UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER

CS/IT Honours
Final Paper 2020

Title: ACO for Orchard Row Finding in Aerial Images

Author: Timothy Simons

Project Abbreviation: ORCHARD

Supervisor(s): Patrick Marais

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	0
Theoretical Analysis	*0*	*25*	15
Experiment Design and Execution	*0*	*20*	10
System Development and Implementation	*0*	*20*	0
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
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	**80**

ACO for Orchard Row Finding in Aerial Images

Timothy Simons
Department of Computer Science
University of Cape Town
September 2020

Abstract

The advent of unmanned aerial vehicles (UAVs) and accompanying aerial imagery has brought a variety of data-driven techniques to the domain of precision agriculture. Of interest is the identification of rows of trees from aerial images of orchard fields. All existing methods for solving this task utilise the Hough transform for straight line detection. Consequently, they all share the same limitation. That is, they perform poorly on curved rows. Furthermore, many of these methods rely on image processing techniques that aren't robust to varying input images. In this paper, we show that an Ant Colony Optimisation (ACO) algorithm, applied to detections of the underlying shrubs or trees, accurately finds the rows of a given orchard. Specifically, the proposed method achieves high precision and recall scores on a variety orchard inputs, each with differing detection density and row curvature. We show that the proposed algorithm is not only novel but is also an effective and robust approach to tackling the problem of orchard row identification.

1. Introduction

Agricultural management has become an increasingly difficult and cost-intensive task given the increasing size of modern-day farms. Presently, most of the orchard inspection is conducted on foot with workers walking between rows to examine each tree or shrub in an orchard. The manual inspection and constant supervision of trees is proving infeasible for larger farms and at least cost-ineffective for smaller ones. Recent advances in unmanned aerial vehicles (UAVs) and an increased accessibility to near real-time satellite images has enabled the automation of such processes.

Remote sensing of crops takes many different forms, each involving different applications and different types of data. This paper is concerned with the identification of orchard rows from aerial images of orchard fields. The acquisition of orchard row data lends itself to many applications, ranging from general crop planning to the identification of missing trees in an orchard. This problem has seen little work in previous years and the few methods that have been developed all opt for the same approach. That is, the image of the orchard first undergoes some form of image pre-processing followed by the application of the Hough transform for row detection [7, 23, 26]. The problem with this approach is that it assumes orchard rows follow straight lines. This is a brittle assumption that doesn't hold in the real world. Farmers must often plant rows to adhere to the landforms in their region. A typical example would be landforms containing varying elevation, which requires rows to be planted along contour lines. Sometimes rows are simply planted on an ad hoc basis, which also results in complex row patterns.

In this paper, we reframe orchard row identification as a combinatorial optimisation problem. This is achieved by treating every tree or shrub in an orchard as a distinct point. Now, instead of detecting rows from pixel data, we are finding rows from a set of points, with each point being the centre of a tree. This reduction in the problem description opens new avenues of exploration, particularly those related to graph structures and graph theory. We propose an Ant Colony Optimisation (ACO) algorithm for finding rows in an orchard. ACO is a well-known technique that is designed for finding good paths through graphs and is thus a perfect candidate for solving this problem.

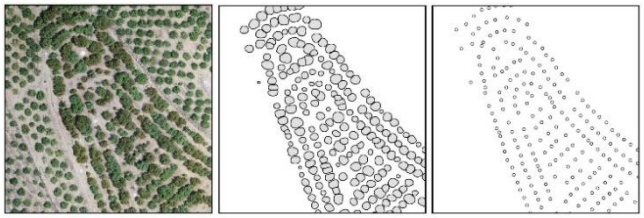


Figure 1: Orchard input image (left), tree detections (centre) and detection centroids (right).

First, we obtain detections of the underlying trees or shrubs using an existing object detector. Each of these detections take the form of a polygon whose centroid represents the centre of a tree. A feasible solution to the problem can now be described as a series of rows where each row represents a set of connected edges. This domain of discrete solutions is complex, more so than many other NP-hard problems. Indeed, we are not only trying to find a single optimal path in a graph, we are trying to find a set of paths whose configuration is optimum. This slight variation in the problem description adds many layers of complexity to the problem. Additionally, this is a multi-objective optimisation problem. The notion of an optimal row configuration for an orchard encodes the many assumptions made by the farmer who planted the orchard. Consequently, the objectives of our multi-objective algorithm should encode these underlying assumptions.

We have found that the best predictors of optimal rows are edge distances and edge angles. The first objective is therefore to maximise the difference between inter-row and intra-row distances; the second objective is to minimise the angle between two consecutive edges of a row. Different orchards will prefer different objectives. For an example, an orchard may constitute straight rows with irregular spaces between trees. This orchard will prefer the angle objective. Another row might be heavily curved but have consistent spaces between trees. This orchard will prefer the distance objective. Other orchards may prefer some combination of the two. We assume that *a priori* preference information of an orchard is known and thus combine these two objectives into a single objective using the weighted sum approach.

The algorithm proposed in this paper follows the ACO metaheuristic and is thus an ACO algorithm. Specifically, it is an elitist variant that uses random iterates and stochastic optimisation to ensure convergence over time. Simple agents (ants) probabilistically construct candidate rows whilst they adapt during algorithm execution to reflect their acquired search experience. The result of the ACO stage of the algorithm is a set of disjoint rows which are then combined during post processing. Our method achieves high precision and recall on a variety of orchard inputs. Moreover, our studies suggest optimal ACO parameters for different classes of orchard inputs.

We organise the paper as follows: Section 2 summarises previous approaches to the problem of orchard row identification. It also includes an overview and a few relevant applications of Ant Colony Optimisation. The proposed ACO algorithm for orchard row finding

is then outlined in Section 3. The experimental setup follows in Section 4 and results discussed in Section 5. We conclude the paper and propose future work in Section 6 and Section 7 respectively.

2. Related Work

2.1 Hough Transform for Row Detection

The Hough transform is a simple mathematical technique originally used in image analysis for detecting straight lines [15]. The relevance of this technique comes from the fact that orchard rows can be perceived as lines of trees or plants when viewed in an image. The idea of using the Hough transform to detect rows has been explored in several papers, in the context of both ground [1, 2, 19, 29] and aerial images [7, 23, 26].

