in each tile to apply the Hough transform.

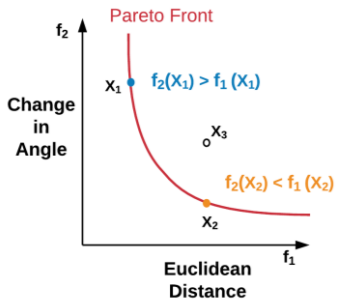
3.2 Straight Line Algorithms (Particle Swarm Optimization)

The particle swarm optimization (PSO) algorithm is a stochastic optimization method based on simulating the movement of groups of biological organisms such as schools of fish and flocks of birds. It is a meta-heuristic search algorithm that searches the solution domain independently of problem-specific details.

Xu and Tie (2013) proposed a method for straight line detection based on the particle swarm optimization algorithm. Each particle in the swarm represents a straight line initialized between two edges points. The fitness function of the PSO algorithm is the number of edge points (node centroids) that lie on the line for any instance of a particle. This method has a quick execution time and is robust to noisy data[4].

3.3 Multi-objective algorithms

The goal of any multi-objective algorithm is to identify the collection of non-dominated solutions on the Pareto Front (refer to Figure 2). These solutions represent the optimal trade-offs between the different objectives where one solution cannot be improved without worsening another solution. As seen in Figure 2, X_1 and X_2 are non-dominated solutions which cannot be changed without degrading one of the objective functions.



f_1 represents the Euclidean distance between nodes.

f_2 represents the change in angle.

X_3 is a dominated solution

Figure 2: Pareto Optimality

Coego and co-workers presented an improvement of the IDMOA* algorithm which searches for non-dominated solutions and simultaneously accounts for all objectives, called PIDMOA (Pareto Front Iterative Deepening Multi-Objective A*).

Each iteration uses a new set of threshold non-dominated vectors for the search. While searching using iterative deepening, the search will be stopped when at any node, its vector value is dominated by a vector in the threshold. This algorithm input is a directed graph, a start node and a set of possible end nodes. The algorithm finds the set of all non-dominated paths that transverse from the start node to one of the goal nodes. The algorithm stores all the sets of solutions it finds. The sets of solutions are made up of a pair, $(\gamma, f(\gamma))$ where γ is a solution node and $f(\gamma) \in F(\gamma)$ is the value of a solution path to γ [5].

3.4 Levenberg-Marquardt algorithm for curve fitting

Least-squares regression is a well-known statistical method that can approximate curves from a set of points. The method approximates a line or curve of best fit through a set of points. One such algorithm is the Levenberg-Marquardt algorithm which is also known as the Damped least-squares algorithm. Smit et al. (2010) used this method to identify curved grapevine rows in the field of Viticulture[6]. Another, more robust, least-squares algorithm is the moving least-squares algorithm which is used in surface reconstruction. The moving least-squares algorithm can detect sharp curves and cusps whereas the Damped least-squares algorithm can only determine smoothed curves.

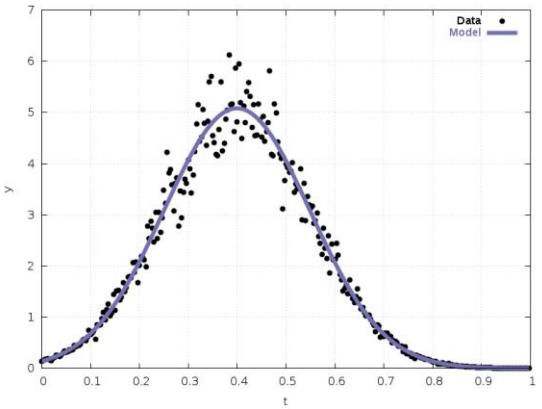


Figure 3: A least-squares regression fitting a curve through a set of points that are sparsely populated.

