CS/IT Honours Final
Final Paper 2020

Title: **Orchard Row Identification using Aerial Images**

Author: **Ryan McCarlie**

Project Abbreviation: **ORCHARD**

Supervisor(s): **Patrick Marais**

Category	Min	Max	Chosen
Requirement Analysis and Design	0	20	0
Theoretical Analysis	0	25	0
Experiment Design and Execution	0	20	5
System Development and Implementation	0	20	20
Results, Findings and Conclusions	10	20	20
Aim Formulation and Background Work	10	15	15
Quality of Paper Writing and Presentation	10		10
Quality of Deliverables	10		10
Total marks	**80**		

Orchard Row Identification using Aerial Images

Ryan McCarlie
Department of Computer Science
University of Cape Town
September 2020

ABSTRACT

The human population has grown rapidly in recent years, significantly increasing the need to optimize and safeguard food production and security. Subsequently, the agricultural industry is faced with the challenge of meeting the growing demand for food. To meet the projected food demands of 2050, agricultural production will need to rise by 70%. This work presents an automated approach to identify individual rows of trees which are planted coherently within an orchard, using images obtained from an unmanned aerial vehicle. Automatically detecting rows of trees or crops allows farmers to quickly establish where there are issues with yield and fertility. The proposed method introduced is a robust Multi-objective Depth First Search algorithm (MODFS) which can accommodate uncertain and curved rows. This row detection problem can be simplified to a graph theoretic problem which is remarkably like crop row detection. Thereafter, to solve the row identification problem, various optimization techniques are used to identify graph structures that represent rows made of trees. To build the MODFS algorithm, research was conducted on the industry-standard pathfinding algorithms, state-of-the-art multi-objective evolutionary algorithms and finally image pixel analysis methods. The MODFS is statistically shown to be robust to straight orchard rows while showing promising results for the identification of curved rows. The MODFS can capture steadily curved orchards, but struggles to identify the underlying pattern of orchards with dense pockets of trees or sharp curves. This method is adaptable to multi-orientated straight orchards and will produce robust rows even with multiple orientated substructures within the orchards. The competitive results of the algorithm in numerous cases confirm that it can identify orchard rows with an acceptable level of certainty.

CCS CONCEPTS

• Precision Agriculture • Optimization • Row Detection

KEYWORDS

Line Detection, Path Planning, Multi-objective Search

1. Introduction

The demand for the agricultural industry to provide sufficient crops is increasing exponentially due to the rapid growth in the human population. Unfortunately, the space for agricultural expansion is limited as the majority of land available is currently occupied or unavailable. Consequently, the increase in production needed to fulfil the demand must come from a higher crop yield on the currently occupied land [1]. Annually, the loss of crops due to diseases and pests can reach 40% of the global crop yields and this is expected to increase in the years ahead [2]. The agricultural industry must deploy methods that maximize farm profitability, minimize crop loss and reduce their environmental impact in order to meet demands [3]. A modern-day solution to these challenges is Precision agriculture (PA) which has recently surged in the farming industry due to the labour shortage coupled with the economic pressures on farmers. PA encourages an agricultural system that benefits from minimal input effort but still maintains a highly efficient sustainable system. Unmanned Aerial Vehicles (UAV) are the new standard for surveying a large landmass and are essential in the precision agriculture workspace. With the mass scale of modern farming, it is not uncommon to have numerous patches where trees are unable to grow or have died out. To ensure the farmers have the best possible understanding of their orchards, this project provides a new cost-effective method to fix these agricultural issues promptly. Through the mass surveillance of orchards, the identification of areas where trees need to be replaced becomes easier. Automatically detecting rows of trees or crops allow the farmer to quickly establish where there are issues with regards to yield and fertility.

The proposed, two-step algorithm that is implemented is named the Multi-objective Depth First Search (MODFS). This proposed solution will identify orchard rows by performing a graph search on the possible combinations of rows which are manufactured from available tree nodes. The first phase comprises of building confident short rows. The second phase extends these short rows by using a depth first search to identify good candidate rows to propagate into and continuously searches for new potential rows.

The main goal of this work is to identify underlying tree row structures in an orchard given the data is obtained from an aerial drone. The expected output is a collection of predicted rows made up of the given tree objects in each orchard. In current literature, crop row detection has focused on identifying rows from pixel data, whereas the objective of this work is to find rows from a set of points, with each tree centre representing a point. Due to this, not

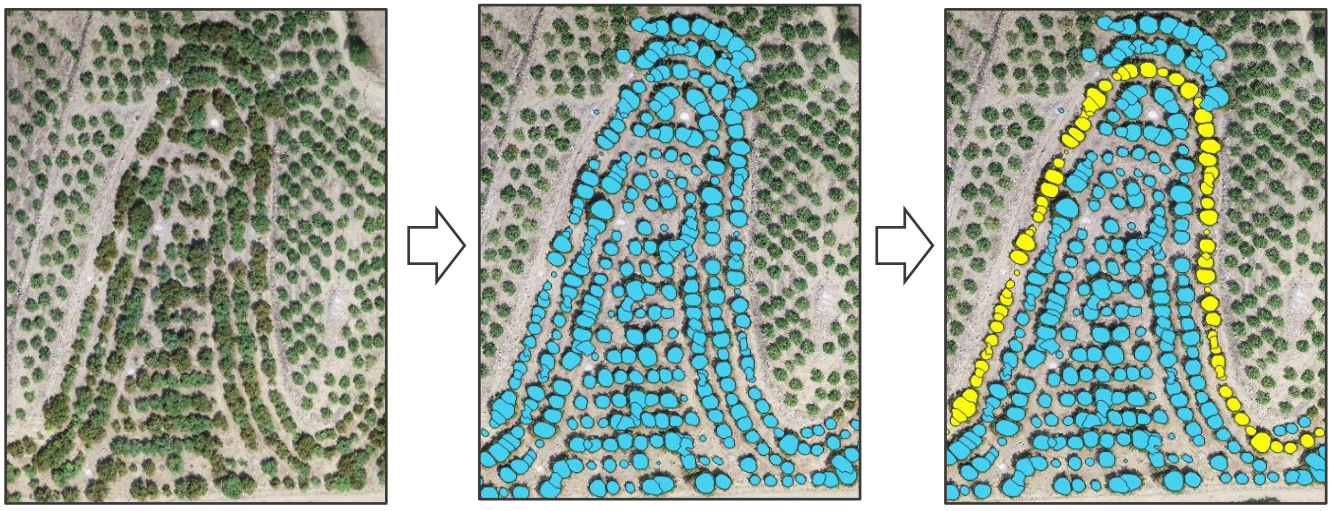


Figure 1: An example of the input data of a curved orchard and the identified row. The first image is the raw aerial image of the orchard. The middle image contains the tree polygon objects. The right image shows an orchard row in yellow.

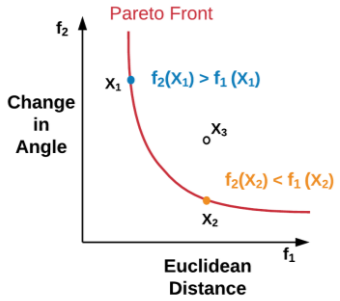
much research has been conducted within this niche. This leaves a large amount of room to work with and produce novel solutions. Various pathfinding solutions only work with a single predefined start node which, for the proposed solution, it would need to be expanded upon to have a set of multiple possible start and end nodes. When identifying the rows, two main issues arise which increase the complexity of the problem. The first being that the orchard rows are planted in a combination of curved and straight-lines, with curved rows being particularly difficult to identify. The second problem being that the abundance of poorly classified trees leads to variations in tree size and positioning across the orchard.

The MODFS algorithm aims to produce robust predictions while accounting for curved rows and poorly classified tree objects. The accuracy of the approach will be measured using Precision and Recall [4]. An auxiliary aim is to produce practical orchard rows with respect to the three previously mentioned accuracy metrics. These aims will ultimately conclude whether the use of a MODFS algorithm be can be implemented in the agricultural industry in their pursuit of a sustainable farming ecosystem.

This report contains 5 sections. Section 2 covers the basic domain knowledge for the MODFS and provides an overview of the previous and related work. In Sections 3, a detailed description of the proposed method, evaluation metrics and project components are explained. In Section 4, an in-depth analysis of the results is produced and reasons for these results are discussed. In Section 5, the final conclusions are provided and possible extensions for future work are discussed.

2. Background and Related Work

The goal for this multi-objective optimisation of orchard rows is to minimize two objective functions that relate to the Euclidean distance and average change in angle within a row. This translates to identifying the collection of non-dominated solutions on the Pareto Front [5]. In Figure 4, this would be the identification of X1 and X2 on the Pareto front – where these represent orchard rows which are non-dominated solutions.



f1 represents the Euclidean distance between nodes.

f2 represents the change in angle.

X3 is a dominated solution

Figure 2: Pareto Optimality

A potential solution to the orchard problem is pathfinding algorithms, which find the shortest path between a source and a destination node This shortest route could represent a row. The two classic pathfinding algorithms are Dijkstra [6] and A* [7]. These algorithms work with graphs that have either weighted edges or weighted nodes. Another possible solution is the class of evolutionary algorithms which are meta-heuristic search algorithms [8] that search the solution domain independently of problem-specific details and do so very quickly. In addition, crop row detection using image pixel date could be a potential solution.

The following related work section covers techniques that solve combinatorial optimisation problems. The three main algorithms reviewed are pathfinding techniques, multi-objective evolutionary algorithms, and image analysis methods for crop row detection.