Currently, to the best of our knowledge, the Hough transform has been the only technique used to identify orchard rows from aerial images. Despite the applicability of this approach, it is accompanied by numerous limitations. First and foremost, it is limited to detecting straight rows. Many of the papers using this technique neglect to mention this limitation and only consider straight rows as inputs. Additionally, all methods utilising the Hough transform require an extensive amount of pre-processing. The backbone of this pre-processing phase is a series of morphological operations that try to separate and reduce rows into distinct contiguous lines. These morphological operations are very much orchard dependent and rely heavily upon the definitive separation of rows from background pixels represented by soil, grass or other vegetation. The pre-processing employed by these methods is also dependent on spatial resolution and illumination changes of input images [23].

Of the aforementioned methods, only a single paper addresses the presence of curved rows [26]. This approach takes advantage of the fact that a curved line can be approximated through a series of straight lines. Each aerial image 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 the tiles to detect the rows in that tile. The detected straight lines are then combined to form the curved lines that represent the rows of the orchard. Experimental results of this method aren't too promising. Detections of crop rows produced poor precision and recall scores on considered orchard inputs. This approach also relies on the assumption that rows aren't too heavily curved. Rows that depict a high degree of curvature can't use image tiling as there would be too few trees in each tile to apply the Hough transform.

2.2 Ant Colony Optimisation (ACO)

Ant Colony Optimisation is a probabilistic technique for solving hard combinatorial optimisation problems that can be reduced to finding optimal paths through graphs. ACO is loosely based on the behaviour of ants and the mechanisms they employ when searching for food. Ants secrete pheromones that serve as a signalling mechanism that allow them to collectively navigate their environment. Accumulated pheromones represent the ants' preferences, typically based on quality of food, distance from the nest and amount of food available. These pheromone trails change dynamically over time to reflect the ants' acquired search experience.

ACO builds on this idea by incorporating heuristics specific to the problem being solved. The artificial ants in ACO construct candidate solutions by iteratively adding components to a partial solution. These components are chosen probabilistically as a function of trail pheromones and domain-specific heuristics. The stochastic component of ACO ensures the ants construct a diverse range of candidate solutions. This allows the algorithm to escape local optima and eventually approach a global optimum. In addition, ACO includes an optional local search algorithm, which takes a candidate solution and tries to find a better solution given some appropriately defined neighbourhood of the current solution. The algorithmic skeleton of a generic ACO algorithm is shown in Figure 2.

```
Algorithm Ant Colony Optimisation
1:  procedure ACO
2:      Initialisation
3:      while (termination condition not met) do
4:          ConstructSolution
5:          ApplyLocalSearch   % optional
6:          UpdatePheromones
7:      end
8:  end
```

Figure 2: Skeleton of a generic ACO algorithm.

During the main loop of the ACO algorithm, the ants construct candidate solutions to the problem. A global best candidate solution is updated every iteration. The termination criteria will typically involve the convergence of this global best solution.

When constructing candidate solutions, the artificial ants will add solution components probabilistically. If we assume a solution component to be an edge connecting node i and node j, the probability that an ant h will choose to traverse from node i to node j is calculated as

$$p_{ij}^{h} = \frac{(\tau_{ij}^{\alpha})(\eta_{ij}^{\beta})}{\sum_{k \epsilon N_i}(\tau_{ik}^{\alpha})(\eta_{ik}^{\beta})} \quad (1)$$

where τ_{ij} and η_{ij} are the pheromone and heuristic values associated with the edge connecting node i and node j. α and β control the relative importance of trail and heuristic information respectively. N_i is some appropriately defined neighbourhood of node i.

Once the ants have constructed their solutions, and these solutions improved upon through local search, edge pheromone information is updated. This ensures that edges or solution components belonging to good solutions are more desirable in following iterations. The pheromone values are updated using the update rule

$$\tau_{ij} \leftarrow (1-p)\tau_{ij} + \sum_{h} \Delta\tau_{ij}^{h} \quad (2)$$

where τ_{ij} is the pheromone value for edge ij, p is the evaporation rate and $\Delta\tau_{ij}^{h}$ is the pheromone deposited by the h^{th} ant on edge ij. The evaporation rate p is needed to prevent the rapid convergence to a sub-optimal solution. The amount of pheromone deposited $\Delta\tau_{ij}^{k}$ depends on the quality of the solution found by the h^{th} ant and is defined differently depending on the problem domain.

The first example of an ACO algorithm was called Ant System (AS) [12, 13], which was applied to the well-known travelling salesman problem (TSP). Although the results were promising, AS could not compete with state-of-the-art algorithms for TSP. Nevertheless, AS inspired many subsequent works that would later go on to achieve world class performance in their respective problem domains [14]. Examples of successful applications of the ACO algorithm include applications to probabilistic TSP [3], scheduling [5], assembly-line balancing [6], the quadratic assignment problem [27], and so on.

An application of the ACO algorithm that bears relevance to the problem of orchard row finding is that of edge detection [18, 21]. The similarity between ACO for edge detection and ACO for orchard row detection comes from the fact that neither algorithm is trying to find a single optimal path in a graph. Instead, they are both finding a set of paths whose configuration is optimum. ACO for edge detection treats each pixel in an image as a node in a graph. Artificial ants move from pixel to pixel, marking the traversed pixels with pheromones. The transition rule is based on the changes in grey level intensity of a 3×3 pixel region, where the centre pixel is the ant's position and the surrounding pixels are the ant's traversable neighbourhood. The resulting pheromone trails serve as a rough-cut

