A Novel Algorithm and Model for Autonomous Drone Pollination of Undomesticated Plants

William Bain, Abolfazl Razi, Manveen Kaur, Long Cheng, Jim Martin, Raul Amin

April 2023

1 Abstract

In this work, an algorithm is proposed and a model is developed for the use of a swarm of drones to pollinate plants. This algorithm is Abol: mention that the motion oath is driven by the online processing of drone visual observation /computer vision based on the mean-shift algorithm designed for clustering point data. An implementation of particle swarm optimization (PSO) and a density map influence algorithm are introduced to the mean-shift algorithm Abol: to do what? to solve what problem? to enhance what objective?. The mean-shift algorithm with and without PSO and density map influences are tested. Ultimately, it is found that the algorithm with PSO and with density map influence both outperform the algorithm without these influences Abol: by what margin, what is the gain we obtained?.

Abol: We can finalize the abstract at last, but some general comments/thoughts on the abstract: 1- As Dr. Cheng said better to put a few words on what is the missing gap, why current methods of pollination are not efficient, and what we are trying to achieve. 2- mention what is the inspiration/idea behind using PSO and density map. 3- we would need to provide more quantification of the improvements, for example, saying that we cover the area with x% of the time required if we would use bind (horizontal) sweeping, etc.

2 Introduction

Abol: Some general thoughts about the paper, and I included detailed comments on the body of the paper: better to include some comparative results with respect to other methods. Finally, let's mention constraints and requirements, like not allowing drones to share high-throughput imagery and only allowing them to share lightweight information. Another good direction is using a simpler computer vision algorithm instead of the more complex DL-based Yolov object detection and obtaining a soft density map instead of distinct point objects and then using it to pollinate the flowers. Also, as we discussed leveraging the results of previous years [at least to initialize the algorithm] can be advantageous.

In recent years, populations of natural pollinators have declined. There is an emerging body of research on using drones as artificial pollinators in controlled environments Abol: what is controlled environment. These drones help pollinate in agricultural settings when natural alternatives such as honeybees are unavailable Abol: highlight more advantages for using drones compared to other methods, things related to cost, covering hard-to-reach areas, etc..

This technology has been used in places like greenhouses to pollinate flowering plants such as strawberries (Se-young, 2019). Polybee, a company that specializes in autonomous drone pollination, has shown the scalability of this approach. Their technology focuses on using computer vision algorithms to automate the task of pollination, even going so far as to construct 3D models of plants and their distribution throughout a greenhouse down to centimeter precision (Polybee, n.d.). While this is certainly impressive technology, it goes to show how crucial it is that these algorithms are used in controlled environments.

Much of the research into artificial pollination using drones doesn't account for the need for pollinators in natural environments. With natural pollinators struggling, this could cause many plant populations to dwindle as well (Rhodes, 2018). Additionally, even when natural pollinators are abundant, they tend to prefer pollinating invasive species over native ones. This helps invasive species overtake and out-compete native ones, decreasing biodiversity. This could be mitigated with artificial pollinators, as they can specifically target native species for pollination with computer vision algorithms (Caverly, 2018). As such, it is important to study the possibility of artificial pollination in more natural settings.

The purpose of this work is to simulate pollination in such a natural setting. A field of flowers is modeled as a series of generated points on a plane, and models of drones are made to find a path to pollinate those flowers. The model of the field evolves over time, as a real field would change from month to month or year to year, and the drone swarm uses data about the field from the past to inform its path planning in the present.

The Abol: rest of this paper is organized as follows. The details of this model are discussed in section 3. In section 4, the data that were obtained are presented. In section 5, the results pertaining to this data are discussed. In section 6, the effectiveness of this model is discussed along with future work.

Abol: The introduction is informative and fluent. It would be better to add a few more related work ....

3 System Model

We assume that the wind created by the blades of a drone has an effect radius Abol: give a notation like R, and that any plant within that radius is being shaken. We also assume that each plant shaken for a certain amount of time, Abol: denoted by t_c can be considered fully pollinated. This is the time at which shaking a flower anymore with wind would be no more effective at spreading pollen to its neighbors.

The field over which the drone swarm is pollinating is modeled as Abol: a rectangular/square area with sides

and

an array of points. These points are generated Abol: randomly with the Scikit-learn make_blobs method (Pedregosa et al., 2011) Abol: better to say according to what distribution. Abol: to obtain a non-homogeneous set of points that mimic natural plant fields. An example of what a distribution of the flowers looks like can be seen in Figure 1.