The problem with a least-squares approach is that the line of best fit is susceptible to noise and outliers. This is a problem with orchard data because the trees must grow in a mapped-out boundary and thus can become very dense. Although the moving least-squares algorithm applies a weighted moving least-squares, it can still be affected by noise when there are dense areas or the entire area is filled with points - this is the case with orchard data.

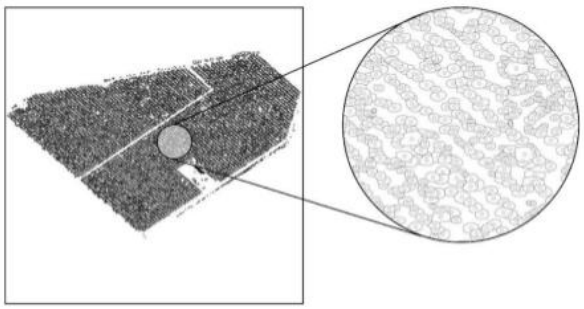


Figure 4: Example input data that contains dense tree growth which prevents an unmodified least-squares method from being applied to the data.

A solution going forward is to couple a least-squares approach with some sort of directionality approach so that the data can extract and work with a subset of data points - the KNN (K nearest neighbours) algorithm provides a closest point which when coupled with a gradient threshold can be used as a directionality guide. Another possible approach is combining a straight-line method such as the Hough Transform with a curved approach such as a least-squares

algorithm. The straight-line method can map out the straight parts of the orchard leaving a subset of points for the least-squares algorithm to work with. This reduces the noise which would otherwise reduce the applicability of a least-squares algorithm.

4. Procedures and Methods

The architectural design of the project can be described as a shared component coupled with a set of modular components. The shared component of this project refers to the data acquisition and orchard detection processing, which is elaborated upon below. The orchard processing comprises functionality common to all three approaches. From there, each team member will go on to implement their respective components, with each component being a method for solving the orchard row detection problem. Method evaluation is also divided according to each of these approaches. See the Appendix for architecture diagram.

4.1 Data acquisition

The data for this project is provided by Aerobotics, which is a company that uses aerial imagery and artificial intelligence to provide farmers with farm management solutions. Each data input represents the aerial information of an orchard. This information comprises three files, namely two *GeoJSON* files and a single *GeoTIFF* file. The first file contains the tree or shrub detections of an orchard with each detection being a polygon that has some associated confidence value. The second file represents the boundary of the orchard to be analysed and the third is just a georeferenced image of the actual orchard. The orchard image and their detections are visualised in Figure 1.

4.2 Orchard detection processing

A certain amount of detection processing needs to be undertaken prior to the application of our respective methods. The locations of the orchard detections are provided in a geographic coordinate system (GCS), which poses a serious challenge to many arithmetic calculations needed for orchard analysis. For instance, calculations such as determining the distance between two points or the area of a polygon become exceedingly complex and possibly more computationally expensive than the two-dimensional alternative. To solve this problem, we need to convert the geographic coordinates of our orchard detections into projected coordinates.

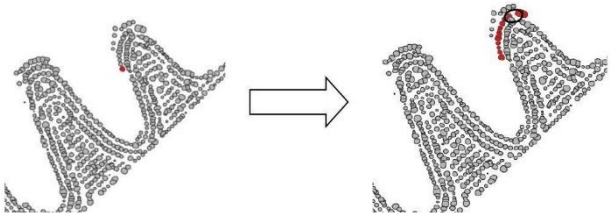


Figure 5: Shows the algorithm from a starting node expanding into a whole row. The circle indicates a trouble area where the threshold comparisons must be made to find the correct curve.

4.3 Row estimation using augmented polynomial fitting

4.3.1 Problem Reduction

The problem can be reduced to fitting a low-level polynomial of either degree one (straight line) or degree two (quadratic curve) through a set of points. The problem can then be further broken down into straight rows or curved rows. The straight rows can be accurately identified with the Hough Transform. The crux of the algorithm will be to determine whether the row is straight or curved.