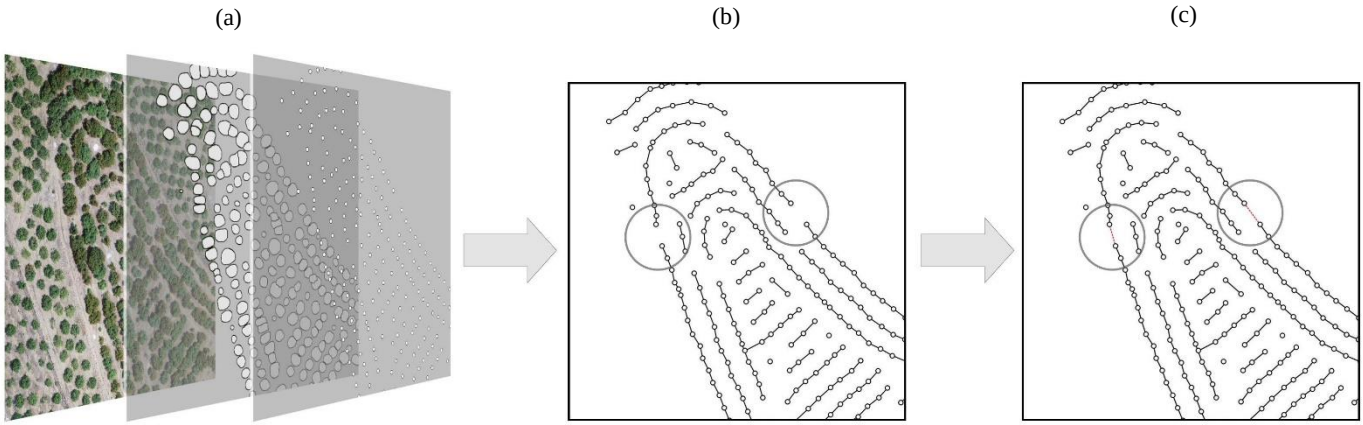


Figure 3: Orchard row finding pipeline with pre-processing (a), final rows of the ACO algorithm (b) and the final rows after stitching (c). Observe the differences between the circles in (b) and the circles in (c). The crux of the pipeline is the ACO algorithm; pre-processing and stitching constitute a small part of the process.

representation of the edges in an image. These pheromone trails then undergo thresholding and morphological thinning to produce the final edge pixels in the image. The ACO algorithm for edge detection produced excellent results on a selection of benchmark images [18].

3. Methods

The input data of the proposed method is a series of points with each point representing the centre of a tree. These points are simply the centroids of tree detections obtained from an existing object detector. The detections themselves are simple polygon features provided in GeoJSON format. These inputs only have a single non-spatial attribute, which is the confidence score associated with each detection.

3.1 Pre-processing

Unlike previously explored methods, pre-processing constitutes an inconsequential component of the row finding pipeline. The first step is the straight-forward thresholding of detections using the associated confidence scores. Detections with a confidence score below a certain threshold are removed from the search space. This threshold value depends on the orchard type as the performance of the object detector varies with different types of trees or shrubs.

Detection locations are provided in a geographic coordinate system (GCS) and consequently require some conversion into a projected coordinate system (PCS). The chosen PCS mapping depends on the location of the orchards.

3.2 ACO for Row Finding

We have redefined the problem of orchard row identification as a multi-objective combinatorial optimisation problem in which rows represent optimal paths through graphs. The optimality of these rows is encoded as some linear combination of distance and angle information, where smaller distances and smaller changes in direction are preferred.

Let's consider a graph $G = (P, E)$ with P being a set of points and E the set of edges fully connecting those points. The points $h \in P$ and $i \in P$ will represent tree-centers h and i respectively. We define a partial row r_p as the ordered set

$$r_p = \{c_i^t \mid i \in P \; and \; c_i^t > c_h^{t-1}\} \tag{3}$$

where c_i^t refers to a solution component that adds point i to the partial row r_p at construction step t. The component c_i^t can also be thought of as the edge $(h, i) \in E$ if the previously added component was c_h^{t-1}.

The heuristic value η_{hi} of the edge $(h, i) \in E$ is simply a linear combination of scaled distance and angle heuristics. We define the heuristic value of a component $\eta(c_i^t) = \eta_{hi}$ for $t > 0$ and $c_h^{t-1} \in r_p$ where

$$\eta_{hi} = w_1 d'_{hi} + w_2 \theta'_{ghi} \tag{4}$$

$$d'_{hi} = 1 - \frac{d_{hi}}{\sum_{k \in allowed_{n_p}(c_h^{t-1})} d_{ik}} \tag{5}$$

$$\theta'_{ghi} = \begin{cases} \frac{\theta_{ghi}}{180°} & if \; c_g^{t-2} \in r_p \\ 1 & otherwise \end{cases} \tag{6}$$

where d_{hi} is the Euclidean distance of edge $(h, i) \in E$ and $allowed_{n_p}(c_h^{t-1})$ is the n_p nearest unexplored points to the previously added point $h \in P$. The value θ_{ghi} is the angle at vertex h enclosed by rays $(g, h) \in E$ and $(h, i) \in E$.

We now define the pheromone value of a component $\tau(c_i^t) = \tau_{hi}$ for $t > 0$ and $c_h^{t-1} \in r_p$ where τ_{hi} is pheromone value associated with the edge $(h, i) \in E$. This pheromone value indicates the desirability of the component c_i^t. So, to summarise, each ant iteratively adds solution components c_i^t to a partial row r_p based on $\eta(c_i^t)$ and $\tau(c_i^t)$ until it becomes a candidate row r.

Given this context, we can define a suitable ACO algorithm for orchard row finding. The proposed algorithm differs slightly from the generic ACO algorithm. The difference comes from the fact that we are no longer finding candidate solutions but are instead finding candidate rows which collectively define a candidate solution.

Algorithm Ant Colony Optimisation for Orchard Row Finding

```
1:  procedure ACO
2:      Initialisation
3:      while termination condition not met do
4:          for i := 1 to number of search iterations do
5:              ConstructRows
6:              UpdatePheromones
7:          end
8:          RemoveBestRows
9:      end
10: end
```

Figure 4: Skeleton for the proposed ACO algorithm for orchard row finding.

The inner loop of the revised ACO algorithm serves the same function as the main loop of the general ACO algorithm and represents the search phase conducted by the artificial ants. Search iterations differ in that, instead of keeping track of only a single global best solution, we now consider the global best set of candidate

rows. A candidate row r_i is preferred over candidate row r_j if $v(r_i) > v(r_j)$ where $v(r)$ is the heuristic value total of row r. This global set represents the elitist component of the algorithm and is updated after each search iteration. The ants that find candidate rows in the best set are referred to as elitist ants.

$$v(r) = \sum_{c_i^t \in r} \eta(c_i^t) \;\; for\; t > 0 \qquad (7)$$

The criterion defined by (7) ensures that the best set of candidate rows comprise longer rows with higher heuristic values (i.e. longer, straighter rows with smaller intra-row distances). It is hoped that this criterion adequately reflects the implicit assumptions made by farmers when planting orchard rows.

Once the search phase is complete, a subset of the global best is selected for removal and is added to the set of final rows. This subset comprises the best m_e candidate rows based on (7) where m_e is an algorithm parameter. The outer loop terminates when there are no longer any points left in the search space. The main steps of the proposed ACO algorithm are explained further in the following.