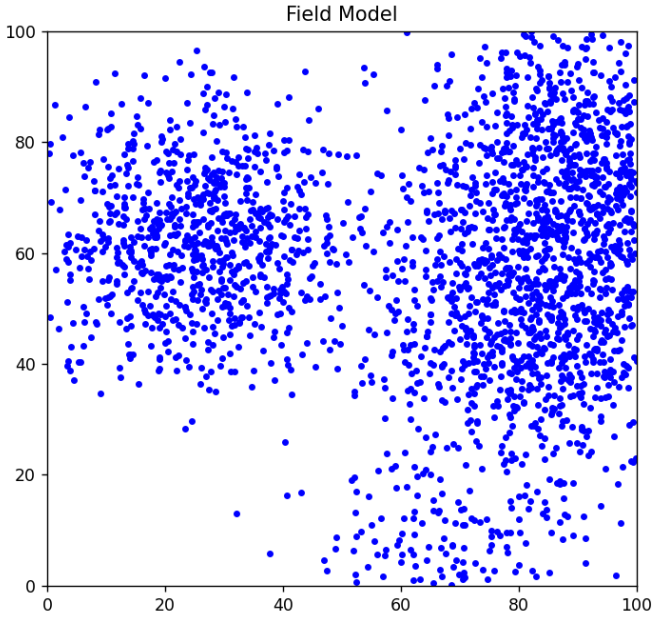

Figure 1: An example of a model of a field of flowers.

The field model also changes with time. The field is pollinated in generations. After each generation, a certain percentage of flowers are inserted, a percentage are deleted, and a percentage are moved. The flowers that are deleted from the model are selected randomly Abol: with some probability P_d: In general, it is better to use notations for most parameters if we will use them in our analysis or results. The flowers that are moved are then selected randomly from the flowers left after deletion. A random distance is generated for each flower to move. The distance Abol: follows a uniform distribution over the range $[0, R_m]$.Each flower moves in a separately generated random direction in the xy-plane. The flowers to be inserted into the model are generated similarly to the original randomly

generated field. They share the cluster centers with the original field. The intention of this evolution is to model a drone swarm pollinating a natural field of plants across multiple months or years, Abol: where there is some level of temporal correlations between the fields. In other words,. While a field may not look exactly the same from one time period to the next, it is likely to share at least some common locations for its plants.

By setting the percentages of flowers that are removed from, moved in, and added to the field, the user can decide how much in common one generation shares with the next.

The drone swarm used to pollinate this field is also modeled as a Abol: setof points. Each drone is given a position and a velocity vector. Each drone has two radii. One is for the area of flowers which it can shake with the wind from its blades, denoted R_e. One is for the area of flowers which the drone can see in its camera, denoted R_c. Each flower that is seen by a drone has its position stored in a dictionary. This dictionary maps the flower's position to the amount of time it has been shaken by a drone. This dictionary is shared across the swarm, and is not unique to each drone. It is assumed in this model that data can be shared between drones in a swarm without error Abol: This is the place that we can consider imperfect networking by saying x% of information is subject to error for this paper or the extension of this work!!!.

Abol: We can say something about the fact that the flower positions are not priory known to drones, rather they are extracted by computer vision [online processing of aerial imagery], then we can include some detection error by including false negative and false positive. For instance, if $X = \{x_1, x_2, \cdots, x_N\}$ is the set of original points, we can have $Y = \{y_1, y_2, \cdots, y_M\}$ be the set of detected pints where the intersection of two sets is $X \cap Y = ((M+N)/2).(1-P_e)$ where P_e is detection error.

This simulation updates all of its data in frames Abol: not clear, what do you mean. As such, the refresh rate of the simulation is denoted f.

3.1 Mean-Shift Algorithm

Abol: put a few sentences about how the Min-shift algorithm works and why we used it as the core of our algorithm.

Each drone's movement is informed by the mean-shift sliding window algorithm. Flowers that have been shaken for less total time are prioritized more than those that have been shaken for longer. As such, each drone's velocity is calculated as follows.

$$\vec{V_j} = \left(\frac{\sum_{R_c} w_i * \vec{r_i}}{\sum_{R_c} w_i} - \vec{r_j} \right) * f, \tag{1}$$

where, the velocity vector of the jth drone is denoted as $\vec{V_j}$. The positions and weights of all flowers within the radius R_c of the position of the drone, $\vec{r_i}$, are considered. The positions of the ith flower is denoted as $\vec{r_i}$. The priority of the ith flower is denoted as w_i. Each flower's priority is calculated as follows.

$$w_i = max(t_c - t_j, 0) \tag{2}$$

Here, the t_c denotes the time for the flower to be considered fully shaken, whereas t_j denotes the time that the jth Abol: better to use notations like j^{th} here and everywhere flower has been shaken. In this way, flowers that have been shaken for a time of t_c or longer have a priority of 0, and aren't considered at all in the calculation of the drone's velocity. Abol: This approach is very interesting, just a question. Do you allow drones to move away from a flower before the completion of the pollination, so other drones come toward them (with Prob. proportional to the remaining time $t_c - t_j$) to complete the pollination? if this is the case, the drones need to share how much they pollinated any flower, and such a table should be shared among them. Anyway, it is better to clarify this.