2.1 Pathfinding

Ferguson and Stentz proposed an interpolated-based path planning algorithm, called Field D* [9], that works with uniform and non-uniform sets of grids. This any-angle method calculates lower-cost paths than other path planning algorithms because transitions are allowed between two points on adjacent grid cell edges, instead of using only grid corners or cell centres to transverse. Another method which produces paths with fewer turns was proposed by Xu and co-workers, called Link* [10]. Link* is a heuristic search pathfinding algorithm that can identify any-angle paths with fewer turns on a grid map. Both these methods have some the advantages of an A* algorithm but try to enforce smoother paths between goal states. In terms of the row identification problem, both of these algorithms would encourage smooth orchard rows to be identified since they both possess the any-angle path property. When searching between tree nodes for a potential path, any ridged rows would not be correct since the rows wouldn't follow the natural contours of the orchard. This is a requirement as the proposed algorithm needs to capture both straight and curved rows while minimising the change in angle objective function. Both methods require that each orchard is converted to a grid to allow for transversal to identify orchard rows. One issue with changing the orchard row problem into a large grid is that a normal grid would consume too much memory and is infeasible.

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 [11] (Pareto Front Iterative Deepening Multi-Objective A*). Each iteration uses a new set of threshold non-dominated vectors for the search and this method uses iterative deepening. The idea of using a threshold to select solutions which relax every iteration seems like an intuitive idea. The changing threshold will allow for more confident rows to be identified first, before developing the uncertain rows on top of the pre-existing confident rows.

Comba and co-workers [12] proposed an unsupervised algorithm for vineyard detection using 3D point-cloud mappings that are created from UAV imagery. The method begins an evaluation of features at each point in the cloud, using a small local radius of 5m, by using the local terrain surface and height evaluation. This algorithm shows robustness for curvilinear rows which is a key characteristic needed for our algorithm. The local circle evaluation techniques mentioned above is an excellent idea to capture local information within an orchard without being dependent on a global orchard row direction. The local circle evaluation may boost the chance for the identification of difficult curved rows while still permitting straight rows to be captured within the same orchard.

2.2 Evolutionary algorithms

Baron [13] proposed the use of Genetic Algorithms for line extraction. This algorithm uses two free parameters to allow for the extraction of straight-lines in the 2D plane. This result could be used to identify straight-lines in tree orchard examples with many overlapping polygon objects due to robustness to outliers. Xu and Tie [14] proposed a method for straight-line detection based on the particle swarm optimization (PSO) algorithm. Each particle in this algorithm represents a straight-line initialized between two edges points. The fitness function of the PSO algorithm is the number of edge points (nodes) that lie on the line for any instance of a particle. The PSO and GA are both time-efficient methods which produce accurate results. These two algorithms could provide simple and accurate initial row detections which the MODFS could leverage off to produce accurate rows.

2.3 Image Analysis

Hough Transform (HT) methods, initially proposed by Hough [15], are industry-standard algorithms for finding shape primitives. Straight-line detection is the simplest case of these methods and is

based on computing accumulations of counts corresponding to pixels that lie in straight-lines. The potential of using the HT methods stems from the fact that the tree centroids could represent these points, and a straight-line extracted could indicate a row in the orchard. There are several academic papers which apply the HT methods to crop row problems with reasonable success [16-18]. Unfortunately, the HT is limited in that it can only detect crop rows which are planted in a straight-line. While various techniques like approximating curves with many small tiles which contain straight-lines [19] and predefining a shape to try and identify in the source image [20] do exist, the HT is not flexible enough to robustly handle straight and curved row detections within our problem. The idea of breaking curves up into smaller segments to classify as straight-lines initially seem like a good way to deal with difficult sharp curves within an orchard. This will enable many short rows to be merged together to form an integrated curve that can be improved up at a post-processing stage. This idea is what inspired the initial straight-line component in the MODFS algorithm.

There are several methods in the literature for crop row detection which focus on the pixels of an image and not a graph theoretic approach. These methods use input images from various sources as input to their solution mechanism. The techniques usually differ according to their detection principle, namely, Hough Transform, Linear Regression [21, 22], Horizontal strips [23, 24], blob analysis [23, 25], stereo vision [26, 27], vanishing point [28, 29], and filtering [30, 31]. Unfortunately, all of the above-mentioned methods solve the crop row detection problem while using an approach which requires the source images from non-aerial source images (ground-based camera equipment), or do not transform the source image into a graphing problem to be analysed independently of the source image. Due to the previously mentioned reasons, these methods cannot be used to solve the orchard row identification problem.

3. Design and Implementation

3.1 Dataset

The data used in this work is unlabelled raw data which can be used, viewed and manipulated using QGIS[1]. Each orchard input represents information collected via an aerial image. The first file contains polygon shape detections, which represent the trees, this can be seen in the middle image in **Figure 1**. Additionally, each tree has an associated confidence value which indicates how certain the object detector is of the tree's presence. The second file indicates the boundary of the orchard on which row detection is being performed. The third is a georeferenced image of the actual orchard which can be seen in left image in **Figure 1**.

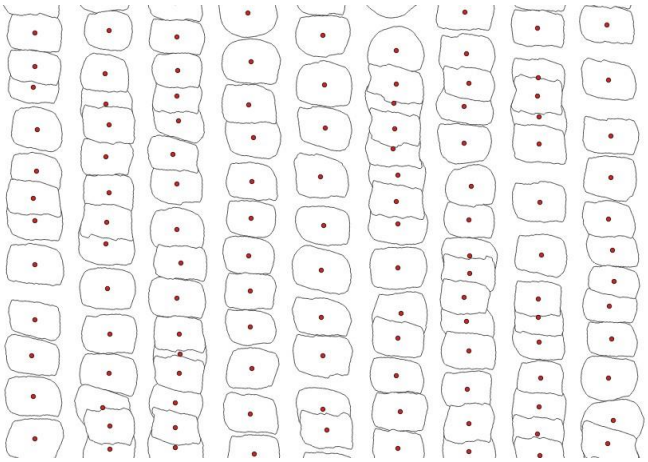


Figure 3: Red dots indicating the centroids of the polygon tree objects.

The orchards have a significant variation in row structures, tree structures, inter-row widths and number of trees. Since the smallest orchard from the provided datasets has 2326 trees, a subset of the points has been extracted from three of the orchard instances to allow for faster training. The extracted orchards are selected to ensure certain edge cases are contained within the subsets. Additionally, three full orchard instances have been hand-labelled to contain ground-truth solutions. The ground-truth solutions are hand draw lines which are connected to the centroids of the trees which subjectively are the underlying real rows within the orchard. These ground-truth rows will enable accuracy testing to be performed on the predicted rows. Following this, the precision and recall can be extracted to enable a quantitative metric to test results. Both precision and recall are measures which indicate the similarity of the predicted solutions to ground-truth solutions.

Table 1, in the appendix, shows all the metadata per orchard instance. Each orchard has a description, along with the number of trees, average tree confidence, standard deviation of tree confidence, average orientation in degrees, average and standard deviation of inter-tree distance, and orchard difficult.

3.2 Data Structures

The two main objects used throughout the MODFS implementations are the Tree and Row objects. The Tree object encapsulates the data for the individual trees. The Tree object variables indicate its position, ID, confidence, local orientation and whether it is part of a row. The Row object is made up of numerous Tree objects. The Tree objects keep track of the trees which have been added to the row, the average orientation of the row and the first and last tree within the row. Additionally, this object also keeps track of its unique row ID, and the current objective values.

While implementing the MODFS algorithm, the current list of Row objects and Tree objects are always stored in arrays. These arrays hold the collections of objects which are individually retrieved when being used for calculations. Additionally, these lists may also act as stacks and queues but they are still just variants of arrays. The only time this storage mechanism is not used is within the DFS method. A collection of potential rows is stored within a dictionary. The key for each row is the ID of the last tree within the current row. The dictionary holds the Row object and the three objective values. This dictionary is then sorted according to the objective value and the best row is selected. Below is an example of how the trees are stored.

{648: {"Row": 651,711,715,648, "Angle": 0.67, "Distance": 0.33, "Weighted": 0.45}}

In the example above, the last tree ID of the row is 648. The ID's of the trees which make up the row are 651,711,715,648, respectively. The Angle, Distance and Weighted objective function scores for this row are 0.67, 0.33 and 0.45 respectively.

3.3 Objective Functions

A weighted objective function linearly weights each objective so that a combined objective function can be optimized like a single objective function.

Objective Function	Equation
Distance	$\frac{\sum_{i=1}^{N-1} \sqrt{(\mathbf{x}_{i-1} - \mathbf{x}_i)^2 + (\mathbf{y}_{i-1} - \mathbf{y}_i)^2}}{(N-1) * 2}$
Change in Angle	$\frac{\sum_{i=1}^{N-1} \mathbf{Diff}(\boldsymbol{\theta}_{Average}, \boldsymbol{\theta}_i) * \alpha}{180}$ Where $\boldsymbol{\theta}_i = \boldsymbol{arctan2}(\frac{\mathbf{y}_2 - \mathbf{y}_1}{\mathbf{x}_2 - \mathbf{x}_1})$

Distance Objective Function: N is the number of trees in a row. The Distance objective is divided by 2(N-1) to give rows which have

[1] https://qgis.org/en/site/

more trees a smaller score – this encourages longer row to be selected over shorter rows.

Angle Objective Function: **Diff** calculates the angle difference between $\boldsymbol{\theta}_{Average}$ and $\boldsymbol{\theta}_{i}$ - then scales that value to be between 0 and 180 degrees. $\boldsymbol{\theta}_{Average}$ is the average change in the angle of all the trees with the current row. $\boldsymbol{\theta}_{i}$ is the acute angle between two adjacent tree objects within the current row.

The angle objective is multiplied by a penalty term α to discourage rows that have any connections (between any two adjacent trees in a row) which result in an angle that is a very different orientation to the rest of the row. This will result in an increase of the score for rows with a bad connection and decreases the chances of these rows being selected by the MODFS algorithm. The two cases of alpha are:

$$\alpha = \begin{cases} \mathbf{Diff}(\boldsymbol{\theta}_{Average}, \boldsymbol{\theta}_{i}), & > threshold \\ 1 , & \leq threshold \end{cases}$$

Which means when a connection angle is greater than the threshold, the numerator is squared. Otherwise, no penalty term is added.