4.3.2 The algorithm

The algorithm will apply the K-Nearest-Neighbours algorithm with K set to 1 from a specified source node (depicted as a red node in Figure 5) that has been identified as a starting node. The algorithm then determines the nearest neighbour - the node with the smallest Euclidean distance from the source node - and two threshold comparisons are made: (1) The first threshold is the intra-row distance(the distance between the trees) so that the algorithm can identify if the nearest neighbour is from the same row(distance is between the two nodes is less than the threshold) or from another row(distance between the two nodes is greater than the threshold). This threshold works under the assumption that the intra-row distance is likely to be less than the inter-row distance when good practices are followed. (2) The second threshold is a specified change in gradient so that the algorithm can compare the change in gradient of the two nodes to the threshold. If the change in gradient is less than the threshold then the angle of change will be small enough to limit the neighbour to exist in the same row, otherwise the neighbour could be an outlier in the inter-row space and affect the directionality of the row by joining two different rows. If the nearest neighbour satisfies these threshold conditions i.e. the distance and change in gradient between the two nodes are less than the thresholds, then the neighbour becomes the source node and the original source node is added to an array of nodes that is considered to be from the same row. If the distance between the two nodes is greater than the threshold or the only node is in the inter-row space, the row is considered to come to an end - either a missing tree or the end of the row. The nodes in the array are then extracted and a least-squares (Levenberg-Marquardt) method is applied to fit a polynomial through the points. While there are still nodes that have not been visited, the algorithm will repeat until all nodes have been classified into a row.

4.4 Orchard Row Detection using Multi-objective Depth-First Search

The row finding problem can be simplified to searching a directed graph where the nodes are tree objects, and the edges are the Euclidean distances between the nodes. Row detection, on this directed graph, can be completed by finding disjoint subgraph structures that represent tree rows. The challenging disjoint subgraphs are found by elongating confident rows (overlapping rows that are in a straight line) to capture the curved and uncertain straight rows. My approach uses a straight row detection algorithm to identify the “confident rows” and then uses a Multi-objective algorithm to extend these confident rows to identify the difficult curved row segments.4.4.1 Initial Straight Row Detection Algorithm

The reason for implementing an initial algorithm is to reduce the workload on the main algorithm. Rather than letting the main algorithm try to identify straight and curved rows simultaneously, the dedicated straight-line algorithm will create a better platform for the main algorithm to start from. The initial algorithm would reduce the number of nodes the main algorithm needs to process drastically and will essentially reduce the total time of execution.

The initial algorithm is a Particle Swarm Optimization algorithm to identify straight lines that run parallel to a few object centroids. This PSO’s fitness function will capture the number of nodes on a given line in the search space (with some leeway so that nodes can be near the line). The encoding of the particles will be the parameters that make up the line. A point will be deemed part of the line if it is near the candidate line, this means it is smaller than the threshold distance, $\boldsymbol{D_L}$, from the line. This result comes from various mathematical operations that calculate the distance between a point and a line in the domain space. This algorithm will be a direct implementation of the PSO straight-line detection method covered in the related work section.

This PSO is a great algorithm for the initial straight-line algorithm because of the quick execution time, suitable results and algorithm implementation ease. Due to this algorithm only being a pre-cursor to the main algorithm, it needs to be easy to implement and run quickly. Additionally, this algorithm produces multiple lines in a single execution while still being extremely robust to outliers.

There are various Python libraries for using multi-objective optimization with metaheuristics. The best frameworks currently are PyGMO, Platypus and DEAP. The PSO algorithm could be easily implemented via one of these frameworks.

4.4.2 MO-DFS main detection algorithm

The depth-first search algorithm is an intuitive method for discovering the best elongated rows. The initial search will use a greedy like search to identify the all potential nodes which to DFS into. While searching into these nodes, at each sub-node, there is a decision of which node to further search into. The DFS can determine which will be the best objects to add to a row.