1. *Initialisation*
   At the start of the algorithm, all the parameters are set. m ants are chosen to search for candidate rows over n search iterations. All values in the pheromone matrix τ are set to zero. The heuristic matrix is defined by (4).

2. *Row Construction*
   In the row construction phase, the set of *m* ants each construct a candidate row *r*. This is done iteratively by adding solution components c_i^t to a partial row r_p every construction step t. Each ant starts at a random point $i \in P$ such that $r_p = \{c_i^0\}$ at $t = 0$. During each subsequent construction step $t > 0$, an ant extends r_p by adding the component $c_i^t \in N(r_p) \subseteq C$ where

$$N(r_p) = Pareto(\, allowed_{n_p}(c_h^{t-1}) \,) \qquad (8)$$

$$Pareto(Q) = \{c_i^t \in Q \,|\, \{c_j^t \in Q |\, c_j^t \succ c_i^t, c_j^t \neq c_i^t\} = \emptyset\} \qquad (9)$$

   A component c_j^t strictly dominates ($\succ$) component c_i^t if both distance d'_{hj} and angle θ'_{ghj} heuristics of c_j^t are greater or equal to that of c_i^t. The pareto optimal set of nearest unexplored points $N(r_p)$ of a partial row r_p acts as a filter that removes substandard components from the set of traversable components. In other words, if a component results in an edge that is longer *and* less straight than that of another component, it is no longer considered in the neighbourhood of traversable components. In addition, an implicit threshold is used to prevent ants from constructing rows that reverse direction ($\theta < 90°$). This threshold can be thought of as the minimum possible angle between any two edges of any row in an orchard.

   The component $c_i^t \in N(r_p)$ is chosen probabilistically using the following transition rule, which is simply equation (1) rewritten for this problem description.

$$p(c_i^t | r_p) = \frac{\tau_{ij}^{\alpha}\, [\eta(c_i^t)]^{\beta}}{\sum_{c_j \in N(r_p)} \tau_{ik}^{\alpha}\, \left[\eta(c_j^t)\right]^{\beta}} \qquad (10)$$

$$\forall\, c_i^t \in N(r_p), t > 0$$

   The process of adding solution components c_i^t to the partial row r_p continues until $N(r_p) = \emptyset$ or all components in $N(r_p)$ result in $\theta < 90°$. Once an artificial ant has constructed a the candidate row in one direction the row is reversed and the process continued in the other direction $constructRows(r_p^*)$ where r_p^* is a reversal of partial row r_p. This simply ensures the ants construct the candidate row in both directions given some random starting point. The result is a candidate row r.

3. *Pheromone Update*
   The pheromone update rule is defined by (11) which is simply (2) rewritten for this problem description. We define the update rule $\tau(c_i^t) = \tau_{hi}$ for $t > 0$ and $c_h^{t-1} \in r$ where

$$\tau_{hi} \leftarrow (1-p)\tau_{hi} + \sum_{c_i^t \in r | r \in R} \Delta\tau(c_i^t) \qquad (11)$$

   p is the evaporation rate and *R* is the set of candidate rows found during the current search iteration. The amount of pheromone deposited on a component $c_i^t \in r$ is defined as $\Delta\tau(c_i^t) = \Delta\tau[t]$ where

$$\Delta\tau[t] = (f * g)[t] \qquad (12)$$

$$(f * g)[t] = \sum_{u=-q}^{q} f[u]\, g\,[t-u] \qquad (13)$$

$$g[t] = \eta(r[t]) \qquad (14)$$

   This is essentially a rolling window calculation (or moving average) with a window function *f* and a window size *q* applied to the heuristic values of row r where $r[t] = c_i^t$. The window function used is a triangle window function that assigns the largest weight to the heuristic value of current component and smaller weights to the values of components further away and on either side of the current component. This ensures that the pheromone deposited by an ant on a component c_i^t is also a function of other components in the same candidate row r.

   Recall that the deposited pheromone on an edge indicates the desirability of that edge. The importance of the rolling window calculation comes from the fact that no component or edge can be considered good or bad in isolation. Suppose we have a small, straight edge $(h, i) \in E$. This would give us a large heuristic value η_{hi}. Now suppose the components on either side of that edge result in low heuristic values, i.e. they are characterised by long edges and/or result in drastic changes in direction. If this is the case, then surely the edge $(h, i) \in E$ is not so desirable. The opposite can be said if the edge $(h, i) \in E$ has a small heuristic value but connects edges with large heuristic values. This idea extends to the entire row r containing the edge $(h, i) \in E$, albeit with less importance being placed on edges further away from the considered edge. Hence, the reason for the rolling window calculation.

4. *Row Removal*
   A subset of size m_e of the best set of candidate rows are considered final rows. The points that make up these rows are removed from search space. This corresponds to the biological idea of ants eating the food that they have found. The evaporation rate ensures the artificial ants adapt to this removal to find the next best candidate rows in further search iterations.

3.3 Stitching

Stitching is the process of joining disjoint final rows. If the ends of two disjoint rows are roughly collinear and are sufficiently close to one another, the rows are joined. More concretely, stitching is a process that recursively joins rows to a current row. This is done in opposite directions, starting with the endpoint on one side of the row and then proceeding with the other side.

First, the nearest endpoints to the current endpoint are retrieved. These nearest endpoints belong to disjoint final rows different from the current row. The endpoints and penultimate points between these rows and the current row are compared for collinearity. The row that results in the least change in direction is added to the current row. This process continues until all final row endpoints have been considered. We employ the same implicit angle threshold as the one in Section 3.2 to prevent the joining of rows that extend in opposite directions.

4. Experimental Setup

The data used in this paper comprises a selection of varying orchard input images. We consider four different orchard inputs with each input portraying a unique set of row and detection characteristics. The selected inputs are ordered by *row curvature* and *detection density*.

Orchard	Curved	Dense
A (Fig. 5)	*No*	*No*
B (Fig. 6)	*Yes*	*No*
C (Fig. 7)	*No*	*Yes*
D (Fig. 8)	*Yes*	*Yes*

Table 1: Selected orchard inputs with varying row curvature and detection density.

For the purposes of this paper, we define curved inputs as any orchard containing curved rows. Dense inputs refer to orchards that result in irregular clusters of closely grouped detections. Many detections in dense inputs don't visibly adhere to any row in the orchard. These detections may be spurious or simply correctly identified trees or shrubs that don't belong to any row. Dense inputs are also characterised by irregular intra-row spacing. A centroid subset for each of the considered inputs can be seen in Figure 5.