When calculating the weighted objective function, the angle and distance metric are individually scaled between zero and 1 so that each value has an equal input into the weighted objective function.

3.4 Weighted Multi-objective DFS

The weighted a *priori* objective function will be implemented over a *posterior* multi-objective function owing to the clever assumptions that can be deduced from the orchard instances metadata. To be able to identify orchard rows for different orchards, a *posterior* multi-objective method would be a good unbiased selector since no objectives are favoured. However, when a *priori* assumptions are presumed, a weighted objective function would produce accurate predictions with less computational time and space requirements. Due to this, a *priori* weighted objective function is implemented over a *posterior* multi-objective approach. This information, which is used to make a *priori* assumption, includes the mean and the variation of the inter-tree distance, the number of trees within an orchard, and the variation of the angles within an orchard. Unfortunately, if the incorrect objective weights are assigned, poor predictive results will follow. A consequence of using a weighted objective function is the requirement of scaled objective functions, otherwise, the weighted coefficient will lead to a biased sampling from the solutions on the Pareto Front [5].

The standard deviation of angles and inter-tree distance metrics are the biggest factors in determining the weighting. This assumption leverages the fact that orchards with a smaller variation of angles are more likely to be straight, uniform orchards which have a general orientation – meaning that the angle orientation objective function should be favoured. If a large standard deviation occurs, the orchard is not uniform and probably has curved rows or is densely saturated.

The mean inter-tree distance and standard deviation of these distances also offer information about the orchard. The coefficient of variation (CV) is calculated to measure the dispersion of data around the mean[32]. A large CV represents trees which are erratically dispersed around the orchard – this suggests that the orchard is either curved or has densely distributed trees. The non-uniform orchard is best handled with a larger angle threshold and a smaller initial tree step size.

3.5 Constraints

The angle objective function proved to be challenging to govern since normal arithmetic can't be used to calculate angle mean and variation. The method identified to calculate the mean and standard deviation make use of conversions to the unit circle and polar coordinates. Following this result, all angles were constrained to lie within the first and second quadrants (0-180 degrees). This constraint still allows for bi-directional interpretation of row orientation. However, the main drawback of this decision is that short new connections occur on a predicted row when an edge is made in the opposite direction to the trajectory of the row. This leads to some rows having the last connection which goes in the opposite direction to its trajectory. This occurs when checking the check values for the angle constraint before making a new connection. Since the angles are always converted to be within the first and second quadrant, a bi-directional row growth is allowed in both directions. This issue can be alleviated with a check of the angles within a row.

Throughout the algorithm, new edges between trees are always checked and only built if they satisfy the angle constraint. The angle constraint states that the difference between two reference angles must be smaller than the angle threshold between the two check values. The first check value is always the angle between the selected tree and the potential neighbour tree. The second check value can either be a specific tree object's local angle metric or the average angle of a row object. If the difference between the check values is less than the angle threshold, the new edge is created, otherwise, the edge is not deemed a good connection and it is skipped.

Similar to the angle constraint, new edges are never created when the Euclidean distance between the two is greater than the global mean inter-tree distance. This prevents rows being created which may transverse over multiple pre-existing rows or simply edges that are too far apart.

Angle Objective Function - Local and Global Angle Measure

In a uniform orchard instance, the edges between trees which are similar in direction to the global orchard orientation are potential rows. However, a global angle can only be used with orchards which contain only straight, uniform orchard rows. In the case of an orchard with curved rows, the global angle will be a generalization of the entire orchard and will force the algorithm to incorrectly penalize the curved rows. In the agricultural industry, curved orchards are common and developing an algorithm which cannot compensate for this minor issue would be futile. The use of a local tree angle is preferred when deciding a trees best orientation. The local tree angle is a variable which can determine the orientation of a pocket of trees within an orchard containing straight and curved rows. The average angle between all of the tree objects within a small neighbourhood of the selected tree object is used to calculate the local angle. This assures that angle information pertains only to a small section of the orchard without being dependent on the general direction of the entire orchard.

3.6 Software Implementation

Python 3 was used to implement the MODFS algorithm. Python has proven to be a good choice with its easy development syntax, efficient list manipulation, as well as diverse and powerful built-in libraries which have made the development process easier. Other libraries used for this work include Matplotlib for graphing data, GDAL for manipulating geospatial raster data, Shapely for manipulating geometry objects, and NumPy for quick scientific calculations.

An Object Orientated approach was used to ensure all the code is modular and can be updated easily in the future. Each main algorithm is developed as an individual method to allow for code reuse and good overall project structure. The code has been well documented and this will ensure it is maintainable to any alternative developers who would like to work on this project.

3.7 Orchard detection algorithm

The MODFS algorithm can be split up into two main components. Firstly, the straight-line detection algorithm implements a simple

Nearest Neighbours search for points that are close together and forms short rows according to a simple angle constraint. The second component is the Multi-Objective Depth First Search which takes the rows from the previous component and expands these rows to produce longer rows. These components are explained in section 3.7.2 and 3.7.3 respectively.

3.7.1 File I/O and Tree Object Initialization

For each orchard, the polygon shapes are extracted from the raw detection geoJSON file. These values are then converted to the correct coordinate reference system (CRS) using the pyproj library which is a coordinate transformation library. The centroids and confidence values of each of the polygon shapes are then used to initialize Tree objects which are stored in a global array of Tree objects. The local angle of each tree object is also calculated and stored within the objects.

3.7.2 Straight-Line Algorithm

The straight-line algorithm produces many short confident orchard rows. This algorithm optimises results according to a distance function only, with an added angle constraint. A subprocess of this algorithm is the file I/O and the initializing of the tree objects.

A K Nearest Neighbours (KNN) method has been implemented since it is intuitive, easier to implement and more ground-up approach rather than implementing a random search PSO algorithm. The PSO algorithm would require more time to implement compared to the out-of-the-box logical KNN approach from the Scikit-learn package[2]. This KNN method allows for the tree centroid data to be fitted using Euclidean distance. This ensures a quick search of the local neighbourhood around a selected tree object for N trees. The returned tree centroids indicate the closest tree, as well as the distance between the two trees. This is done in O(N log(N)) time complexity with linear space requirements.

The algorithm has 3 parameters which are Number of Rows to grow (N), Depth per row (D), and max angle (Theta). N is set to the number of trees in the orchard multiplied by 75%. This is to ensure a large proportion of tree objects are added to row objects but not forcing too many bad connections to be added.

The output of the straight-line algorithm is then checked by a post-process cleaning algorithm. This check runs through each edge made on each of the rows and flags edges that are not within the angle constraints of the local average angles of the neighbouring trees. The rows with drastically changing angle are then removed entirely from the orchard predicted rows.

Algorithm Straight-Line Detection

```
procedure Straight Line Detection ( N, D, Theta)
    Initialize Tree Objects
    Initialize Solutions Array
    for number of trees N do
        Random_Tree = Select Random Tree
        for depth of row do
            Neighbours = Get KNN Neighbours of Random_Tree
            for neighbour in Neighbours do
                Edge = connection between Random_Tree and neighbour
                if Edge meets constraints
                    Make new row with Random_Tree and neighbour
                    Random_Tree = neighbour
                    Add New Row to Solutions
                    Break
            end
        end
    end
    Return Solutions
End
```

3.7.3 Multi-objective Depth First Search Algorithm

The initial straight row detection creates easily identifiable straight rows as can be seen in **Figure 4**. Each row is a collection of various assigned tree objects where the row is represented as the red dotted line. This algorithm is initiated from the two endpoints of the previously created straight rows and will propagate outwards to the potential candidate trees. These candidates will be selected using the K-Nearest Neighbours algorithm with Euclidean distance as its metric – this is equivalent to a Breadth First Search for finding the initial neighbouring nodes to search into. All candidate nodes and sub-nodes are chosen only if they are within the maximum distance threshold and maximum angle change threshold from the associated straight row. In **Figure 4** on the left image, the yellow lines with arrows heads show potential extensions, while the blue line with an arrowhead indicates a row that did not meet the minimum constraints. When the node propagates, it will search from the source into each of the candidate neighbour nodes (green trees) and it will add each potential sub-node to the stack until a specified depth, D, is reached. D is a parameter that will be evaluated to find the optimal depth, but this will likely change depending on the orchard instance.

While traversing the different sub-nodes, the algorithm will add the potential row (which is an extension of the original row with the sub-node added as an additional tree) to a collection of the selected rows. Once the depth has been reached, the best row will be selected according to the lowest weighted objective value of all the rows which are in the solution set. This solution selection is shown in **Figure 4** on the right image by the red dotted line. This new row will be added to the predicted orchard rows, and the base row which it was created from will be removed. This process will be repeated for a specified number of times to increase the number of rows. This Depth First Search algorithm is run on all the current rows in the orchard and will either keep the same row or just extend the base row to include more trees within the row. A priority queue is created using the rows, the queue is then sorted by the weighed objective in ascending order. The best rows are grown first to encourage

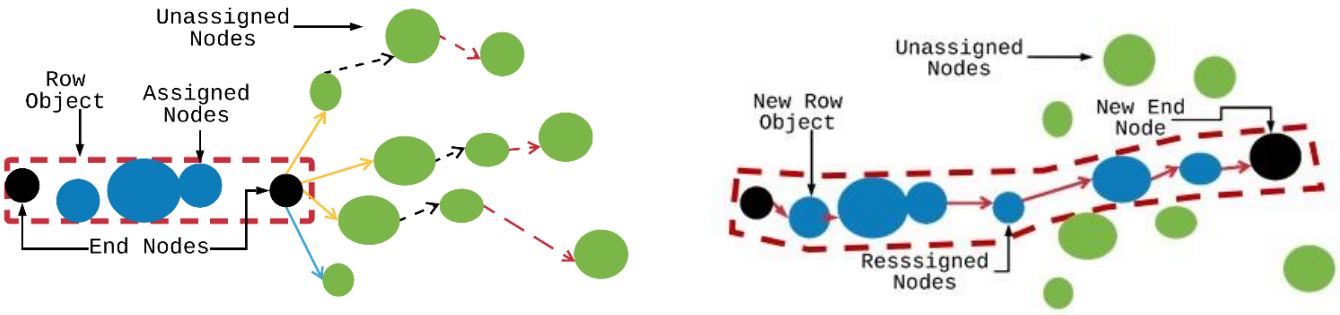


Figure 4: Shows the MODFS Tree Selection on the left. The picture on the right shows the row expansion into the selected best row. Black nodes represent row endpoints. Blue and green nodes represent assigned and unassigned nodes respectively. The red dotted line represents an orchard row.