The main algorithm to capture straight and curved difficult lines is a multi-objective depth-first search (distance, angle, and height if available). In the case of height data being available, implementation of 3D objective functions will integrate this additional data into the algorithm. The algorithm will be discussed as if there is no additional height data available.

In the orchard row detection problem, it can be argued that we do not know whether a smaller distance or smoother curvature between trees will be preferred during detection. Each orchard will differ by the objective functions that govern the coherent rows of trees. This means using a single-objective algorithm or a weighted combination of objectives may not produce a suitable output for every input. For example, one orchard may portray rows with constant curvature and missing trees whilst another may have distinct rows with erratic curvature. In this case, the first orchard will prefer the curvature objective whereas the second prefers the distance objective. Other orchards may prefer some combination of the two. For this reason, we would need to identify the Pareto optimal set.

4.4.3 Objectives

The two objectives are Euclidean distance and the average change in angle between all the adjacent nodes in a row. Both of these objectives will be minimized. Penalty terms will be added to account for uncertain classifications.

4.4.4 Algorithm Explained:

The initial straight row detection creates easily identifiable straight rows. Each row is a collection of various (blue) assigned tree objects where the row is represented as the red dotted line (see Figure **6**). This algorithm is initiated from the two endpoints (black points in Figure **6**) of the previously created straight rows and will propagate outwards to the potential candidates. These candidates will be selected using the K-Nearest Neighbours algorithm, using Euclidean distance. Each candidate node and sub-node are chosen if they are within the maximum distance threshold and maximum angle change threshold from the associated straight row (refer to the yellow lines to potential extension objects in Figure **6**). When the node propagates, it will search from the source into each of the candidate nodes and it will recursively "dive into" each potential sub-node until a specified depth. While traversing the different sub-nodes, the algorithm will analyse the potential objective values of that entire candidate row (fixed row and all candidate sub-nodes). The algorithm will accept the extension if it is the best solution in the set of solutions produced (seen in Figure 7). The row will then identify the new endpoints and rerun the algorithm from these endpoints until some stopping criteria are met. The direct implementation will be modelled from the framework of the PIDMOA* using the specified objectives.

PIDMOA* has been selected due to it being an excellent depth-first search algorithm which can take multiple objectives into account when searching for a solution. It makes use of iterative deepening, which utilizes the completeness of a breadth-first search algorithm

There are currently no python libraries for easily implementing the PIDMOA* algorithm. Nevertheless, pseudocode is provided for the algorithm and there is an abundance of depth-first search academic literature.

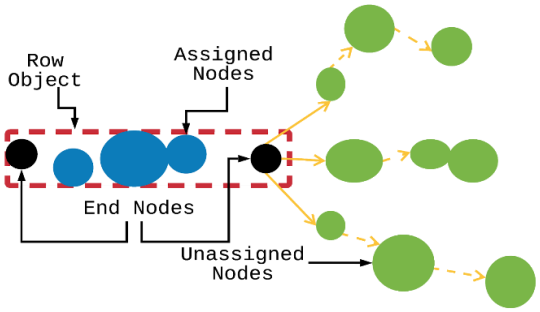


Figure 6: MODPS algorithm

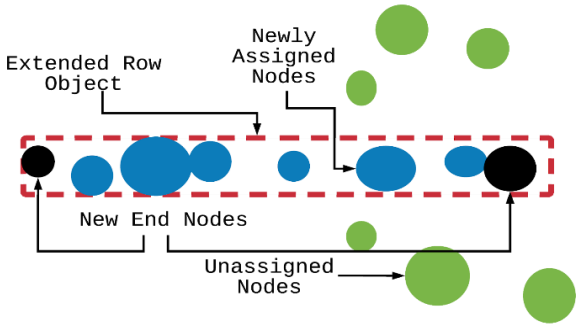


Figure 7: MODPS algorithm assigning new nodes to a row

4.4 Modified ACO approach

Ant Colony Optimisation (ACO) is a metaheuristic for solving hard combinatorial problems, especially those which can be reduced to finding optimal paths through graphs. This is ideal for orchard row detection as an orchard row is an optimal path in a graph, where the graph is the set of tree detections. ACO is loosely based on the biological behaviour of ants as they search for food in an environment.

