Angry Ants: Final Report

Jasmin Uribe and Yunhao Xu

May 7, 2013

Abstract

Analyzing the motion and behavior of social insects requires tracking many individuals over time. Automated tracking is unreliable and inefficient. Ants frequently interact with each other which can cause the algorithm to begin tracking a different ant. The high density of ants being observed also increases the complexity of automated tracking. The authors of [5] use a "citizen scientist" model where volunteers track ants. A representative trajectory is then extracted for each ant using Frechet average and median trajectory. The authors propose a global approach in which all ant trajectories are considered simultaneously. In this paper we construct a trajectory graph for data collected by citizen scientists and explore a global greedy approach for extracting representative trajectories. We also explore the sensitivity of this global approach to changes in input parameters using generated data. We measure the average variance in pixels of each representative trajectory from the ground truth. Our results demonstrate that for the collected data, the global greedy approach works well; matching if not out-performing the automated solution. We also demonstrate that the global greedy algorithm is sensitive to probability of user error and the clustering threshold.

1 Introduction

Biologists are exploring the social dynamics of ant colonies in order to better understand social organization in animal groups. In studying ant colonies, data is collected by tracking ant motion and interactions. Following each individual ant manually is impractical and time consuming. Automated tracking algorithms are unreliable and inefficient. Ants interact frequently by meeting, touching or crawling over each other. This makes tracking an individual ant difficult for an automated system.

The authors of [5] have created a system that allows volunteers to track ants in videos in order to apply the "citizen-scientist" model to this problem. Users build a trajectory by clicking on a specified ant at every time step. Once enough data is collected an 'average' trajectory is extracted using a "local" approach: a representative trajectory for each ant is computed using either a Frechet average or median trajectory. The authors then compare these representative trajectories to an existing automated ant tracking system and to the 'ground truth'. See Fig. 1.

A complementary "global" approach is described briefly in [5]. In this approach trajectories for all k ants are considered simultaneously. Citizen scientists often mistakenly switch from tracking one ant to tracking another when two or more ants intersect. The global approach is motivated by this phenomenon; an ant trajectory could contain valid portions of another ants path. See Fig. 2.

Some work has been done on the global approach, in [6], several algorithms, both local and global, were compared using collected data. Here, we explore the representative trajectories extracted by a global greedy algorithm. We then compare the resulting trajectories to the 'ground truth' and measure the average variance. In order to further test the quality of the algorithms, we generate pseudo-random ant paths, simulate user data, then extract representative trajectories and compare to the actual ant paths. We explore the sensitivity of the global greedy algorithm to input parameters.

This paper is organized as follows. Related work is described in Section 2. Section 4 contains the graph generating algorithms. Section 5 details the collection and generation of data. The experiments are described in section 7. We discuss results and describe future work in sections 8 and 9.

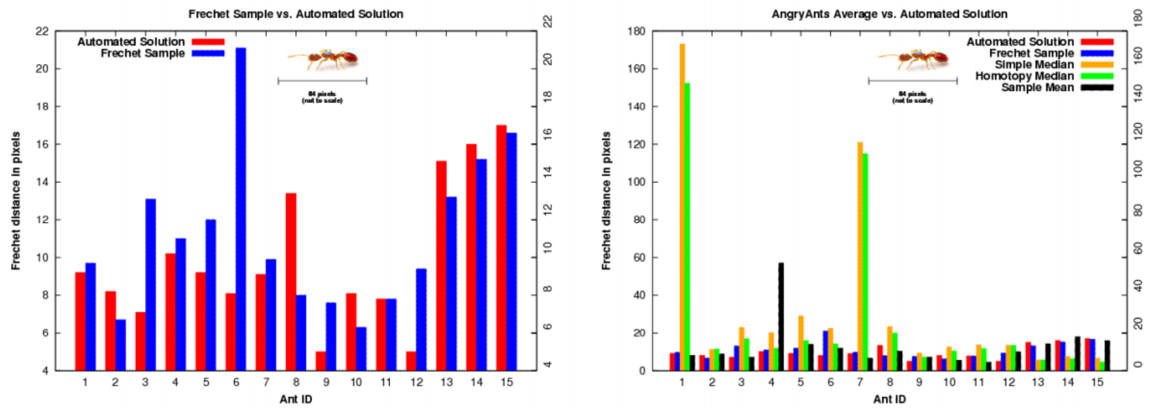


Figure 1: Frechet distance between trajectories and 'ground truth' [5]