[2] https://scikit-learn.org/stable/modules/neighbors.html

elongating already confident rows but still allowing the other rows to expand.

The remaining trees that haven't been allocated to any rows are then stitched together using a stitching algorithm. Firstly, the algorithm makes some additional short rows between free trees that satisfy the angle and distance constraints. Secondly, it runs a row stitching algorithm that collects all the row endpoints (first and last trees within a row) from all the rows in the orchard and attempts to merge these points similarly to the straight-line algorithm. If two endpoints are selected to be stitched, those two rows will be compared to the potentially merged row. If the potential row is better than both rows, it will be added to the orchard and the other two rows will be discarded. If not, nothing will be changed. These two methods are run a number of times with relaxing constraints to ensure that the majority of the free points are allocated to rows if they are within the angle and distance thresholds.

Algorithm Depth First Search

```
20: procedure MODFS ( Row, D, Theta )
21:     Initialize empty Stack
22:     Solutions = Empty Dictionary
23:     Add initial Row to Stack
24:     While Stack not empty do
25:         R = Stack.pop()
26:         for neighbour in getNeighbours( R )
27:             Temp_Row = add neighbour tree to row R
28:             if Temp_Row meets constraints
29:                 Add Temp_Row to Solutions
30:                 Add Temp_Row to Stack
31:         end
32:         If Iterations > Depth do
33:             Break
34:     end
35:     Return Solutions
36:
37: End
```

The algorithm has three parameters which are the Initial Row to initiate the DFS from (Row), Depth to search for rows (D), and Max Angle (Theta). After the MODFS method, a post-process method is applied similar to the method applied after the straight-line algorithm.

Moving Window Feature

An orientation-driven feature that is used throughout the MODFS algorithm is the row's moving window angle. When previously checking if an additional tree should be added onto a pre-existing row – the angles between the tree and row's endpoint tree is compared to the pre-existing row's average angle. This check justifies whether the tree is aligned with the current average direction of the row. Unfortunately, this check will fail if a straight row starts to bend in any direction. To alleviate this issue, the pre-existing rows average angle will only be calculated by the last 6 trees (parameter which can be changed) and not the entire row. This feature allows for rows to gradually curve without the being flagged due to its flexible nature. Increasing the window size will ensure only gentle curves will be accepted. Decreasing the window size allows for more erratic curves to be accepted by the MODFS.

Confident Orchard Trees

Each tree object has a confidence value between zero and one. This parameter indicates the level of confidence that the tree object actually exists at that location. When finding the initial straight-rows of the orchard, the confidence parameter was integrated into the search for rows. The unconfident trees which had low confidence values were not included with the subset of trees passed to the straight-line algorithm. The consequence is that fewer trees are available for selection, therefore, fewer row combinations are produced by the algorithm. However, more confidence rows are produced since they are formed from a higher quality batch of trees. The threshold for trees to be included to the subset is the mean minus one standard deviation of the confidence of that orchard.

Parameter Tuning

Parameter tuning was performed over several data sets and results were analysed per parameter against the orchard score. The most influential parameters that were tuned are:

- K Nearest Neighbours Number: K closest trees
  - Curved and Dense orchards prefer smaller K sizes
- Depth of trees for straight-line and MODFS algorithm
  - Curved and Dense orchards prefer smaller Depth
- Weighted objective percentages
  - Straight uniform orchards prefer higher angle weighted objective functions
- Average moving window size for row angle
  - Curved orchards preferred smaller moving window sizes

Through the parameter tuning of the orchards, general parameters that suit all the orchard types are unidentifiable. The curved orchards had the most trouble with finding optimal parameters. The parameters which had the best general impact were implemented.

3.8 Evaluation

The main qualitative methodology of evaluation will be to compare the rows produced visually by the MODFS approach to the unlabelled version of the orchard rows. Each orchard solution will be judged in each category. These categories serve to show the different aspects of the algorithm and where it may fail.
The qualitative score options are very good, good, bad, very bad or average. The categories are:

- How well the overall row structure is captured
- How robustly the perimeter trees are handled
- Predicted Row Length to orchard data
- Frequency of incorrect classifications
- Frequency of missing classifications

The three testing orchards, 32377, 36502, 36507, have been used to perform a comprehensive statistical analysis of the results due to the availability of hand-labelled ground-truth rows. The ground-truth orchard can be found in the appendix. After each execution for these testing orchards, the predicted rows are accompanied by an accuracy and precision statistic. For both testing and normal orchards, an orchard score is produced as a metric for how accurate the algorithm worked on that particular orchard. The orchard score is a combination of the mean weighted objective function value and number of unassigned trees. The goal is to minimize the orchard score value as each of the parameters which combine to make it need to be minimized.

4. Results and Discussion

4.1 Straight-Line Algorithm

In **Figure 5**, the small straight rows do an excellent job at capturing the overall orchards structure with a few exceptions. The exceptions are either incorrectly orientated orchard rows or unassigned trees. The unassigned trees which haven't been allocated to any row were not selected by the straight-line algorithm since they were not within the constraints. These exceptions usually occur around the edges of the orchard where there aren't many points to connect to. After the straight-line algorithm has been run, a few errors appear in the orchard rows which can be seen as the red lines in **Figure 5**.

These incorrect rows have been identified by the post-process step and will be removed from the orchard. These rows are produced in less than 5% of all the edges which occur and are easily identifiable for uniform orchards.

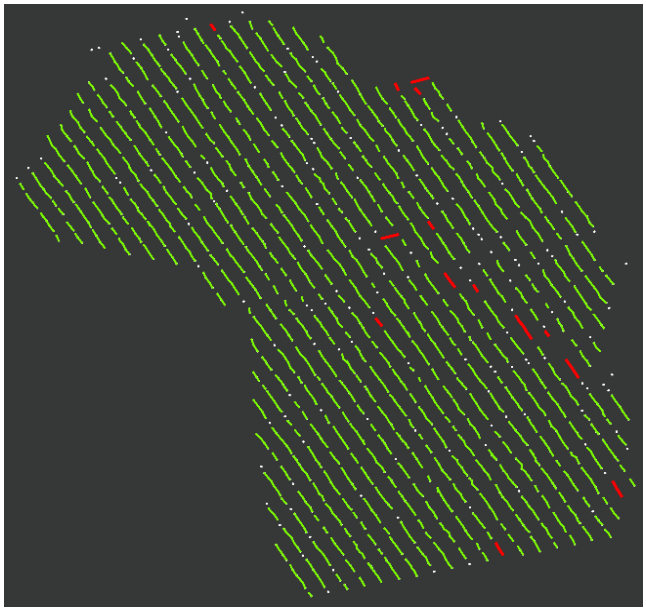


Figure 5: Straight-line algorithm applied to Orchard 36507. Green lines indicate the predicted short rows and red lines indicate rows that are removed by the post-processing method

Figure 6 shows two cases where the straight-line algorithm does not work correctly on curved rows. In the yellow square, there are accurate curved short rows which the post-processing step removes. This occurs because the post-process stage only checks an individual edge at once without considering the entire row. In the blue square, there are short predicted rows which do not follow the correct orientation kept within the orchard. The rows occur because there is a dense pocket of unstructured trees which obscures the general orientation information in that area of the orchard. The straight-line algorithm cannot extract an accurate local angle metric. This issue requires the MODFS algorithm to robustly identify the correct rows according to both angle and distance objective functions.

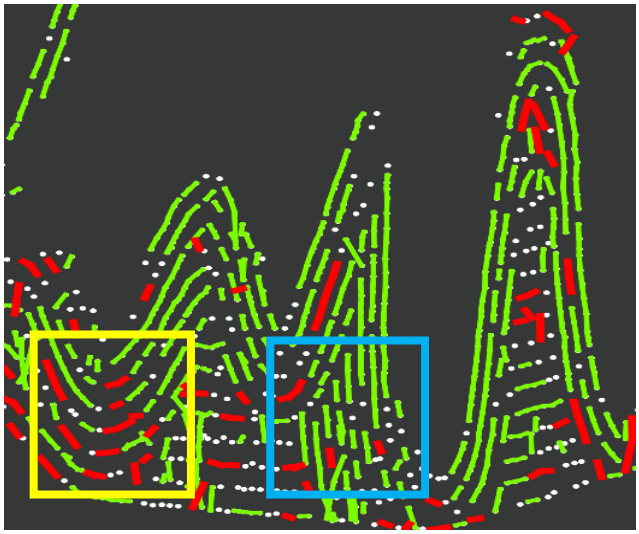


Figure 6: Straight-line algorithm applied to orchard 32377. Yellow and Blue square represent problem areas.

Figure 7 shows the output of the MODFS when it is run on the orchard 32377. The algorithm can be seen to partially capture the curved structure of the rows by breaking the arc up into smaller sections. On the left image in **Figure 7**, the gentle curve is captured accurately by a few line segments with the straight-line algorithm. In the right image, the sharp curves at the peak of the orchard have been somewhat replicated which shows an important idea behind the initial algorithm. Building on these results, the MODFS can now be introduced to integrate these line segments to produce long smooth rows.