As such, each drone attempts to move toward the weighted average position of the flowers that it sees in its camera, using the priority of each flower as the weight of its position.

Under this algorithm, when $\sum_{R_c} w_i = 0$, or when the drone sees no flowers with any priority in its camera, it will continue with a constant velocity until it finds a flower with some priority, or until it reaches the edge of the area designated as the field. Abol: again, a good approach, but we can also program the drones to bounce back from the edge and continue flying until not observing a plant for a predefined time, to make sure the mission is not completed earlier than necessary. This would be useful for small areas where the drone can quickly reach the side of the square.

On its own, this methodology of determining the drone's velocities has them moving until they find a dense area of flowers, and then being still until that area is considered fully shaken. Once several drones have discovered a cluster of flowers, they do well in exploring and pollinating all of the flowers in that cluster. However, if one watches the drones through the animation functionality of the simulation, one will notice that the drones spend a large portion of time wandering around areas which either have no flowers in them, or in which the flowers are fully pollinated. For this reason, Particle Swarm Optimization was introduced to solve this problem. Abol: Good reasoning, we can also elucidate this better by saying that we are seeking algorithms where agents do not decide merely based on their local observation, but rather by the global information obtained by others, and then we can proceed by saying that PSO has such feature since it blends direction to local optima with the direction to global optima so far found by other agents.

3.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) was introduced as follows. Each drone has a bias in its velocity towards a weighted mean position of all the drones. This is calculated as follows. The total priority of the flowers that a given drone sees in its camera is found as follows.

$$W_j = \sum_{R_c} w_i \tag{3}$$

The total priority seen by a drone is just the sum of the priorities of each flower seen in its camera. So the priority seen by drone j is denoted as W_j. The average weighted position of all the drones is denoted as $\vec{R_{PSO}}$ and is found as follows. If no drone has any flowers with any priority in its camera (that is, if $\sum_j W_j = 0$), PSO won't be considered for that frame of the simulation.

$$\vec{R_{PSO}} = \frac{\sum_j \vec{R_j} W_j}{\sum_j W_j} \tag{4}$$

The velocity necessary to get from the drone j to the position $\vec{R_{PSO}}$ is denoted as $\vec{V_{PSO}}$. Mathematically, this is written as follows.

$$\vec{V_{PSO}} = (\vec{r_{PSO}} - \vec{r_j}) * f \tag{5}$$

The bias that each drone has towards the position $\vec{R_{PSO}}$ is denoted as α. As such, the velocity of the drone j after the PSO influence is calculated as follows.

$$\vec{V\prime_j} = (1 - \alpha)\vec{V_j} + \alpha \vec{V_{PSO}} \tag{6}$$

3.3 Density Map Influence

Abol: The afore-mentioned algorithm uses the current observation of the field that is obtained by the swarm of drones with computer vision algorithms. This is ignorance of the information which is readily available from the last test, noting that there is a high correlation between the test field in consecutive pollination phases. In this section, we propose a method that utilizes the previous observation to accelerate the collective search algorithm. Obviously, the stronger the correlation, the more advantage would be attainable for using prior information.

After each generation of pollination, a density map is created using the location of every flower that the swarm discovered in previous generations. This is done using the Scikit Learn's KernelDensity class (Pedregosa et al., 2011) Abol: I would suggest that we can say we used Scikit library for our simulation at the beginning of the results section, and skip mentioning individual classes throughout the paper. An example of what this density map might look like can be seen in the following figure.

For each drone, a circular mask was applied to this density map centered at the drone's location and with a specified radius around it. This radius is denoted R_d. I suggest we put a table of notations at the beginning of the "3. System Model" for readers' convenience. An example of a masked density map can be seen in Figure 3.

The point of highest density in this masked density map is then found using the Numpy argmax function. We will denote this position as $\vec{R_D}$. Similarly to the PSO implementation then, the velocity for the drone j is as follows.

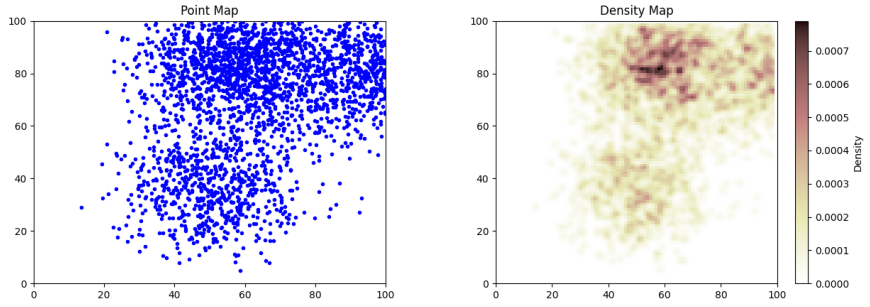

Figure 2: This is an example of what a density map may look like. Abol: better to use more formal language like "illustration of an exemplary density map. Left: YYY, Right: ZZZ..." On the left is a point map wherein the points represent locations of flowers. On the right is a density map generated from those points. Note that every point was used in the generation of the density map.