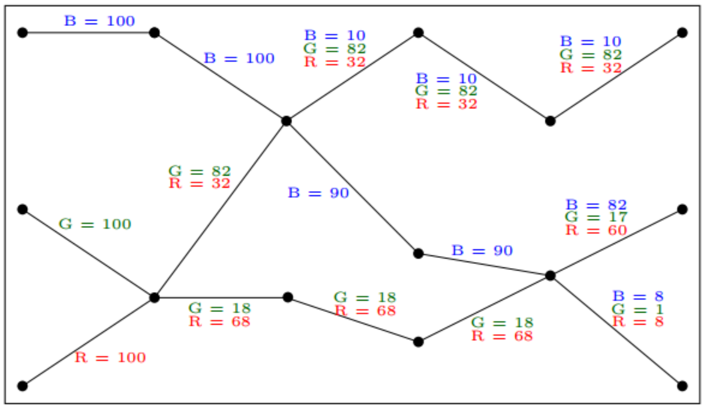


Figure 2: Trajectory graph G [5]

2 Related Work

Object tracking is defined as the problem of estimating the trajectories of certain objects in an image as they move around a defined space. Other relevant data such as orientation, shape and area of an object may be collected as well. Tracking objects is complex in many ways due to noise in the images, complex object motion, illumination changes and loss of the third dimension in 2D images, to name a few. Many approaches have been proposed for this problem, each addressing a slightly different issue.

[13] by Yilmaz *et al.* provides a survey of tracking methods grouped into broad categories. The purpose of the paper is to provide a library of tracking algorithms allowing readers to select the most appropriate based on their needs. They discuss the importance of object representation, object appearance, feature selection, object detection, background subtraction, and image segmentation. Different types of object tracking- point tracking, kernel tracking and silhouette tracking- are also discussed, as well as which algorithms are best suited to each.

The authors of [9] present a method for tracking bees that uses static and adaptive appearance templates as well as "geometry-constrained resampling of particles for handling appearance change." Tracking bees, much like tracking ants, is difficult since the objects are all very similar and the appearance of an object can change over time. With bees, the placement of the wings, the orientation of the bee, and the uneven lighting affect how the bee will appear in the image. Appearance templates provide a more complete description of the appearance, making the algorithm more sensitive to specific features of the bee. The added resampling step improves the prediction of head and thorax orientation. Together these modifications improve tracking when compared to a particle filtering based tracking with Gaussian modeling of appearance.

In a more recent paper, the authors of [8] present a method for tracking multiple ants. They "prevent drifts by maximizing the coverage of foreground pixels at a global scale" and "improve speed by reducing Markov chain length through dynamically changing the target proposal distribution for perturbed ant selection." They used the fact that in many cases the number of ants being tracked remained constant for a certain period of time combined with the assumption that all ant pixels are tracked to reduce the algorithm erroneously switching from one ant path to another. They also increased efficiency by using a "variable target proposal distribution." Not all ants move at every time step, so instead of using a fixed distribution they dynamically vary the distribution based on the level of expected motion. These changes improve both accuracy and speed.

One of the major problems with automatic tracking methods is drifting, which occurs when the tracker loses the object it is tracking. In 2012, [11] addressed this problem by enabling a user to examine a fraction of the entire sequence for validating and correcting mis-trackings. Frames containing drifts were identified "based on the amount of un-tracked foreground pixels;" this modification improved automated tracking by 25%.

Markov chain Monte Carlo based particle filters were used by [14] to effectively deal with interacting targets whose behavior is dependent on its interactions with others. They propose a model that maintains the identity of the objects throughout an interaction in order to reduce drifting. The algorithm is based on the assumption that certain objects do not behave independently, but rather an encounter between two objects can result in very different behavior. This paper demonstrates how "a Markov random field motion prior, built on the fly at each time step, can adequately model these interactions."

Once the data has been collected, using an automated tracking algorithm or otherwise, it must then be analyzed. There are two general approaches: the local approach, in which we consider individual trajectories, and the global approach, in which we consider all trajectories. Some research has been done on determining an average path given a data set of many paths and much work has been done on averaging polygonal paths. In [10], they present an approach for finding recreational trails using aerial images and maps. They assume the two end points of the trail are given, then find the most likely trail between the points. They estimate the likelihood that an observed piece of an image is on the trail using three different models. They then integrate this approach with a more global approach by noting that pieces of a trail must link up to connect the start point with the end point.