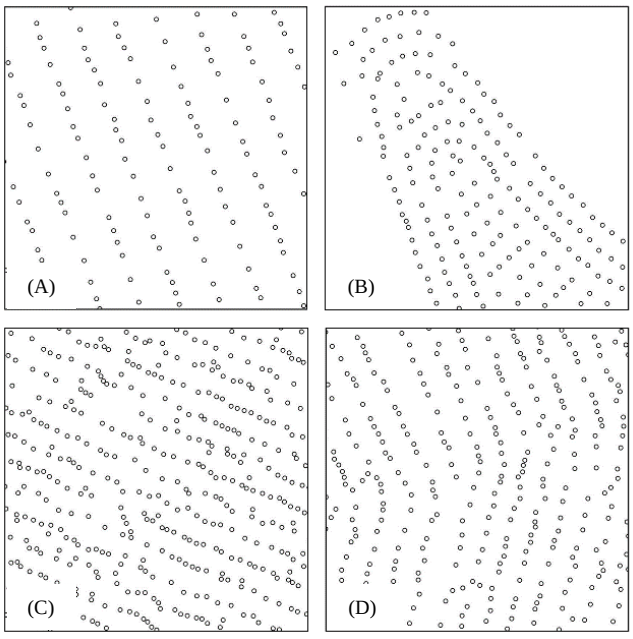


Figure 5: Subsets of detection centroids for each considered input.

We compare the performance of the proposed ACO algorithm on each of these orchard inputs. The evaluation metrics used are *precision*, *recall* and *intersection over union* (*IoU*) defined by equations (15), (16) and (17) respectively. To compute these metrics, we compare model-produced rows against hand-labelled rows. Model and hand-labelled rows comprise a series of edges which are compared for similarity. A *true positive* (*TP*) constitutes a model edge that correctly identifies a true edge, a *false positive* (*FP*) refers to a model edge that doesn't correspond to a true edge and a *false negative* (*FN*) refers to a true edge that hasn't been identified by a model edge.

$$precision = \frac{TP}{TP + FP} \tag{15}$$

$$recall = \frac{TP}{TP + FN} \tag{16}$$

$$IoU = \frac{TP}{TP + FP + FN} \tag{17}$$

The parameters for all experiments, unless otherwise stated, are the default parameters in Table 2. Experiments were conducted on an Asus laptop running Windows 10 x64. The machine has 16.0 GB of RAM and a four core AMD Ryzen 7 CPU with clock speeds of 2.30 GHz.

Ant Colony Parameters	
Ant colony size (m)	15
Number of search iterations (n)	15
Number of elites (m_e)	5
Ant Parameters	
Neighbourhood size (n_p)	4
Distance weight (w_1)	0.5
Angle weight (w_2)	0.5
Alpha (α)	0.5
Beta (β)	0.5
Pheromone Parameters	
Evaporation rate (p)	0.1
Window size (q)	5
Window function (f)	Triangle

Table 2: Default parameters of the proposed ACO algorithm for orchard row finding.

5. Results and Discussion

First, we conduct an exploratory analysis of important ACO parameters to identify their effect on performance for different orchard inputs. Using this analysis, we construct models with optimal parameters and visualise the results. Lastly, we consider the time complexity of the proposed ACO algorithm.

5.1 Parameter Selection and Analysis

5.1.1 *Key parameters*

In this section, we analyse algorithm parameters listed in Table 2. This is a difficult undertaking as the proposed ACO algorithm exhibits many confounding (or lurking) variables. One such example is that of deposited pheromones defined by (12). To illustrate the effect of this confounding variable, we consider the complex relationship between the number of search iterations (n), the relative importance assigned to pheromone trails (α) and the evaporation rate (p).

Suppose we want to test the effect of the number of search iterations n on algorithm performance. When increasing parameter n, we increase the number of candidate rows constructed during the search phase of the algorithm. This increases the number of candidate rows considered for the global best set. This also has the effect of increasing the pheromones deposited on all trails prior to row removal. Now suppose, for a particular α and p, increases in deposited pheromones reduce exploration of the artificial ants. Given this behaviour, we find that increasing parameter n will result in little to no change in algorithm performance. This is a direct result of increases in deposited pheromones, which lead to sub-optimal candidate rows in the global best set. Different values for α and p may result in the opposite confounding effect, i.e., guides the ants to higher quality solutions resulting in an increase in performance. In this example, we show that there exists a spurious correlation between the parameter n and algorithm performance. The confounder is the deposited pheromones (coupled with the values of α and p) which incorrectly imply causation between the parameter n and algorithm performance.

This is one among many other, albeit more subtle, examples of confounding variables in the proposed ACO algorithm. Given the presence of these lurking variables, the effects of parameters on algorithm performance should not be considered in isolation, but rather compared across the various orchard inputs. This comparison is statistically sound because confounding variables remain constant across inputs, meaning that any difference in algorithm performance is a direct result of differences between orchards. Figure 8 in the Appendix illustrates the effect of several key parameters on algorithm performance. Considered parameters include the number of search iterations n, the parameters α and β, and the relative weights w_1 and w_2 of distance and angle heuristics respectively. Average *precision*, *recall* and *IoU* scores are plotted as a function of these parameters with each score being averaged over three algorithm runs.

The first conclusive result of the findings presented in Figure 8 relates to the parameter n. These findings show that dense inputs (C and D) require more search iterations (n) to find optimal rows than sparse inputs (A and B). This result is intuitive because the neighbourhood $N(r_p)$ of any partial row r_p is larger for dense inputs than for sparse inputs. That is, the artificial ants have a wider range of available solution components $c_i^t \in N(r_p)$ to consider when constructing a candidate row r. An increased number of options mean more paths to explore, which can only be explored through an increase in the number of search iterations (n).