$$\vec{V_D} = (\vec{r_D} - \vec{r_j}) * f \tag{7}$$

The drone will move towards this densest position by applying a certain bias to its velocity. That bias is denoted as β. So drone j's velocity after considering the density map is calculated as follows.

$$\vec{V\prime_j} = (1 - \beta)\vec{V_j} + \beta\vec{V_D} \tag{8}$$

3.4 Other Considerations

If a drone reaches the edge of the area designated as the field, it moves away from that edge at the reflection angle of the velocity it had when coming to that edge.

When the camera radius, R_c, of a drone is larger than the effect radius, R_e, an issue can occur wherein the drone stops moving. This occurs when three conditions are met. The first is that the drone has stopped because it has reached the mean weighted position of the flowers seen in the camera. The other two are that R_e has no flowers of any priority in it, and that R_c does. Examples of what this situation might look like can be seen in Figure 4. The solution to this was to have each drone detect when these three conditions were met, and then only consider the flowers in R_e for path planning for a certain amount of time. Mathematically, when this happens, drone j's velocity is calculated as follows.

$$\vec{V_j} = \left(\frac{\sum_{R_e} w_i * \vec{r_i}}{\sum_{R_e} w_i} - \vec{r_j} \right) * f \tag{9}$$

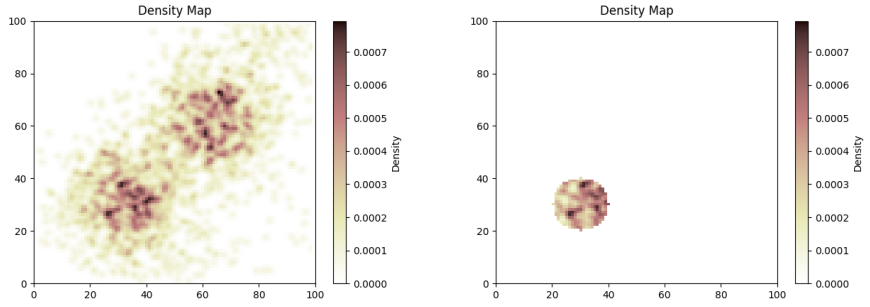

Figure 3: Abol: again: use a more formal language An example of what a masked density map might look like. On the left is the original density map. On the right is the density map after a circular mask of radius 10 centered at (30,30) has been applied to it.

This change in the way that the drone's velocity is calculated lasts for a time that is set by the user. This time is denoted as t_r. Note that this is not an issue when $R_c = R_e$.

Because each drone is searching for the densest area of flowers, the drones in a swarm have a tendency to cluster in the same area. This is especially true with higher α values for PSO, as the drones all have a bias to move toward one location. As such, collision avoidance was introduced as follows. A safe distance for drones to be apart from each other is assigned to the swarm. This distance is denoted R_s. Each drone broadcasts its position to each other drone. If drone A finds itself within $4R_s$ Abol: can be some constant times R_s or simply R_s to be more general, and then we tune f of drone B, a vector is generated pointing from drone A directly away from drone B as follows.

$$\vec{r_{CA}} = (\vec{r_A} - \vec{r_B}) * f \tag{10}$$

This vector is then added to the drone's velocity with a magnitude proportional to the inverse of the distances between the drones squared. Mathematically, the drone's velocity is calculated as follows.

$$\vec{V\prime}_j = \vec{V}_j + \frac{\vec{r_{CA}}}{\|r_{CA}\|^2} * f \tag{11}$$

If drone A finds itself within R_s of drone B, the drone's velocity is calculated as follows.

$$\vec{V\prime}_j = \frac{\vec{r_{CA}}}{\|r_{CA}\|^2} * f \tag{12}$$

In this way, the drones will have some bias toward moving away from each other when they are within $4R_s$ of one another. This bias grows as the drones

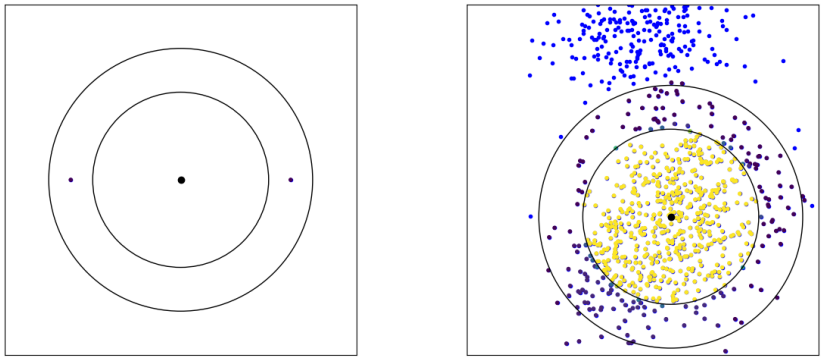

Figure 4: These are examples of a situation in which a drone would get stuck calculating its velocity using equation 9. On the left is a toy example meant to show how this problem can occur. Both flowers are within the radius R_c, but neither are within R_e. The drone is at the mean position of the flowers in the camera, so it does not move. On the right is a more realistic example of how this might happen. Note that the flowers within R_e are all fully shaken, so they have a priority of $w = 0$. As such, it sees flowers with nonzero priorities in the radius R_c, but none of them are within R_e. Abol: nice illustration!