The modified ACO algorithm aims to improve on this idea by incorporating domain-specific heuristics into the general ACO algorithm. The fundamental difference between the proposed modified ACO and the general ACO algorithm is that a traversal of an ant in the construction phase gives a path that represents a candidate solution to the problem. In the modified ACO, an ant's path represents a candidate row in the corresponding orchard. Stated differently, ACO algorithms produce many candidate solutions with every iteration of the algorithm but the modified version only obtains a single solution once the algorithm has completed. This is an important difference that motivates many of the design decisions of the modified ACO.

The ants in the modified ACO will traverse the tree detections according to distance and angle metrics. These two metrics form the heuristics of the modified ACO algorithm. The idea behind this is that trees that are closer together and trees that follow the same line or curve, will generally be in the same row.

The strength of the modified ACO algorithm comes from the phase known as *local search*, which takes the ants' constructed solution and tries to find a better solution. In this phase, the algorithm employs domain-specific heuristics to improve on the candidate rows. These heuristics utilise the knowledge specific to how farmers plant orchard rows. This knowledge includes the fact that no two rows will intersect, rows will never have sharp angles or the assumption that intra-row distances will always be smaller than in inter-row distances. It cannot be overstated that this is the *coup de grâce* of the modified ACO algorithm. No prior works were able to

utilise these heuristics as none of them have their basis in graph theory.

A further aspect of *local search* worth mentioning is the mechanism of removing candidate rows. When the pheromone trail of a candidate row surpasses some threshold, this row will be removed from the search space and placed in a global set (which undergoes postprocessing at a later stage). This has a great biological correspondence when thought of in terms of how actual ants search their environment. The tree detections disappearing from the search space is equivalent to food being finished in the ants' environment. The pheromone trails associated with the removed candidate row will disappear allowing the next best candidate row to be identified.

There is a Python framework for ACO, known as ACO-Pants, but it is geared towards problems that can be reduced to an instance of the Travelling Salesman Problem (TSP). Given the modified nature of the proposed ACO algorithm, ACO-Pants is inapplicable and thus implementation for the modified ACO will have to be from the ground up.

4.5 Evaluation

Row detection analysis and evaluation will be a challenging part of the project. This is partly due to the lack of labelled data and partly due to the small amount of data. The row detections of our respective algorithms will be compared to hand-detected rows of the orchard inputs. The accuracy metric will be derived from the comparison between edge detections and true edges where each row comprises a series of edges. In the case of the MO approach, metrics associated with Pareto extent and Pareto spread will also be considered. Pareto spread or Pareto diversity shows solutions that uniformly cover the pareto optimal front.

5. Ethical, Professional and Legal Issues

This project and data do not involve any sensitive or private data that belongs or can represent humans in any way possible. Therefore, there are no ethical ramifications with regards to people. However, there may be with regards to the farming industry as this data is a real-life image of an orchard that is owned by a farmer. The data was given to us unconditionally for the sole purpose of research. This indicates there are no commercial/legal issues.

6. Anticipated Outcomes

Since all three proposed methods are novel approaches to detect orchard rows, our primary anticipated outcome will be a review of how applicable these methods are to this problem domain. Our methods take advantage of existing tree detections, using each detection as a node in a graph. Graph theory is an unexplored area when trying to detect orchard rows from aerial images and therefore our project will not contain comparative results with previous works.

6.1 Expected Impact

We aim to introduce a new paradigm in the area of aerial farm management. Previous works utilise image processing as the basis of their row detector. We use tree detections, in the form of polygons, as the basis of our methods. This new approach may yield accuracy levels that exceed anything that was done before. The recent rise in the fidelity of object detection allows us to explore this space.