Upon viewing the graphs for different values of w_1 and w_2, we notice an unexpected result. A distance weight (w_1) of 1.0 and an angle weight (w_2) of 0.0 is preferred by all orchard inputs. Though unexpected, there are several reasons for this result. The predominant reason relates the structure of considered orchards and how the ground-truth rows of these orchards are defined. Ground-truth rows generally prefer rows with shorter edges than rows with constant direction. This is because, across all inputs, intra-row distances are smaller than inter-row distances most of the time. Consequently, the artificial ants can best match the ground-truth if they minimise distances rather than minimise changes in direction. If the ground-truth were to instead prefer straighter rows, a higher angle weight (w_2) would be preferred. Another reason for this weight preference is the inclusion of the Pareto mechanism defined in (9) and the minimum angle threshold described in Section 3.2. These mechanisms remove obviously incorrect edges that would result in rows with high angle variation or drastic changes in direction. This significantly reduces the search space that the ants must explore and allows the ants to rely less on the angle objective encapsulated by the parameter w_2.

5.1.2 *Implicit angle threshold*

The minimum angle threshold, applied to connected edges of constructed rows, was treated as an implicit component of the proposed ACO algorithm. This threshold simply prevented the artificial ants from constructing rows that reverse direction. That is, any edge pair $(h, i) \in E$ and $(i, j) \in E$ with $\theta_{hij} < 90°$ was not considered traversable. Through experimentation, it was found that this threshold could in fact serve as a valuable tool for guiding ants towards higher quality solutions.

The threshold values that resulted in significant increases in performance were $145°$ for sparse inputs and $115°$ for dense inputs. Threshold values beyond this point resulted in a decrease in *recall* scores. This is caused by true edges being erroneously rejected for exhibiting angles smaller than the specified thresholds.

5.2 Performance

The performance of the ACO algorithm is evaluated using the accuracy metrics defined in Section 4, namely *precision*, *recall* and *IoU*. These metrics are calculated by comparing model-produced rows to hand-labelled (ground-truth) rows for each orchard input. This is an important point to consider, as it is not these accuracy metrics that the ACO algorithm optimises; the ACO algorithm optimises the criterion defined by (7). That is, it optimises the total heuristic values of all rows in an orchard. This fundamental difference leads us to an important conclusion: the performance of the proposed ACO algorithm is a direct function of, not only how well it optimises (7), but also of how closely this criterion reflects the implicit assumptions employed by the hand-labeller when labelling the orchard inputs. This idea is a recurring theme in the following analysis and presents itself differently across each of the orchard inputs.

The best-found performance, and corresponding parameters, for each orchard input are shown in Table 3, the results of which are visualised in the Appendix.

1. *Orchard A (not dense; not curved)*
   Although the proposed ACO algorithm performed the best on this input, the visualised results hint at a possible shortfall of the approach. We notice that the ACO algorithm has erroneously connected rows on either side of well demarcated path. This is an example of where a reduction in the problem description has resulted in some lost information. Looking at just a set of points, it is impossible to tell that these rows are in fact separate.

2. *Orchard B (not dense; curved)*
   This orchard input illustrates an example of where the definition of the ground truth affects algorithm performance. From looking at the hand-labelled rows, we see that a missing tree (or a significant gap between trees) has been taken to signify the end of a row. This interpretation of what constitutes a row has resulted in many *false positive* edges that, under a different interpretation, would be correct. Furthermore, assumptions about rows are not consistent across all hand-labelled rows. In many cases these hand-labelled rows have traversed gaps between trees. This has resulted in *false negatives* that should be considered correct under the assumption that missing trees separate rows.

3. *Orchard C (dense; not curved)*
   This orchard is the most difficult of all considered inputs. Rows have irregular intra-row spaces and connected edges exhibit large angle variations. Trees are clustered together with many trees not adhering to a row. Indeed, in certain localised regions, no visible pattern or structure of trees can be determined. It is only upon looking at the bigger picture can one ascertain the

Input orchard	Detections	Ants (m)	Elites (m_e)	Search (n)	Distance (w_1)	Angle (w_2)	Alpha (α)	Beta (β)	Precision	Recall	IoU	Time (s)
A (Fig. 5)	2461	1	1	1	1.0	0.0	0.5	0.5	0.9921	0.9987	0.9909	1.57
B (Fig. 6)	1985	15	5	15	1.0	0.0	0.5	0.5	0.9637	0.9568	0.9235	65.53
C (Fig. 7)	3043	15	5	50	1.0	0.0	0.5	0.5	0.8910	0.8523	0.7718	361.64
D (Fig. 8)	3816	15	5	30	1.0	0.0	0.5	0.5	0.9816	0.9579	0.9410	229.11

Table 3: Parameters and performance statistics of the ACO algorithm on each orchard input.

direction of the row. The probabilistic nature of the ACO algorithm enables the artificial ants to effectively traverse this complex environment and identify the relevant rows in the orchard.

A single obstacle was encountered that diminished algorithm performance for dense inputs. Numerous *false positives* and *false negatives* were produced by the stitching phase of the algorithm. This can be seen in Figure 6, where the joining of disjoint rows resulted in triangle/trapezium structures with the bottom edge being a *false positive* and the side edges being *false negatives*. In each case, it is arguable whether the missed points do in fact belong to the relevant rows. If we were to assume the stitching to be correct, far higher performance scores would be achieved.

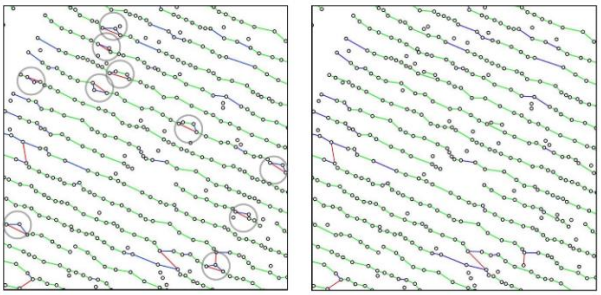


Figure 6: Model produced rows under the assumption that stitching is incorrect (left) and under the assumption that stitching is correct (right).

4. *Orchard D (dense; curved)*
   This input orchard is slightly less dense than that of orchard C and, consequently, results in better algorithm performance. As illustrated in the results for orchard B, curved rows are accurately identified by the ACO algorithm.

5.3 Complexity

The time complexity of the presented algorithm is a function of orchard size and complexity, and the three ant colony parameters in Table 2, namely ant colony size (m), number of search iterations (n) and number of elites (m_e). That is, the time complexity of the algorithm is $O(m\,n\,\frac{|R|}{m_e})$ where $R \subseteq E$ is the set of edges that constitute final rows at algorithm completion.