move closer together. If the drones are within R_s of one another, they no longer even consider their previous velocity Abol: Interesting idea, but we may not need to make it two cases one $< 4R_S$ and one $< R_s$ since the velocity vector automatically grows as they get closer. I mean there is no need to make it over-complex and adding a velocity proportional to the inverse of their distance with an appropriate weight would do the job!. Their only priority is to move way from one another.

The swarm is assigned a maximum speed shared by all of the drones. This is denoted as V_{max}. This value is common for all of the drones in a swarm. During each frame of the simulation, if the magnitude of a drone's velocity is found to be very small (less than $1 \times 10^{-3}\,\mathrm{m\,s^{-1}}$), and W_j is also found to be very small (less than $1 \times 10^{-3}\,\mathrm{s}$), the drone's velocity is adjusted as follows. This adjustment is made so that the drones don't spend a lot of time where there are no flowers of nonzero priority moving slowly. While it's okay for the drones to leave areas of high flower density to search for other flower clusters, it would be inefficient for the drones to be moving slowly while doing this. This adjustment also occurs if the drone's velocity exceeds V_{max}.

$$\vec{V_j} = V_{max} \frac{\vec{V_j}}{\|V_j\|} \tag{13}$$

4 Results

This is a simulation with many adjustable parameters, such as the field size, the number of flowers, the number of drones in a swarm, etc. As such, when testing to optimize certain parameters, many other parameters needed to be held constant. Realistic estimates for what these parameters should be were made. The size of the field to be pollinated was 100 m by 100 m in each trial. The maximum distance that flowers could move between generations was set to $R_m = 5\,\mathrm{m}$. The time for a plant to be considered fully pollinated was set to $t_c = 5\,\mathrm{s}$. The distance at which drones should be apart for their safety was set to $R_s = 1.5\,\mathrm{m}$. The refresh rate of the simulation was set to $f = 50\,\mathrm{s}^{-1}$. The maximum speed for drones in the swarm was set to $V_{max} = 5\,\mathrm{m\,s}^{-1}$. The time for which a drone reduces the radius of flowers it uses for path planning if it has stopped moving was set to $t_r = 0.5\,\mathrm{s}$. Drones were given starting positions equally distributed across the Southern border of the field which they were pollinating. Their starting velocities had them all moving directly Northward at a speed of V_{max}.

4.1 Particle Swarm Optimization

The PSO algorithm described above was tested with several α values varying by order of magnitude. These were each tested on 10 different randomly generated fields for 1 generation. The swarm was given 3 minutes to pollinate the field for the single generation. Each of these tests took place with a swarm of 10 drones. R_c was set to 3 meters, and R_e was set to 2 meters for the drones in each of these tests. The minimum between the time spent pollinating a given flower, t_i, and the maximum useful time spent pollinating a flower, t_c, was summed for each flower i. This was compared to a maximum pollination given by multiplying t_c by the number of flowers, denoted as N. As such, we get the pollination percentage, P, for a given trial as follows.

$$P = \frac{\sum_i min(t_i, t_c)}{t_c * N} * 100\% \tag{14}$$

The pollination percentage was taken for each α value for each field. The results of the tests across each field were averaged according to their α values. The results of these tests can be seen in Figure 6.

The value of $\alpha = 10^{-3}$ had the highest average pollination percentage across 10 trials using randomly generated fields. This value was then tested against the mean-shift algorithm without PSO. The results of this test can be seen in the figure below.

As can be seen in Figure 6 algorithm with PSO at a bias of $\alpha = 10^{-3}$ outperformed the mean-shift algorithm without PSO in every trial. Abol: mention the gain!