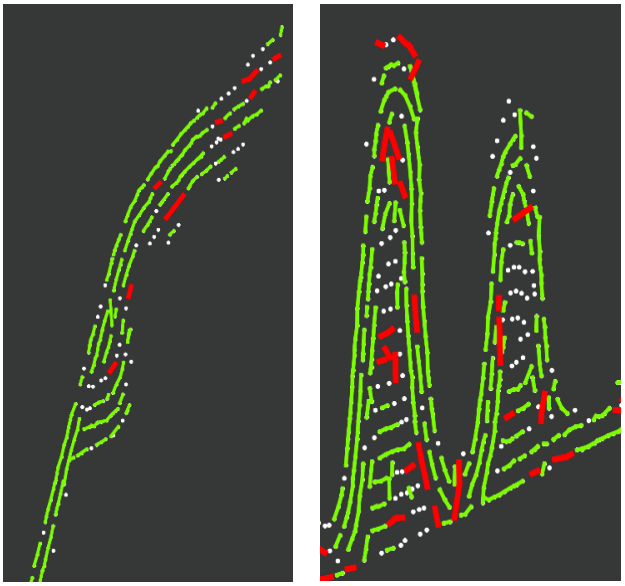


Figure 7: Straight-line algorithm applied to orchard 323577. Green lines indicate the predicted short rows and red lines indicate rows that are removed by the post-processing method.

4.2 Multi-objective Depth First Search

Figure 8 displays the final output of the MODFS algorithm. Notably, the majority of the predicted rows are impeccably orientated with no major angle inconsistencies and the rows span the length of the orchard. This is reiterated by a precision score of 73%. With the exception of some shorter rows and unassigned trees, the results show that the MODFS works exceptionally well on straight orchard rows. One minor issue which has arisen is the incoherent short row that is circled in red. This row goes against the general orientation of all the other rows. This issue arises with trees which are situated around the perimeter of the orchard – these trees have very few options to make a connection which leads to irrational connections.

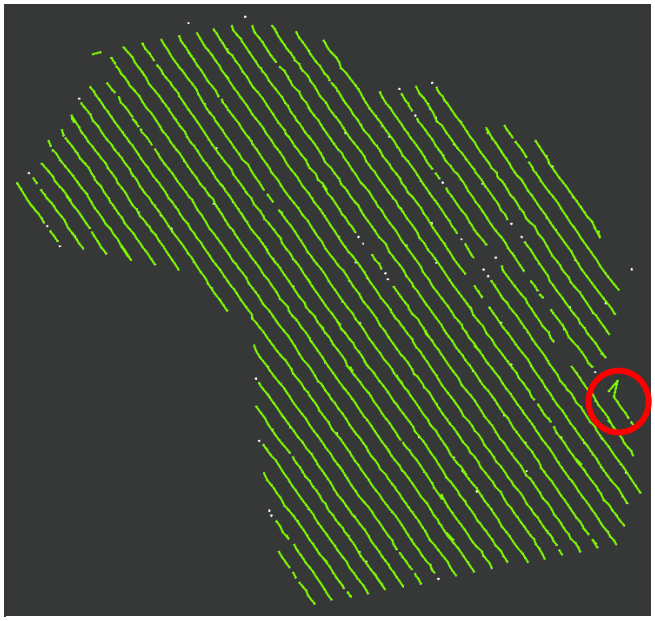


Figure 8: MODFS predicted orchard rows with orchard 36507. Precision of 73% and a Recall of 64%

Figure 9 depicts three images of orchard 32377 after the MODFS algorithm has been executed. Orchard 32277 is a challenging orchard as it has a mix of very sharp curves, with long straight rows, and dense pockets of trees. The algorithm struggles to predict meaningful orchard rows. The top image shows that areas of the orchard are plagued by inconsistent row predictions. There appears to be a cluster of points which restricts the effect of the local angle for identifying the local orientation. The bottom left picture shows

that when there are no points that are densely clustered, the flat curves are identified by the algorithm. The bottom right picture also shows this partially, but many misclassifications are mixed with good row classifications.

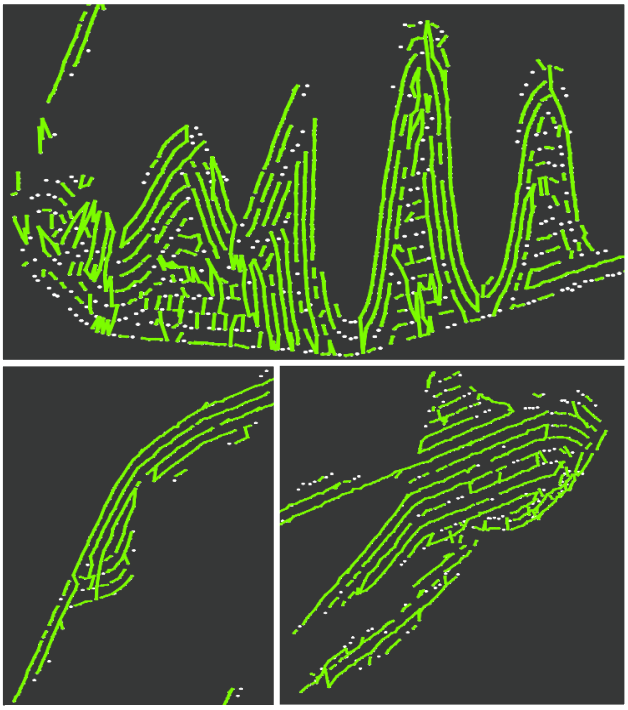


Figure 9: MODFS predicted orchard rows with orchard 32377.

Figure 10 shows an interesting case where a global angle approach would fail to produce good results. This orchard demonstrates the power of using the local angle approach. Both sub-orchards are well classified with few unassigned trees. The predicted rows are generally straight and long, and if not, the row is a culmination of accurate shorter rows. The algorithms inability to create rows that span the entire orchard is surpassed by the accuracy of shorter rows. A notable issue occurs in the bottom right corner of the orchard, where a parallel connection occurs. This issue only arises on the fringes as previously explained.

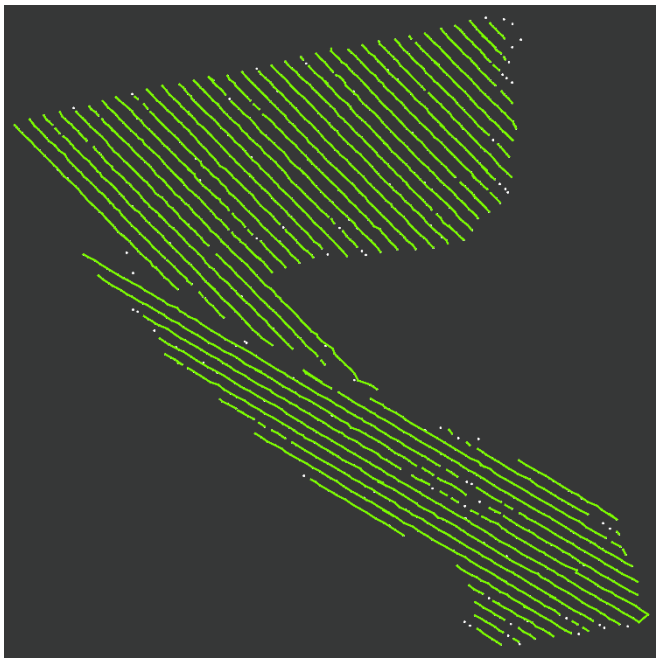


Figure 10: MODFS predicted orchard rows with orchard 36513.

Figure 11 illustrates the predicted rows on orchard 36502. The output shows countless misclassifications and unassigned trees. The slight change in angle and the dense rows create a challenging mix for the weighted objective function to select points which are in close proximity to the row, but also within the angle threshold. The rows are badly linked together near the point where the angle change occurs in the orchard. This is a by-product of numerous bad edges created around this point. The moving window does not help as it's a sharp change and not a gradual curve.

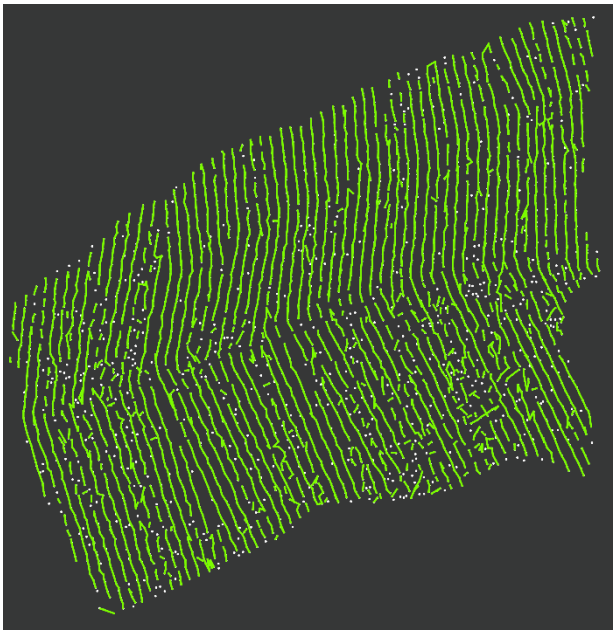


Figure 11: MODFS predicted orchard rows with orchard 36502 with 738 free trees

Figure 12 depicts an orchard with several straight uniform rows. This orchard required a very angle orientated objective function which manages to produce good rows. The main rows produced do match the length of the orchard and capture the overall structure, however, there are numerous outlying trees which have not been assigned to rows. The biggest issue is the number of unassigned trees which remain after the MODFS algorithm has been executed. This problem can be solved by relaxing the constraints to allow for more inaccurate rows to ensure there are fewer free trees. However, this mitigation strategy is never implemented because it will degrade the accuracy of the results. Since the main goal of this proposed algorithm is to produce accurate rows, it will not be implemented.