The execution times of the ACO algorithm for each orchard is visualised in Figure 7 in the Appendix. The parameters m, n and m_e are held constant across inputs. The only varying factors are orchard size and orchard complexity. We notice longer execution times for larger orchards (C and D) than for smaller orchards (A and B). We also observe that, despite orchard D being larger than orchard C, a longer algorithm runtime for orchard C. This is because orchard C is more complex and results in more final rows at algorithm completion. A similar result is observed when comparing orchards, A and B

6. Conclusions

In this paper, we explore a novel approach to the problem of orchard row identification in aerial images. The crux of the proposed method is a modified ACO algorithm in which artificial ants traverse graph-representations of orchards to produce optimal paths that represent the rows of trees in those orchards. ACO for orchard row-finding not only modifies the generic ACO algorithm, but also includes newly devised mechanisms specific to this problem domain. These mechanisms include the nearest unexplored pareto set of a partial row and the rolling window calculation of pheromone updates. Additionally, heuristic values were defined to reflect the multi-objective nature of finding rows of trees within orchards. These objectives minimise intra-row distances and changes in row direction.

Unlike Hough-based methods, the proposed approach constitutes minimal pre-processing and can reliably identify curved rows of trees or shrubs. Furthermore, experimental results illustrate the robustness of the method to orchards with varying tree density.

7. Future Work

The current study of algorithm parameters is limited in scope. Further analysis could be conducted to isolate the effect of each of the parameters on algorithm performance. Additionally, a reduction in the effect of confounding factors can be achieved by increasing the number and variety of comparative experiments in the analysis.

A select group of inputs were considered in this paper. Additional, more varied inputs could be considered in subsequent analyses as well the inclusion of possible failure cases.

8. Acknowledgements

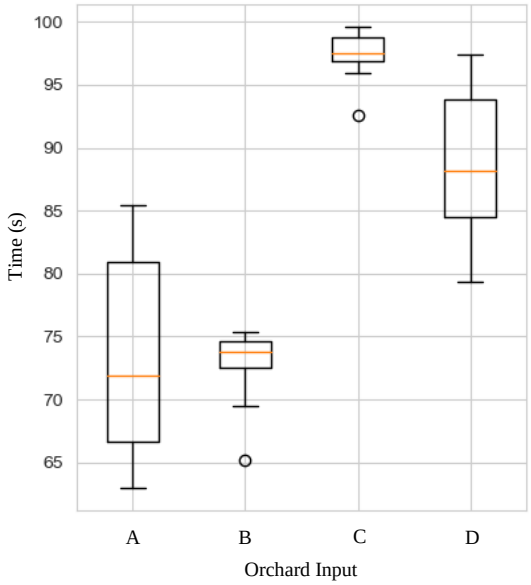
We would like to thank A/Prof. Patrick Marais for all his guidance and supervision throughout the research project. We would also like to thank Aerobotics for providing of the orchard data used in the project.

References

[1] Åstrand, B. & Baerveldt, A.J.J.M. A vision-based row-following system for agricultural field machinery, 15, 2 (2005), 251-269.
[2] Bah, M. D., Hafiane, A. and Canals, R. Weeds detection in UAV imagery using SLIC and the hough transform. IEEE, City, 2017.
[3] Balaprakash, P., Birattari, M., Stützle, T., Yuan, Z. and Dorigo, M. J. S. I. Estimation-based ant colony optimization and local search for the probabilistic traveling salesman problem, 3, 3 (2009), 223-242.
[4] Ballard, D. H. J. P. r. Generalizing the Hough transform to detect arbitrary shapes, 13, 2 (1981), 111-122.
[5] Blum, C. J. C. and Research, O. Beam-ACO—Hybridizing ant colony optimization with beam search: An application to open shop scheduling, 32, 6 (2005), 1565-1591.
[6] Blum, C. J. I. J. o. C. Beam-ACO for simple assembly line balancing, 20, 4 (2008), 618-627.
[7] Comba, L., Gay, P., Primicerio, J., Aimonino, D. R. J. C. and Agriculture, E. i. Vineyard detection from unmanned aerial systems images, 114 (2015), 78-87.
[8] Deb, K., Agrawal, S., Pratap, A. and Meyarivan, T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. Springer, City, 2000.
[9] Doerner, K., Hartl, R. F. and Reimann, M. Are COMPETants more competent for problem solving? The case of a multiple objective transportation problem (2001).
[10] Doerner, K., Gutjahr, W. J., Hartl, R. F., Strauss, C. and Stummer, C. J. A. o. o. r. Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection, 131, 1-4 (2004), 79-99.
[11] Dorigo, M. and Di Caro, G. Ant colony optimization: a new meta-heuristic. IEEE, City, 1999.
[12] Dorigo, M., Maniezzo, V., Colorni, A. J. I. T. o. S., Man, and Cybernetics, P. B. Ant system: optimization by a colony of cooperating agents, 26, 1 (1996), 29-41.
[13] Dorigo, M., Maniezzo, V. and Colorni, A. The ant system: An autocatalytic optimizing process (1991).
[14] Dorigo, M. and Stützle, T. Ant colony optimization: overview and recent advances. Springer, City, 2019.
[15] Duda, R. O. and Hart, P. E. J. C. o. t. A. Use of the Hough transformation to detect lines and curves in pictures, 15, 1 (1972), 11-15.
[16] García-Martínez, C., Cordón, O. & Herrera, F.J.E.J.o.O.R. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP, 180, 1 (2007), 116-148.
[17] Iredi, S., Merkle, D. and Middendorf, M. Bi-criterion optimization with multi colony ant algorithms. Springer, City, 2001.
[18] Jevtić, A., Quintanilla-Dominguez, J., Cortina-Januchs, M. G. and Andina, D. Edge detection using ant colony search algorithm and multiscale contrast enhancement. IEEE, City, 2009.
[19] Jiang, G.-Q., Zhao, C.-J. and Si, Y.-S. A machine vision based crop rows detection for agricultural robots. IEEE, City, 2010.