Abol: here we showed that our algorithm when uses PSO outperforms the one without using PSO, but before this it is imperative to show that our algorithm outperforms baseline algorithms like horizontal and spiral sweeping. You

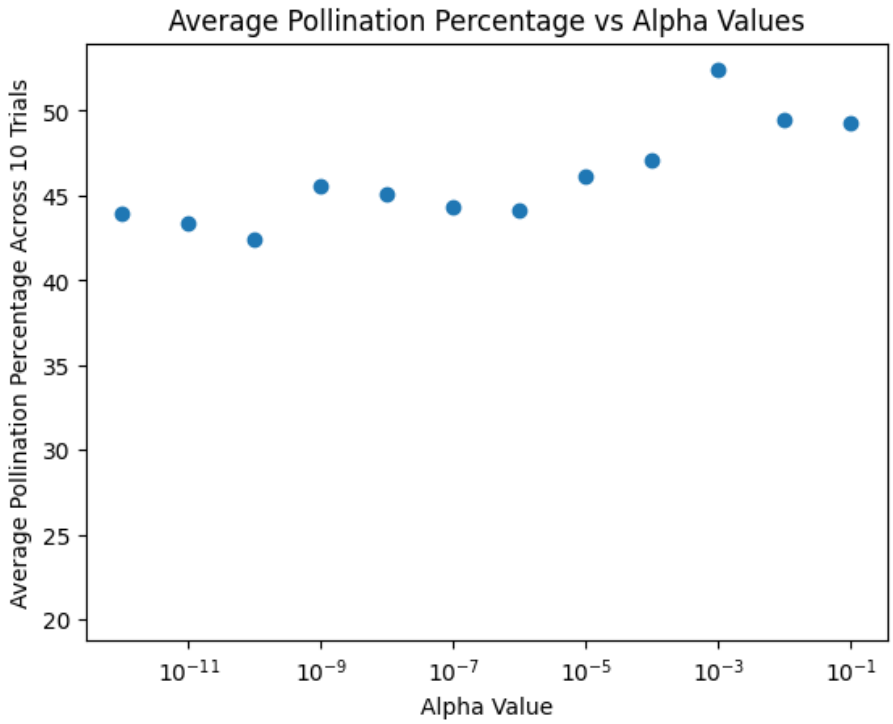

Figure 5: The average pollination percentage across 10 different randomly generated fields averaged by their alpha values. That is, the results for fields 1-10 using an α value of $\alpha = 10^{-5}$ were averaged to obtain a result for this α.

can generate scenarios where the same number of drones are used for pollination but instead of using your algorithm they are given the corner points of the filed and each of them sweeps part of the field, For example you can split the filed of size r^2 into N smaller filed of area r^2/n , then each drone sweeps that region, and then you can compare this algorithm against yours in terms of a) total time it takes to finish the job, b) the success rate in terms of the portion of the plants that are fully pollinated. This would be essential to show that our algorithm outperforms the baseline.

4.2 Density Map Influence

The Density Map Influence (DMI) algorithm described above was tested with several β values varying by order of magnitude. These were each tested on 10 different randomly generated fields. The simulations for each field and for each β value were run for two generations each (as the swarm would have no data to generate a density map from for the first generation). Each of these tests took place with a swarm of 10 drones. R_c was set to 3 meters and R_e was set

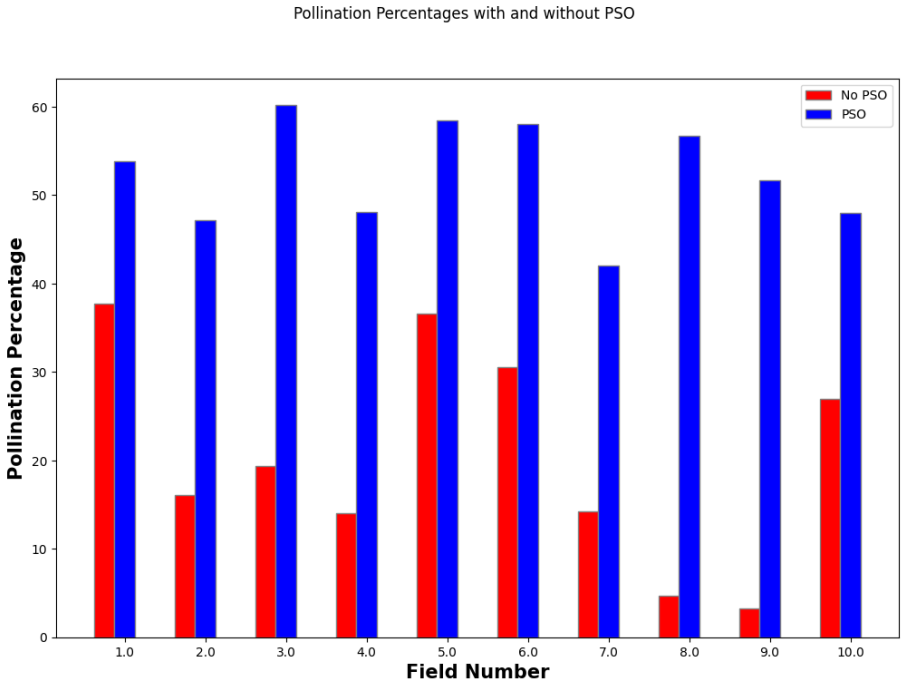

Figure 6: The pollination percentages for 10 different fields for the mean-shift algorithm without PSO and the mean-shift algorithm with PSO at a bias of $\alpha = 10^{-3}$. Abol: I like this figure!