6.2 Key success factors

- Successful understanding and implementation of our novel approaches to this problem domain.
- In the case of computationally expensive methods, finishing enough experiments in time to draw reliable conclusions.
- Extensive testing and analysis to show the feasibility of the relevant approaches in successfully achieving accurate orchard row detection.

7. Project Plan

This section will cover a detailed plan on how we will execute the project.

7.1 Risks

The potential risks associated with this proposed project have been highlighted in the risk matrix. Each risk has been ranked on its probability of occurring and the impact should the risk occur. Additionally, each risk has been considered in terms of its associated consequence and mitigation.

See appendix for Risk Matrix.

7.2 Timeline

The project starts on the 30th of March and concludes on the 19th of October. Due to the COVID-19 pandemic, dates are subject to change.

7.3 Resources required

The following resources are needed for the completion of this project:

- Python 3 - high-level interpreted language for developing the row finding application
- QGIS - an open-source desktop application for viewing of the geospatial data

Python Libraries:

- Fiona - for reading and writing georeferenced geometry
- Rasterio - for manipulating georeferenced imagery
- GDAL - for manipulating geospatial raster data
- Shapely - for manipulating geometry
- Numpy - for handling multi-dimensional data

7.4 Deliverables

The major deliverables for the project include:

- Literature survey due
- Project proposal due, including project plan
- Initial software feasibility demonstration
- Project paper final submission
- Project code final submission
- Final project demonstration
- Poster due
- Web page

7.5 Milestones

The major milestones for the project include:

- Project allocation
- Literature survey draft due
- Project Proposal draft due
- Review of staff feedback on proposals
- Revised proposal finalized and uploaded to Vula
- Develop software evaluation scripts
- Final code refactoring
- Final complete draft of paper

7.6 Work allocation

The project work will be split among the team members in the following ways:

- Ryan and Tim will develop the pre-processing framework.
- All members will assist with the various written artefacts that are developed throughout the project lifecycle.
- All members will produce a row finding algorithm for straight and curved rows which will concurrently identify missing patches trees.

- Leonard will implement the augmented polynomial fitting method.
- Ryan will implement the multi-objective depth-first search method.
- Tim will implement the modified Ant Colony Optimisation method.
- Leonard will hand-labelled the unlabelled dataset provided by Aerobotics

References

[1] Oerke, E.-C. J. T. J. o. A. S. Crop losses to pests, 144, 1 (2006), 31-43.

[2] FAO-FOOD and ORGANIZATION, A. *The state of food and agriculture*. Livestock in the balance Rome, City, 2009.

[3] Soares, G. A., Abdala, D. D. and Escarpinati, M. *Plantation Rows Identification by Means of Image Tiling and Hough Transform*. City, 2018.