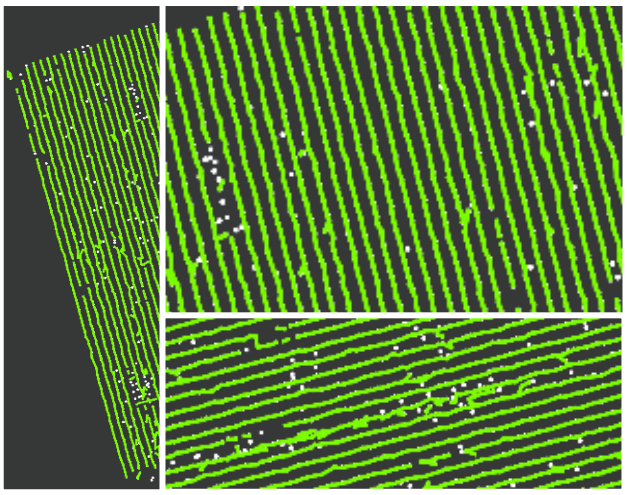


Figure 12: MODFS predicted orchard rows with orchard 43581

Figure 13 illustrates the distribution of precision and recall for the respective testing orchards. Each orchard is run 30 times for each precision and recall to produce these results. Orchard 36507 has superior precision and recall, which is expected due to its simple nature. Orchard 32377 has the lowest precision and recall which is between 50% and 55%. This shows the MODFS can be significantly

improved upon for heavily curved orchards. Orchard 36502 represents an orchard instance with straight and slightly curved rows. It has a precision and recall of 66% and 58% respectively. Focusing on precision, the MODFS provides sufficiently predicted rows which is an indicator that the algorithm has the potential to improve.

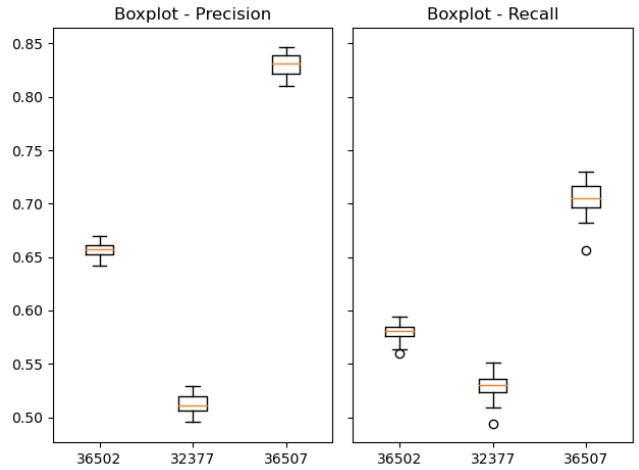


Figure 13: Boxplot showing the Precision and Recall for the three testing orchards

The execution time of the MODFS algorithm is dependent on the number of trees in an orchard and not the type of orchard. This can be seen in **Figure 14** as the average execution time of orchard 32377 and 36507 are close to 300 seconds, and these two orchards have just under 2500 trees each. Orchard 36502 has double that number and has an average execution time close to 700 seconds. This result and additional unseen testing reveal that the growth in execution with respect to the number of trees has exponential growth. This exponential growth is however limited due to the constraints forced upon the MODFS search.

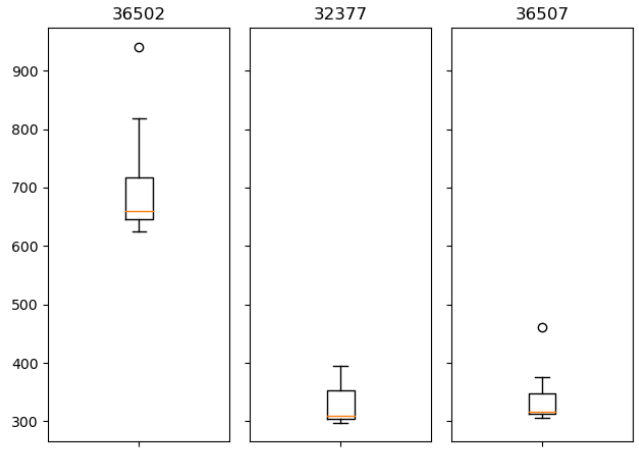


Figure 14: Boxplot showing the Execution Time in Seconds for the three testing orchards

4.3 General Observations

The properties of the MODFS algorithm have been covered about above. The following discussion will focus on the prominent pitfalls of the algorithm. The MODFS algorithm is unable to capture a number of unique orchard cases. These failure cases are the weak points in the algorithm. Identifying and devising potential mitigation strategies for these matters will be discussed.

In most orchards, farmers require a pathway to move within the orchard which provides access to the crops. These service tunnels that run through an orchard naturally lead to more perimeter trees. When service tunnels are present, the rows produced within an orchard will inherently be shorter which leads to less desirable inaccurate rows. These rows are less desirable as the longer the row is, the more assuredly it will capture trees along its trajectory knowing that after each tree the row collects, the higher potential it has for being a confident row. This higher potential is measured by the objective functions being inversely proportional to row size. These perimeter trees can also be seen on the peripheral of all the orchard instances. When trying to make connections between trees, a tree within the orchard will have many more options compared to a tree which is situated on the outskirts of the orchard due to its isolated location. This is because, within the orchard, a tree is surrounded on all side with connection options, but a tree on the outskirt has significantly fewer connection choices. If all of the connection options are unavailable, these fringe tree nodes will remain unassigned trees. This is a better scenario than having undesirable inaccurate rows which would be the case of all trees were forced to join a row. Due to this, no mitigation strategy is implemented. Nonetheless, a good strategy would be to build rows from these perimeter nodes inwards to initially reduce the number of free trees around the border of the orchard and to accommodate these invasive service tunnels. This problem occurs frequently in every orchard and within orchards that have service tunnels.

Within an orchard, a dense cluster of shapeless trees reduces the certainty of the algorithm by not having a clear-cut row - an example can be seen in **Figure 6** within the blue square. The local angle parameter will not provide any insight as the points are unstructured inside that pocket and cannot find a truthful local angle. This encourages more misrepresentation of the angle objective function which assists in selecting rows. Simultaneously, the algorithm will select the closest trees that are following an incorrect orientation which leads to unstructured row predictions. The mitigation is to allow the algorithm to look at a wider range of tree objects in the pocket of dense trees. This can be done by increasing the KNN value to include trees outside the cluster of trees to infer better directional information.

In the top image of **Figure 9**, the predicted rows have not fully captured the peak of the curved rows. Instead, they have been broken up into smaller segmented rows which partially capture the arcs. The rows in **Figure 9** represent uncommon sharp curves within an orchard instance, nevertheless, these do need to be accounted for when implementing a robust identifier. The straight-line algorithm

Orchard Instance	Predicted Row Length to orchard data	How robustly the perimeter trees are handled	Frequency of missing classifications	Frequency of incorrect classifications	How well the overall row structure is captured
32377	Bad	Bad	Bad	Bad	Bad
35516	Bad	Bad	Bad	Bad	Bad
36502	Average	Average	Average	Average	Good
36507	Good	Good	Good	Good	Good
36513	Good	Good	Good	Good	Good
41630	Bad	Bad	Bad	Bad	Bad
43581	Good	Average	Average	Average	Good

Table 2: Summary of the results of the MODFS on each orchard instance

provided a good base to try to capture the curved rows, but the MODFS was not able to replicate the true rows. A potential strategy to improve this problem would be to identify the correct moving window parameter or add a trajectory parameter. The trajectory parameter would allow for sharp connections to be made if these connections are following the slope of the row. The angle constraint usually flags these sharp turns and removes these rows, but if the trajectory parameter is added, it will allow the MODFS to more accurately map curved orchard rows.

Another issue which appears in all of the results of the MODFS algorithm is the excessive number of unassigned trees within the orchard after execution. **Figure 15** shows the average number of unassigned trees after the execution of the algorithm per orchard. These trees have not been assigned rows since they are deemed to worsen the rows which they could possibly be added to. The percentage of free trees to total trees per orchard ranges from 10% to 20% of the total number of trees (see **Figure 15** and **Table 1**). This percentage seems to increase with the size of the dataset, as well as the occurrence of curved rows. The number of unassigned trees will linearly increase as the number of total trees increases. The reason being that as the number of starting trees increases, the number of unstructured trees usually increases too – adding more uncertainty. This leads to more free trees which aren't assigned to rows due to the additional noise these extra trees create for the algorithm. This result shows that the MODFS algorithm prioritises more accurate rows. In addition, this predetermined rule should encourage the algorithm to be more robust to bad tree object classification coming from the object detector phase of the project. This independence comes at a cost of not identifying a few rows which seem straight-forward to predict, but to satisfy the aim of the project a robust row identifier is preferred.

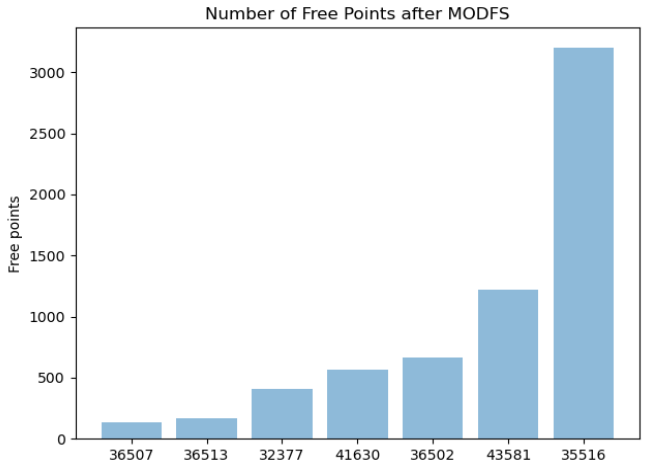


Figure 15: Bar Graph showing the number of unassigned trees after each orchard's execution

Experiment Improvements

- Additional data sets could be hand-labelled to allow for more statistical tests to be performed on more orchards using precision and recall.
- The labelling of the ground-labelled solutions may be biased since they are manufactured with what is deemed to be the best solution by the development team. Using a mutual party to do the labelling would create less bias.
- Additional parameter tuning could have been conducted for all the minute parameters which would only have a minor effect on the output. Nevertheless, with additional time, the same tests that were done for the main parameters would be performed on these auxiliary parameters.