to 2 meters for the drones in each of these tests. The minimum between the time spent pollinating a given flower, t_i, and the maximum useful time spent pollinating a flower, t_c, was summed up for each flower i. This was compared to a maximum pollination given by multiplying t_c by the number of flowers, denoted as N. As such, we get the pollination percentage, P, for a given trial as follows.

$$P = \frac{\sum_i min(t_i, t_c)}{t_c * N} * 100\% \tag{15}$$

The pollination percentage was taken for each β value for each field. The results of the tests across each field were averaged according to their β values. The results of these tests can be seen in the following figure.

The value of $\beta = 10^{-11}$ had the highest average pollination percentage across 10 trials using randomly generated fields. This value was then tested against the mean-shift algorithm without DMI. The results of this test can be seen in the figure below.

The algorithm with DMI at a bias of $\beta = 10^{-11}$ outperformed the mean-shift algorithm without DMI in nearly every trial. The exception to this is in field 6,

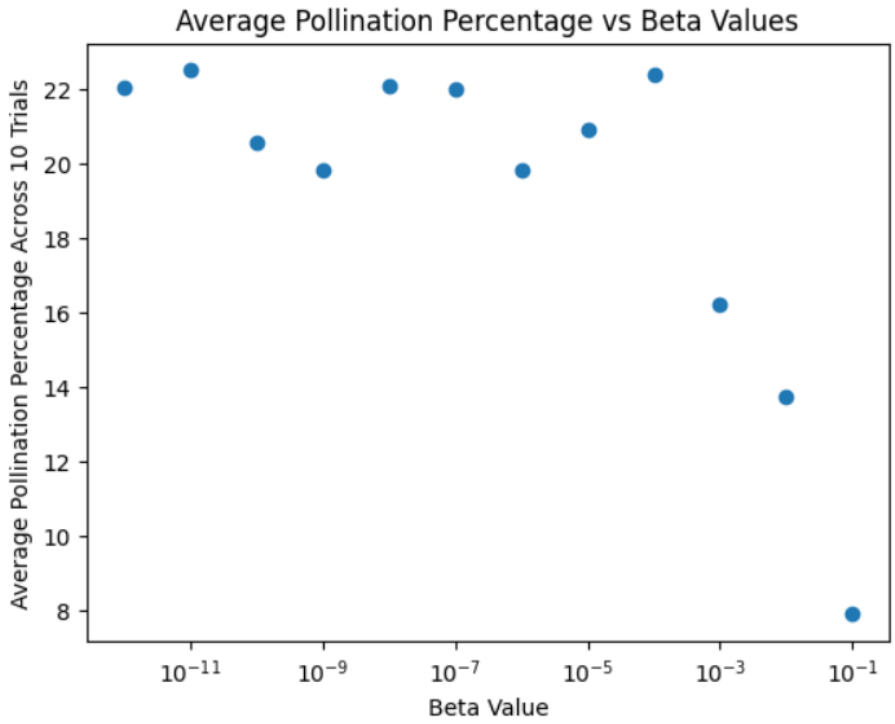

Figure 7: The average pollination percentage across 10 different randomly generated fields averaged by their β values. That is, the results for fields 1-10 using an β value of $\beta = 10^{-5}$ were averaged to obtain a result for this β.

in which the algorithm without DMI has a slightly higher pollination percentage than the algorithm with DMI.

Abol: the last two figures are very good, but you also show that leveraging prior test results is more advantageous when the temporal correlation is stronger. To this end, you can use different correlation or mismatch probabilities like 0.001, 0.01, 0.1, 0.2, ... to generate fields that are super close versus the ones that are less correlated, and then show that for the ones with higher correlations it amkes more sense to leverage prior information.

5 Discussion

Because of the novelty of this model, we aren't aware of results from other studies that these could be compared to. However, the results of using this algorithm with and without PSO and with and without DMI will be compared.

As seen in Figure 6, the implementation of the algorithm using PSO with a bias of $\alpha = 1 * 10^{-3}$ outperformed the algorithm without PSO in every trial.