[20] Mariano, C. E. and Morales, E. J. I. M. d. T. d. A., Technical Report HC- A multiple objective ant-Q algorithm for the design of water distribution irrigation networks (1999).
[21] Nezamabadi-Pour, H., Saryazdi, S. & Rashedi, E.J.S.C. Edge detection using ant algorithms, 10, 7 (2006), 623-628.
[22] Nolan, A., Park, S., Fuentes, S., Ryu, D. and Chung, H. Automated detection and segmentation of vine rows using high resolution UAS imagery in a commercial vineyard. City, 2015.
[23] Ramesh, K., Chandrika, N., Omkar, S., Meenavathi, M., Rekha, V. J. I. J. o. I., Graphics and Processing, S. Detection of rows in agricultural crop images acquired by remote sensing from a uav, 8, 11 (2016), 25.
[24] Rovira-Más, F., Zhang, Q., Reid, J. and Will, J. J. P. o. t. I. o. M. E., Part D: Journal of Automobile Engineering Hough-transform-based vision algorithm for crop row detection of an automated agricultural vehicle, 219, 8 (2005), 999-1010.
[25] Sklansky, J.J.I.T.o.c. 1978. On the Hough technique for curve detection. (10):923-926.
[26] Soares, G.A., Abdala, D.D. & Escarpinati, M. Eds. 2018. Plantation Rows Identification by Means of Image Tiling and Hough Transform. 453-459.
[27] Stützle, T. and Dorigo, M. J. N. i. i. o. ACO algorithms for the quadratic assignment problem, C50 (1999), 33.
[28] Tian, J., Yu, W. and Xie, S. An ant colony optimization algorithm for image edge detection. IEEE, City, 2008.
[29] Vidović, I., Scitovski, R. J. C. and agriculture, e. i. Center-based clustering for line detection and application to crop rows detection, 109 (2014), 212-220.
[30] Yang, Q. and Yoo, S.-J. J. I. A. Optimal UAV path planning: Sensing data acquisition over IoT sensor networks using multi-objective bio-inspired algorithms, 6 (2018), 13671-13684.

Figure 7: Boxplot of execution times from 10 algorithm runs on each input with the default parameters in Table 2.



Appendix

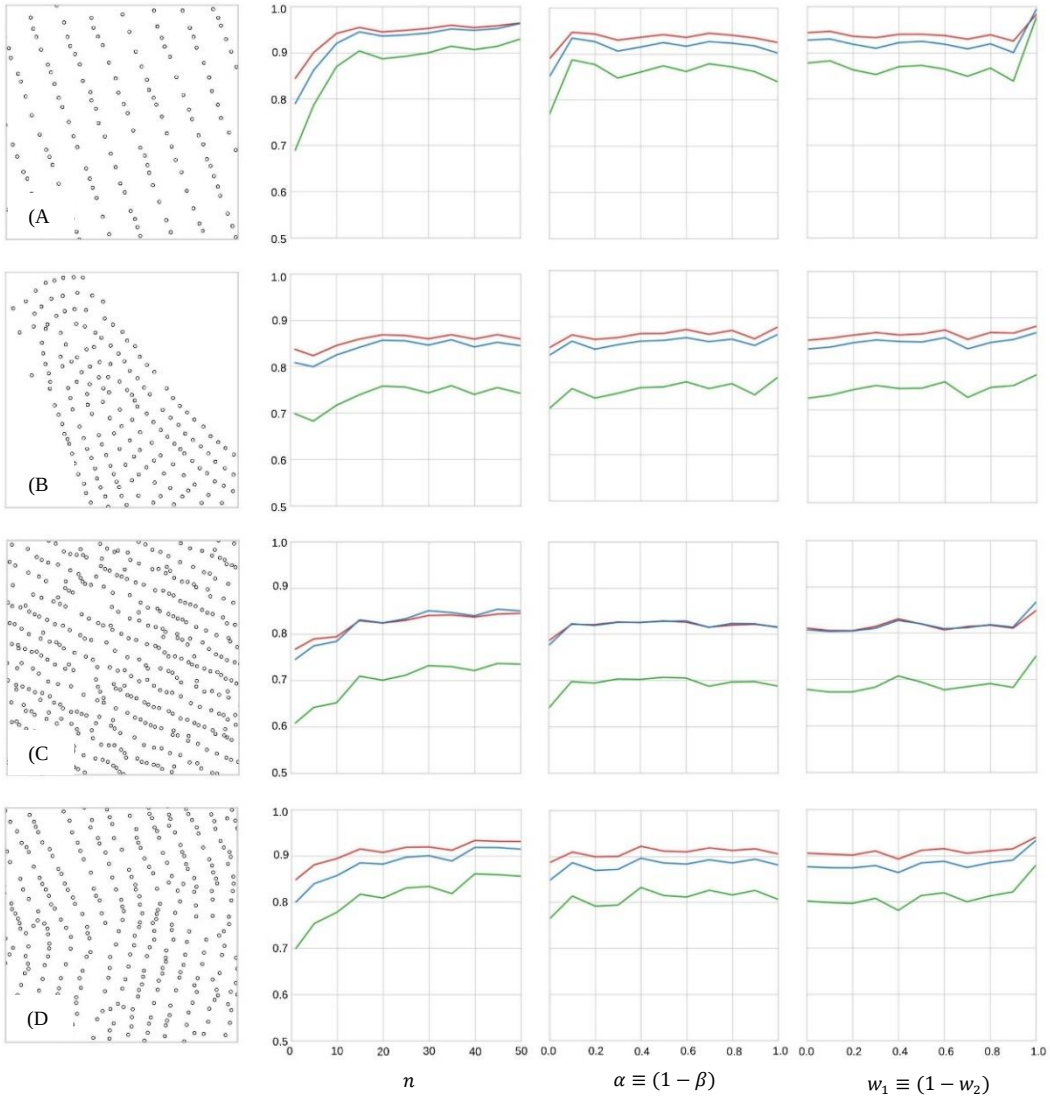


Figure 8: ACO performance scores for a selection of varying algorithm parameters. Red = precision, blue = recall and green = IoU.

Figure 9: Performance results of proposed ACO algorithm on orchard input A.
Green edge = *true positive*, red edge = *false positive*, blue edge = *false negative*.
The precision recall and IoU scores for this result are given in Table 3.

Figure 10: Performance results of proposed ACO algorithm on orchard input B. Green edge = *true positive*, red edge = *false positive*, blue edge = *false negative*. The precision recall and IoU scores for this result are given in Table 3.

Figure 11: Performance results of proposed ACO algorithm on orchard input C. Green edge = *true positive*, red edge = *false positive*, blue edge = *false negative*. The precision recall and IoU scores for this result are given in Table 3

Figure 12: Performance results of proposed ACO algorithm on orchard input B. Green edge = *true positive*, red edge = *false positive*, blue edge = *false negative*. The precision recall and IoU scores for this result are given in Table 3