In summary, The MODFS algorithm produces accurately predicted rows for straight uniformly orientated orchards while struggling with orchards that contain dense pockets of trees or sharp curves. The MODFS is adaptable to multi-orientated orchards and will produce robust rows without a fixed global orientation. The algorithm suffers from the inability to allocating all the trees to suitable rows, as it classifies many connections as unfeasible.

The results for each orchard are summaries in **Table 2**, which is on the previous page**.** The MODFS algorithm is run on each orchard and the results are evaluated with the qualitative metrics decided in section 3.8.

5. Conclusions

The MODFS algorithm was shown to produce sufficiently accurate orchard row prediction for straight and slightly curved orchards. Additionally, the algorithm can adapt to multi-orientated straight orchards and will produce accurate rows. Unfortunately, orchards which contain very dense or curved trees would result in inaccurate rows which do not capture the underlying row patterns. The rows produced on heavily curved orchards show potential because they partially map the arc of rows with a few short rows. Additional areas where the MODFS may be improved upon include; handling perimeter trees and service tunnels within the orchard, and reducing the number of unassigned trees. Nevertheless, a robust algorithm has been developed, implemented and proven to accurately predict orchard rows for numerous cases. This is shown by the numerous orchard predictions which have visually correct rows, as well as the qualitative metrics which boost statistical confidence in the MODFS's predictive power. The aims of the project have only been partially met; however, the project has been successfully informative on where the algorithm works correctly and indicated where it can be potentially enhanced.

Finally, it is our belief that the MODFS algorithm is not ready to be implemented in the agricultural industry. The algorithm is currently computationally feasible in terms of execution time and space required. The rows produced are acceptable for the simple orchard cases, meaning, implementation is feasible for a subset of orchard types. After some alternative feature implementations on the MODFS algorithm, our projection is that it will ready to be implemented to assist farmers in agricultural endeavours by the next iteration of the project.

Future Work

After the implementation of the MODFS, there are a few avenues which seem promising to further this project. Firstly, the alteration which could have the biggest impact on the results would be to implement a pure multi-objective algorithm instead of the weighted objective function. Secondly, a more robust straight-line algorithm could be used to capture better initial short rows. The potential implementations of this could come from the Genetic algorithm[13] or Particle Swarm Optimisation[14] algorithms discussed in the related work. Finally, the integration of height data into the MODFS algorithm would allow it to use contour lines to assist in predicting curved rows more robustly.

Acknowledgements

I would like to sincerely thank my Supervisor, Patrick Marais, for all his help, constant guidance, and encouragement throughout the project. In addition, I would like to thank my two team members, Tim Simon and Leonard Chuang, for their contributions to the Orchard project and all the support they provided. Finally, I would like to extend my appreciation to Aerobotics for supplying the problem and various datasets.

References

[1] Carr, G. *THE FUTURE OF AGRICULTURE*. City, 2016.
[2] FAO-FOOD and ORGANIZATION, A. *The state of food and agriculture*. Livestock in the balance Rome, City, 2009.
[3] Arnó Satorra, J., Martínez Casasnovas, J. A., Ribes Dasi, M. and Rosell Polo, J. R. J. S. J. o. A. R., , vol. 7, núm. 4, p. 779-790 Precision viticulture. Research topics, challenges and opportunities in site-specific vineyard management (2009).
[4] Tan, P.-N., Steinbach, M. and Kumar, V. *Introduction to data mining*. Pearson Education India, 2016.
[5] Miettinen, K. *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012.
[6] Dijkstra, E. W. J. N. m. A note on two problems in connexion with graphs, 1, 1 (1959), 269-271.
[7] Hart, P. E., Nilsson, N. J., Raphael, B. J. I. t. o. S. S. and Cybernetics A formal basis for the heuristic determination of minimum cost paths, 4, 2 (1968), 100-107.
[8] Raidl, G. R. and Puchinger, J. *Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization*. Springer, City, 2008.
[9] Ferguson, D. and Stentz, A. J. J. o. F. R. Using interpolation to improve path planning: The Field D* algorithm, 23, 2 (2006), 79-101.
[10] Xu, H., Shu, L. and Huang, M. *Planning paths with fewer turns on grid maps*. City, 2013.
[11] Coego, J., Mandow, L. and De La Cruz, J. P. *A new approach to iterative deepening multiobjective A*. Springer, City, 2009.
[12] Comba, L., Biglia, A., Aimonino, D. R., Gay, P. J. C. and Agriculture, E. i. Unsupervised detection of vineyards by 3D point-cloud UAV photogrammetry for precision agriculture, 155 (2018), 84-95.
[13] Baron, L. *Genetic algorithm for line extraction*. École polytechnique de Montréal, 1998.
[14] Xu, S., Tie, J. J. S. and Transducers Straight Line Detection Based on Particle Swarm Optimization, 160, 12 (2013), 653.
[15] Hough, P. V. *Method and means for recognizing complex patterns*. Google Patents, City, 1962.
[16] Ji, R., Qi, L. J. M. and Modelling, C. Crop-row detection algorithm based on Random Hough Transformation, 54, 3-4 (2011), 1016-1020.
[17] Leemans, V., Destain, M.-F. J. I. and Computing, V. Line cluster detection using a variant of the Hough transform for culture row localisation, 24, 5 (2006), 541-550.
[18] Åstrand, B. and Baerveldt, A.-J. J. M. A vision based row-following system for agricultural field machinery, 15, 2 (2005), 251-269.
[19] Soares, G. A., Abdala, D. D. and Escarpinati, M. *Plantation Rows Identification by Means of Image Tiling and Hough Transform*. City, 2018.
[20] Ballard, D. H. J. P. r. Generalizing the Hough transform to detect arbitrary shapes, 13, 2 (1981), 111-122.
[21] Hague, T., Tillett, N. and Wheeler, H. J. P. A. Automated crop and weed monitoring in widely spaced cereals, 7, 1 (2006), 21-32.
[22] Søgaard, H. T., Olsen, H. J. J. C. and agriculture, e. i. Determination of crop rows by image analysis without segmentation, 38, 2 (2003), 141-158.
[23] Fontaine, V. and Crowe, T. J. C. b. e. Development of line-detection algorithms for local positioning in densely seeded crops, 48 (2006), 7.
[24] Jiang, G., Wang, Z. and Liu, H. J. E. s. w. a. Automatic detection of crop rows based on multi-ROIs, 42, 5 (2015), 2429-2441.
[25] García-Santillán, I., Guerrero, J. M., Montalvo, M. and Pajares, G. J. P. A. Curved and straight crop row detection by accumulation of green pixels from images in maize fields, 19, 1 (2018), 18-41.
[26] Kise, M., Zhang, Q. and Más, F. R. J. B. e. A stereovision-based crop row detection method for tractor-automated guidance, 90, 4 (2005), 357-367.
[27] Kise, M. and Zhang, Q. J. B. E. Development of a stereovision sensing system for 3D crop row structure mapping and tractor guidance, 101, 2 (2008), 191-198.
[28] Pla, F., Sanchiz, J. M., Marchant, J. A., Brivot, R. J. I. and Computing, V. Building perspective models to guide a row crop navigation vehicle, 15, 6 (1997), 465-473.
[29] Jiang, G., Wang, X., Wang, Z., Liu, H. J. C. and Agriculture, E. i. Wheat rows detection at the early growth stage based on Hough transform and vanishing point, 123 (2016), 211-223.
[30] Hague, T. and Tillett, N. J. M. A bandpass filter-based approach to crop row location and tracking, 11, 1 (2001), 1-12.
[31] Bossu, J., Gée, C., Jones, G., Truchetet, F. J. c. and agriculture, e. i. Wavelet transform to discriminate between crop and weed in perspective agronomic images, 65, 1 (2009), 133-143.
[32] Dixon, W. J. and Massey Jr, F. J. Introduction to statistical analysis (1951).

