ACO for Orchard Row Finding in Aerial Images

Timothy Simons

Department of Computer Science

University of Cape Town

May 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 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 on orchard inputs with rows of varying length and curvature.

# 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.

A screenshot of a cell phone

Description automatically generatedRemote 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. 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.

**Figure 1**: Image depicting the underlying orchard (left), shrub detections (middle) and detection centroids (right).

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.

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, our multi-objective algorithm should also 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 rows. 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 over a variety of orchard inputs. Moreover, our studies suggest optimal ACO parameters for different classes of orchard inputs.

Ideas

* Talk about the human visual system as being capable of discerning complex patterns from data. But for a machine it becomes an NP hard problem.
* Mention what type of object detection is employed.
* Slightly more technical detail in the intro (more jargon??)
* Maybe mention something about NP-hardness: i.e. time needed to solve an instance in the worst case grows exponentially with instance size.
* Mention IoU in conjunction with precision and recall.

# Related Work

## Hough Transform for Row Detection

The Hough transform is a simple mathematical technique originally used in image analysis for detecting straight lines [10]. The relevance of this technique comes from the fact that orchard rows can be perceived as lines 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, 14, 20] and aerial images [2, 18, 23].

Currently, to the best of our knowledge, the Hough transform has been the only technique used to identify orchard rows. 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. This pre-processing is very much orchard dependent and relies 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.

The straight-line problem is addressed by Soares et al. (2018) who uses image tiling and the Hough transform to identify rows in coffee crop fields. This approach takes advantage of the fact that a curved line can be approximated through a series of straight lines [23]. Each aerial image of a coffee crop field is segmented into a series of tiles such that the rows in each tile are approximately straight. This allows the use of the Hough transform in each of 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 values 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.

## 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.

|  |
| --- |
| 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 edgeconnecting node *i* and node *j*, the probability that an ant *h* will choose to traverse from node *i* to node *j* is calculated as

(1)

where and are the pheromone and heuristic values associated with the edge connecting node and node . α and β control the relative importance of trail and heuristic information respectively. is the neighbourhood of node .

Once the ants have constructed their solutions, and these solutions improved upon through local search, the 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

(2)

where is the pheromone value for edge is the evaporation rate and is the pheromone deposited by the ant on edge The evaporation rate is needed to prevent the rapid convergence to a sub-optimal solution. The amount of pheromone deposited depends on the quality of the solution found by the ant and is defined differently depending on the problem domain.

Ideas

* Talk about ACO for edge detection.
* Mention brief history of origin.

Questions

# 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. This detector comprises a Mask R-CNN variant with a ResNet50 backbone and a standard Feature Pyramid Network. The detections are made in overlapping tiles and merged with non-max suppression. 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.

## Pre-processing

Unlike previously explored methods, pre-processing constitutes an inconsequential segment 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 some predefined 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.

## 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 with being a set of points and the set of edges fully connecting those points. The points and will represent tree-centers and respectively. We define a partial row as the ordered set

where refers to a solution component that adds point to the partial row at construction step *t.* The component can also be thought of as the edge if the previously added component was The pheromone value of the edge is defined as in the general ACO algorithm. The heuristic value of the edge is simply a linear combination of scaled distance and angle heuristics. We now define the heuristic value of a component for and where

where is the Euclidean distance of edge and is the *n* nearest unexplored neighbours to the previously added point . The value is the angle at vertex *h* enclosed by rays and .

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 iterations do 5. ConstructRows 6. UpdatePheromones 7. end 8. RemoveBest 9. end 10. end |

**Figure 3**: 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 conducted by the artificial ants. Instead of keeping track of only a single global best solution, we now consider the global best set of candidate rows. Once the search phase is complete, a subset of the global best is selected and added to the set of final rows. The points in these chosen rows are removed from the search space. The outer loop terminates when there are no longer any points 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* iterations. All values in the pheromone matrix are set to zero. The heuristic matrix is defined by (3).

1. *Row Construction*

In this step, the set of *m* ants each construct a candidate row. This is done iteratively by adding solution components to a partial row . Each ant starts at a random point such that . During each subsequent construction step , an ant extends by adding the component .

A component strictly dominates component if both distance and angle heuristics of are greater or equal to that of . The component is chosen probabilistically using the following transition rule, which is simply equation (1) rewritten for this problem description.

The process of adding solution components to the partial row continues until there are no longer any unexplored nearest neighbours or all available components have angle or distance heuristics above some predefined threshold. Once the artificial ant has constructed a the candidate row in one direction the row is reversed and the process continued in the other direction . This simply ensures the ants construct the candidate row in two opposite directions given some random starting point. The result is a candidate row

1. *Pheromone Update*

The pheromone update rule is the same as in (2). We now define the pheromone deposited on a component τ for and

The deposited pheromone on candidate row is a rolling window calculation with a window function *f* and a window size *q* applied to the heuristic values of row where *.* The window function used is a triangle window function that ascribes 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 value of a component is also a function of other components in the same candidate row .

1. *Row Removal*

The best set of found candidate rows are then considered final rows. The points that make up these final rows are removed from the set .

This has a nice biological correspondence when thought of in terms of ant eating their food.

The pheromone trails will adapt in accordance to the food that is no longer there.

Evaluation function is simply the sum of heuristics.

Maybe define the new representation of the problem i.e. series of points which need to construct candidate rows. Search space of points.

Search iterations.

After algorithm we talk about

The proposed algorithm differs slightly from the general ACO algorithm. This is because we are no longer finding candidate solutions but are instead finding candidate rows which collectively define a candidate solution.

This distinction requires the introduction of an inner loop that essentially performs the same function as the main loop of the generic ACO algorithm. During the first iteration of the inner loop, *m* ants will construct candidate rows, which are then saved as being the global best. In each subsequent iteration the set of *m* best candidate rows are updated. Upon completion of the inner loop, the *m* best candidate rows are removed from the search space and are now considered final rows. The outer loop terminates when there are no longer any points left in the search space.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Image | Detection Samples | Number of iterations | Alpha (α) | Beta (β) | Angle weight () | Distance weight () | Time (s) | Precision | Recall | IoU |
| Fig. 1 | 2461 | 15 | 0.5 | 0.5 | 0.0 | 1.0 | 94.57 | 0.9913 | 0.9983 | 0.9897 |
| Fig. 2 |  |  |  |  |  |  |  |  |  |  |
| Fig. 3 |  |  |  |  |  |  |  |  |  |  |

*Initialisation*.

termination condition of the outer loop is simply

During the first iteration of the inner loop, the ants construct candidate rows, which followed by a pheromone update of the constructed paths.

## Stitching

Ideas

* Talk about local search as trying to find a better solution in some appropriately defined neighbourhood.
* Instead of keeping track of a global best, we keep track of the ants that form the global best. This is the selection from which the elites are chosen.

# Experimental Setup

Ideas

* See screenshot

# Results and Discussion

Sometimes jumped across gaps to produce lines longer than real lines.

# Conclusions

# Future Work

* Possibly incorporate confidence into future iterations of the algorithm.P

**Table 1**: Parameters and performance statistics for all visualised results in the paper.

# Appendix