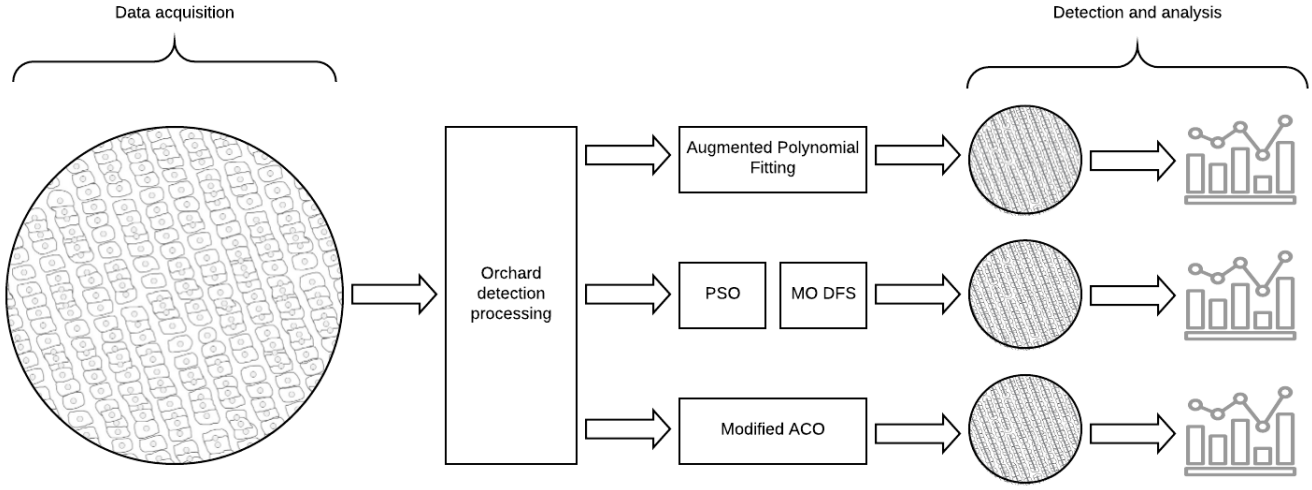
[4] Xu, S., Tie, J. J. S. and Transducers Straight Line Detection Based on Particle Swarm Optimization, 160, 12 (2013), 653.

[5] Coego, J., Mandow, L. and De La Cruz, J. P. *A new approach to iterative deepening multiobjective A*. Springer, City, 2009.

[6] Smit, J., Sithole, G. and Strever, A. Vine signal extraction–an application of remote sensing in precision viticulture. *South African Journal of Enology and Viticulture*, 31, 2 (2010), 65-74.