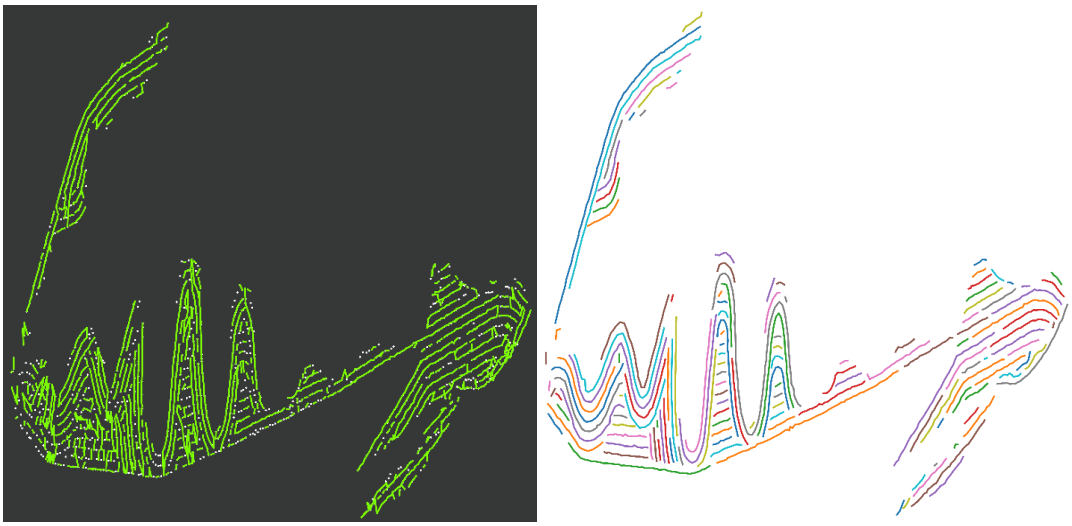
Appendix

Table 1:

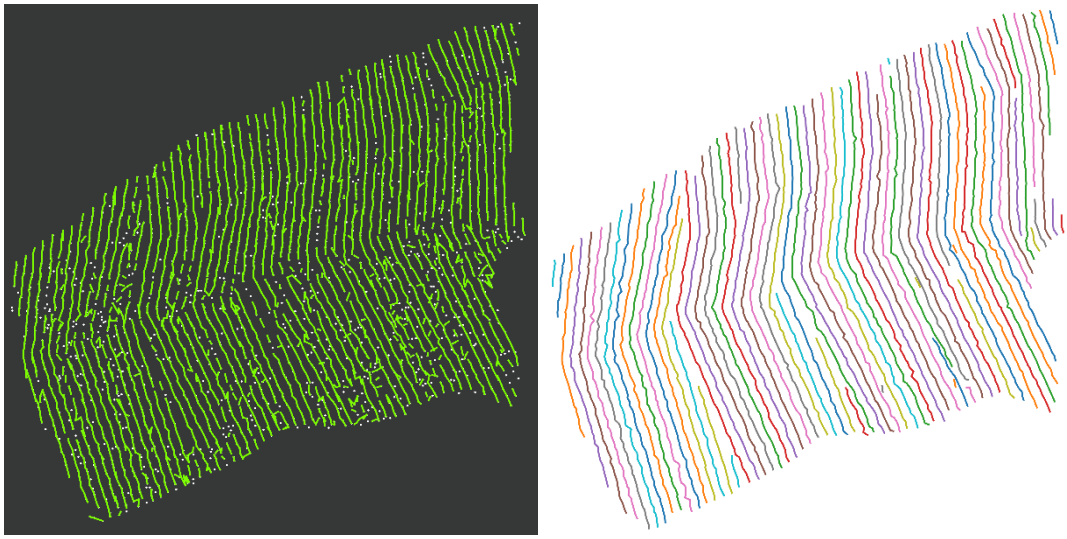
Orchard Instance	Number of classified Trees	Average Confidence	Standard Dev of Confidence	Standard Dev of Local Angles	Mean inter-tree Distance	Standard Dev of inter-tree Distance	Problem Difficulty	Description
32377	2581	0.745	0.336	56.339	10.42	3.63	Hard	Sharply curved rows
35516	14404	0.296	0.214	53.469	2.15	0.97	Hard	Dense uncertain trees
36502	4532	0.685	0.285	39.454	3.54	1.55	Hard	Dense rotating rows
36507	2326	0.894	0.060	20.849	3.65	1.63	Easy	Straight Confident Rows
36513	2435	0.862	0.142	18.901	3.70	1.89	Medium	Pair of overlapping straight orchard
41630	4038	0.573	0.280	35.715	3.19	1.08	Hard	Dense uncertain noisy rows
43581	10190	0.544	0.270	35.839	3.38	1.72	Medium	Dense straight rows with minor noise
36507_Test	235	0.884	0.060	15.223	3.59	1.67	Easy	Straight Confident Ros
32377_Test	281	0.810	0.286	52.217	9.37	3.73	Hard	Sharply curved rows
35516_Test	967	0.341	0.230	53.002	2.15	0.97	Hard	Dense uncertain trees

MODFS Testing Orchard Predicted Rows and Ground-truth Orchards

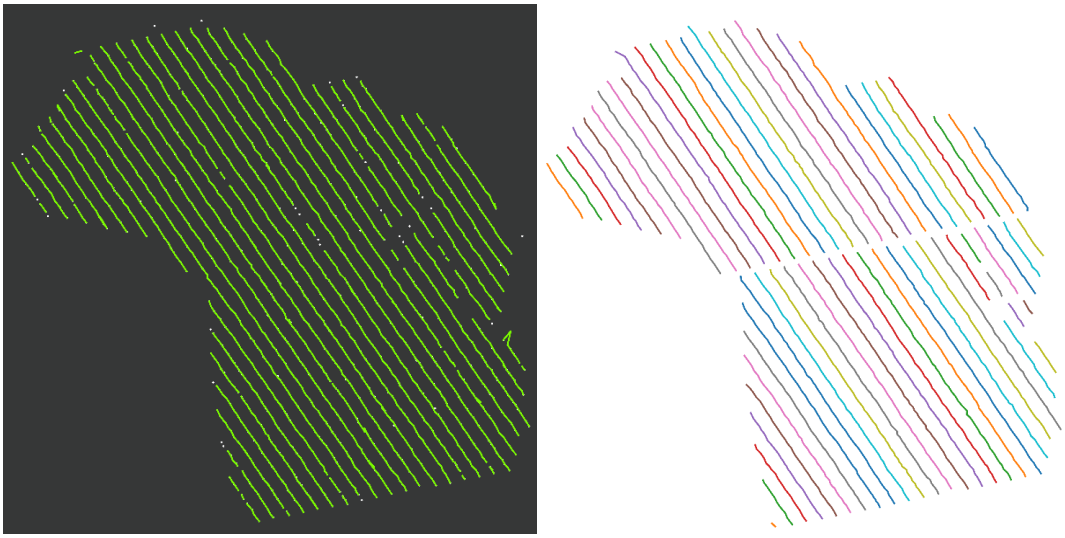
Orchard 32377 and its Ground-Truth Rows



Orchard 36502 and its Ground-Truth Rows

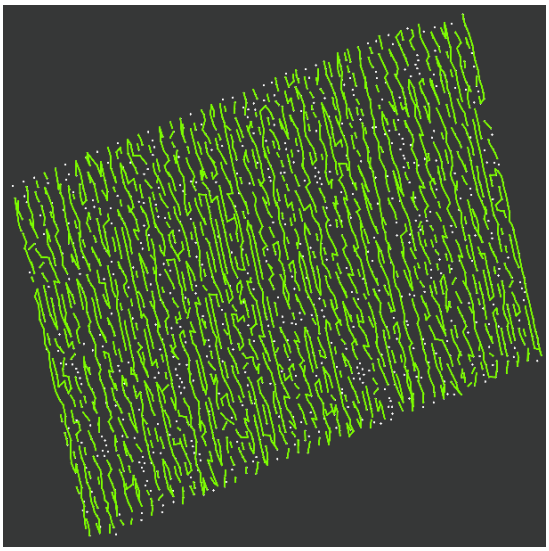


Orchard 36507 and its Ground-Truth Rows

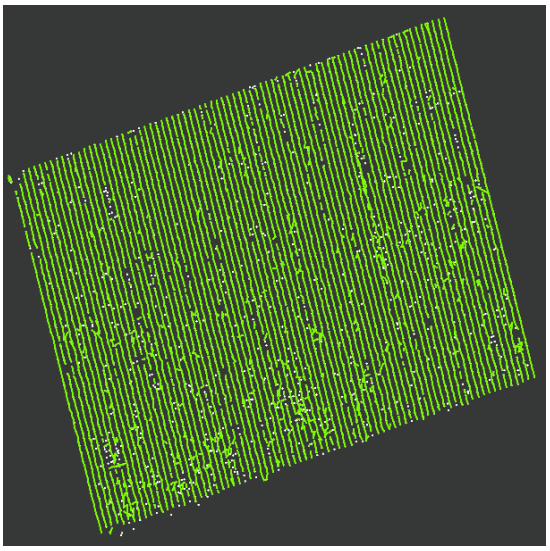


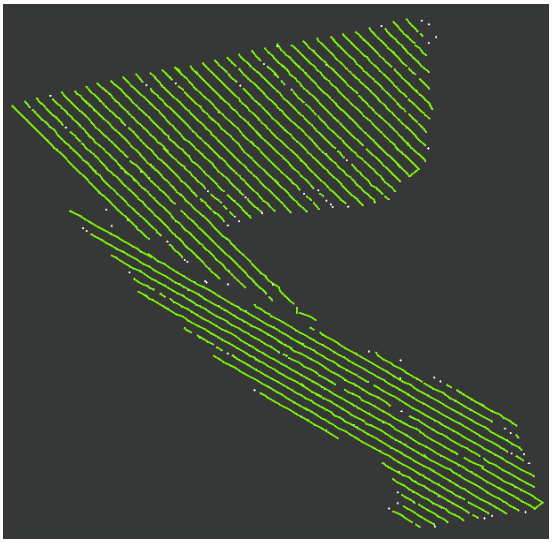
MODFS Orchard Predicted Rows

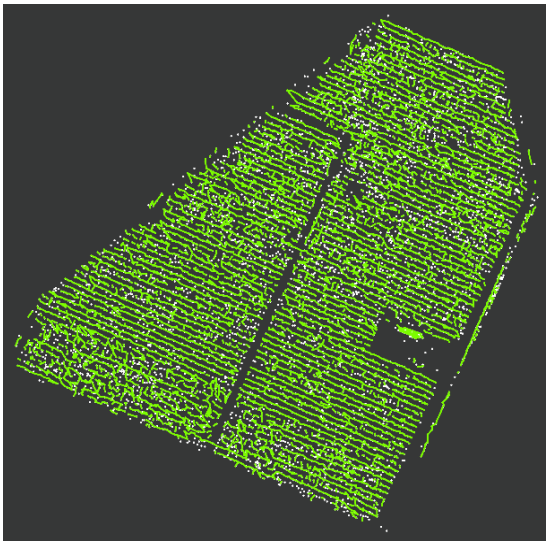
Orchard 41630



Orchard 43581







Orchard 36513

Orchard 35516