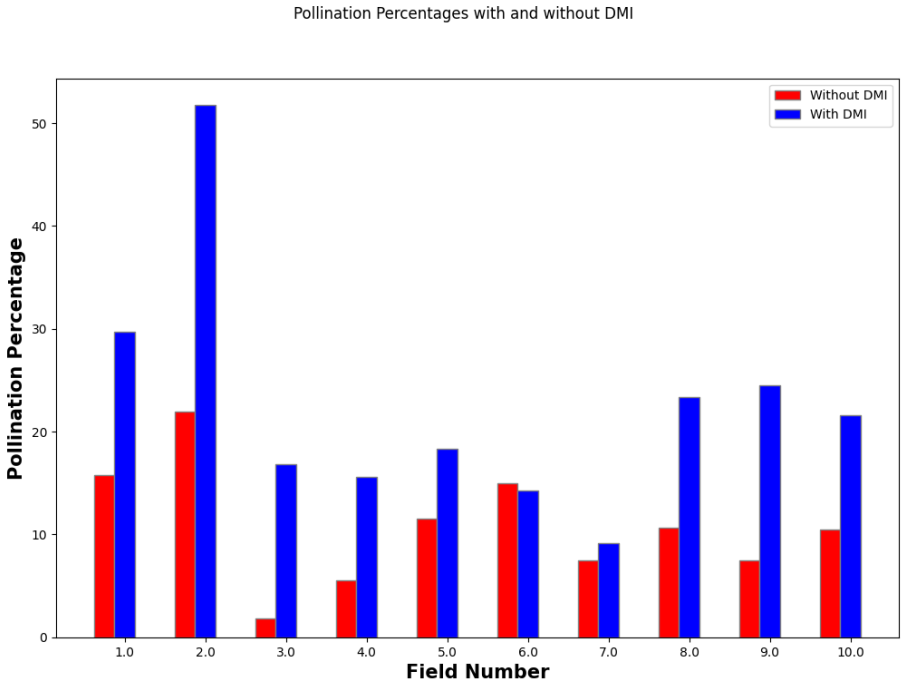

Figure 8: The pollination percentages for 10 different fields for the mean-shift algorithm without DMI and the mean-shift algorithm with DMI at a bias of $\beta = 10^{-11}$.

On average, the tests with PSO had a greater pollination percentage than those without PSO by a factor of 4.70. This is a dramatic increase. It can also be seen in Figure 8 that the increase in certain trials can be much more dramatic than this. In the trials performed on field 3, the algorithm with PSO outperformed the algorithm without PSO by a factor of 15.8. As such these tests demonstrate that the algorithm with PSO is better suited to the task of pollinating undomesticated plants than the algorithm without PSO.

As seen in Figure 8, the implementation of the algorithm using DMI with a bias of $\beta = 1 * 10^{-11}$ outperformed the algorithm without DMI in almost every trial. The exception to this is the test on field 6, in which the algorithm without DMI has a pollination percentage which was greater than the algorithm without DMI by a factor of 1.0508. Despite this, it is clear from looking at all of the trials that DMI generally increased the pollination percentages in this test. On average, the tests with DMI had a greater pollination percentage than those without DMI by a factor of 2.761. This is a dramatic increase. It can also be seen in Figure 8 that the increase in certain trials can be much more dramatic than this. In the trials performed on field 3, the algorithm with DMI outperformed the algorithm without DMI by a factor of 9.269. As such these

tests demonstrate that the algorithm with DMI is better suited to the task of pollinating undomesticated plants than the algorithm without DMI.

6 Conclusion

Abol: abstract and conclusion both are good, but we can update then after completing the rest of the sections. Ultimately, the model developed in this work shows that a path planning algorithm based on the mean-shift algorithm is effective at pollinating undomesticated plants. Adding particle swarm optimization or a density map influence algorithm to this increases that effectiveness.

One direction for potential work on this algorithm is to simply test it with more variables. As previously mentioned, there are a huge number of parameters that can be set in this simulation. While the biases for the PSO and DMI algorithms described above were varied and tested, much more data could be gathered by varying other parameters and holding these variables constant.

Other potential future work on this algorithm involves developing the model further. Notably, this model lacks a third dimension. Adding a third dimension to this simulation would add another dimension to the complexity of calculating drones' velocities. Topography data could be added to the field to study how the terrain effects the ability of the drone swarm to pollinate plants. Other considerations for modelling a swarm of unmanned aerial vehicles could also be added to the simulation. This could include things like obstacle avoidance, weather conditions or simulating computer vision in more detail. Another direction that future work on this algorithm could take is applying it on a swarm of real drones. This would involve training those drones to recognize images of a plant, and using them to pollinate that plant in a certain area. The effects of this pollination could then be studied from an agricultural or ecological perspective, looking at things like population sizes and quality of the plant.

References

Caverly, W. (2018, May 21). Synthetic pollination invasive species. *Bee Culture*. Retrieved from https://www.beeculture.com/synthetic-pollination-invasive-species/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Polybee. (n.d.). *Technology.* https://www.polybee.co/technology.

Rhodes, C. J. (2018). Pollinator decline – an ecological calamity in the making? Retrieved from https://journals.sagepub.com/doi/pdf/10.3184/003685018X15202512854527

Se-young, P. (2019, January 28). *Scientists use drones to artificially pollinate strawberries.* Arirang News. Retrieved from https://www.arirang.com/news/view?id=195454