Appendix

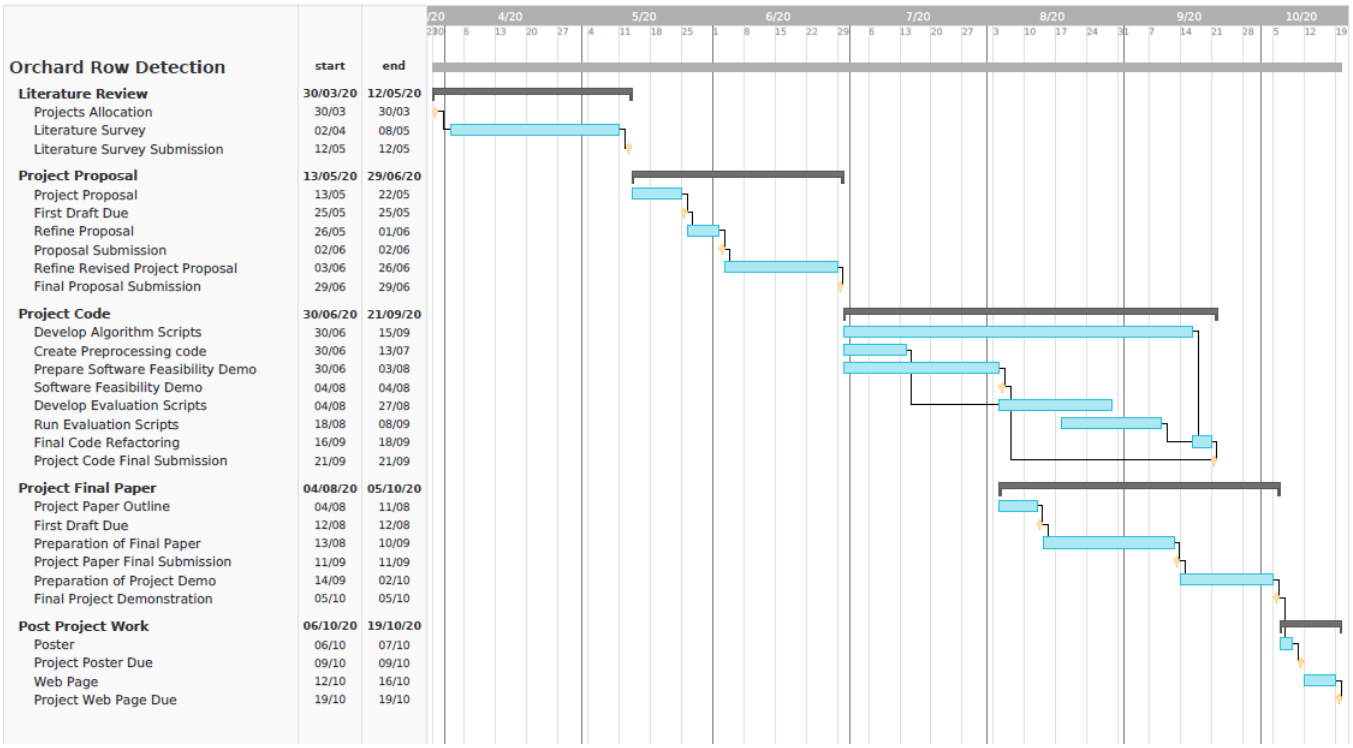
Architecture Diagram



Risk Matrix

Risk Condition	Probability	Impact	Consequence	Mitigation
Team members dropping out of the project	Low	Low	The project workload has been well distributed; therefore, no major pitfalls will occur.	The remaining workload will be distributed among the remaining team members.
Selected algorithm implementation too complex	Med	Med	We would not be able to identify if the algorithm would solve the problem. Termination or Delay of the project may follow.	We would implement our backup algorithms or try to simplify the complex algorithm.
Aerobotics retracting their unlabelled data	Low	Med	The true accuracy of our algorithms cannot be tested. We may have insufficient testing data.	Synthetic datasets can be produced to replicate what was previously seen in the orchard data.
Infeasible execution time of algorithms	Med	High	The project timeline will have to be adjusted.	Use GPU enabled systems to speed up the execution time.
Failure to meet the final project deadline	Low	High	This will result in failure in this module.	Try to reschedule dates with our supervisor to allow for the completion of the project.
Selected algorithms are unable to capture underlying patterns	Med	High	The algorithms will receive bad accuracy scores.	Implement our alternative back-up algorithms.
Trees in orchard data are not planted in any structured manner and do not fulfil underlying assumptions	Low	High	The algorithms will not be able to capture any underlying patterns and will receive an unfavourable accuracy	Request for more structured data from Aerobotics or try to implement new algorithms.

Gantt chart



Orchard Row Detection	start	end
Literature Review	**30/03/20**	**12/05/20**
Projects Allocation	30/03	30/03
Literature Survey	02/04	08/05
Literature Survey Submission	12/05	12/05
Project Proposal	**13/05/20**	**29/06/20**
Project Proposal	13/05	22/05
First Draft Due	25/05	25/05
Refine Proposal	26/05	01/06
Proposal Submission	02/06	02/06
Refine Revised Project Proposal	03/06	26/06
Final Proposal Submission	29/06	29/06
Project Code	**30/06/20**	**21/09/20**
Develop Algorithm Scripts	30/06	15/09
Create Preprocessing code	30/06	13/07
Prepare Software Feasibility Demo	30/06	03/08
Software Feasibility Demo	04/08	04/08
Develop Evaluation Scripts	04/08	27/08
Run Evaluation Scripts	18/08	08/09
Final Code Refactoring	16/09	18/09
Project Code Final Submission	21/09	21/09
Project Final Paper	**04/08/20**	**05/10/20**
Project Paper Outline	04/08	11/08
First Draft Due	12/08	12/08
Preparation of Final Paper	13/08	10/09
Project Paper Final Submission	11/09	11/09
Preparation of Project Demo	14/09	02/10
Final Project Demonstration	05/10	05/10
Post Project Work	**06/10/20**	**19/10/20**
Poster	06/10	07/10
Project Poster Due	09/10	09/10
Web Page	12/10	16/10
Project Web Page Due	19/10	19/10