The authors of [3] "establish criteria that a 'median trajectory' should meet" then present methods of computing said median. Two of the properties of a median trajectory are that any point, p, on the median must also lie on an input trajectory and the minimum number of distinct trajectories p must cross to reach the unbounded face is $\frac{m+1}{2}$, where m is the number of input trajectories. The authors of [5] use this definition of a median trajectory in one of the local approaches.

When considering the local approach, it is necessary to define a measure of how close one path is to another. In many papers the Frechet distance is used. In [7] a practical approximation algorithm is presented for the Frechet distance between two polygonal curves. The Frechet distance can be thought of as the maximum distance a point on the first curve has to travel as this curve is being continuously deformed into the second curve. A new class of curves for which the Frechet distance can be approximated quickly is introduced.

The authors of 'Matching Planar Maps' by *Alt et al.* [1] present an algorithm "for measuring the similarity patterns of line segments in the plane." They consider a polygonal curve and an embedded graph with line segment edges. The goal is to find a path in the graph that minimizes the Frechet distance between the path and the curve. This idea is modified in [4] where exact algorithms are determined for partial curve matching using Frechet distance. Here, they measured partial similarity between curves. The goal was to maximize the total length of sub-curves that are close to each other. They present a polynomial-time exact algorithm for computing this maximum.

Currently there is experimental work being done on a linear programming approach to solve the global network flow problem [6]. In this case, weights are assigned to each edge for each ant. These weights should add up to 1. The goal is to maximize the $(weight) * (cost)$ over all edges and all ants. For each vertex, the sum of incoming weights of each ant should equal the sum of outgoing weights of that particular ant. The conjuncture is that for each edge the weight of one ant will be 1 and the weight of all other ants will be 0. Most recently a counterexample has been found that demonstrates a case in which an integer solution will never be reached.

3 Our Contribution

We explore the global greedy algorithm of extracting a representative trajectory for each ant. We demonstrate that this algorithm is sensitive to changes in the probability of user error and the clustering threshold.
We define a *trajectory* as a sequence of ordered triples $(t_1, x_1, y_1), (t_2, x_2, y_2), ..., (t_N, x_N, y_N)$ where x_i is the x-coordinate and y_i is the y-coordinate of an ant at time t_i. We connect these points with straight lines, assuming ants move at a constant speed between time steps. Let $\tau_1, ..., \tau_n$ a collection of trajectories. We assume each trajectory tracks one of k ants and each ant is tracked at least once. We assume each trajectory has the same number of points. We also assume the starting point of each trajectory is certain. Given this collection of trajectories we wish to compute a representative trajectory for each ant. We define a representative trajectory as the most likely path taken by each ant.
We compute this representative trajectory using a global greedy algorithm. We use a hybrid clustering algorithm to first create a global graph then use a greedy algorithm to determine the most likely path for each ant. Recent work has determined that this problem is NP-hard, but our global greedy algorithm performs well for collected data when compared to the automated solution.
We generate pseudo-random ant paths then simulate user data. We run the global greedy algorithm on this generated data and explore the sensitivity of the algorithm to the threshold clustering value and the probability of user error.

4 Creating the Global Trajectory Graph

In the global approach we consider all input trajectories simultaneously. We do this in order to capture all valid trajectory pieces. When ants a and b meet, citizen scientists often mistakenly switch from tracking ant a to tracking ant b. Although the entire path is no longer correct for either ant, it still contains valid portions for both ants. We take advantage of these portions in the global approach. Given a collection of trajectories we extract a representative trajectory for each ant by first creating the global graph G then extracting representative trajectories using a greedy approach.
At each time step, our data is a collection of observations $\{(t, x_i, y_i, a_i)\}$ where t is the time step, a_i is the ant ID and $i = 1$...total number of users. We use cluster analysis to group high concentration areas and pick the centroid of each group as a vertex of the global graph. For each cluster we know the proportion of observations corresponding to each ant. We connect vertices in two consecutive time steps if for these two vertices there exists at least one ant for which the proportion of observations in both vertices is non-zero. See Figure 2.

4.1 k-means++ Clustering

K-means clustering algorithm is one of the well-known clustering algorithm that aims to partition n observations in K clusters in which each observation belongs to the nearest cluster. However, finding an exact solution to the k-means problem is NP-hard. We choose to find an approximate solution by k-means++ algorithm [2], which chooses initial values for k-means clustering in order to avoid poor clusterings found by the standard k-means algorithm. One of the short comings of standard k-means is that the approximation found can be bad with respect to the objective function when compared to the optimal clustering. k-means++ addresses this by initializing cluster centers before proceeding with the standard k-means optimization, and is guaranteed to find a solution that is O(log k) [2].
We run preliminary experiments on the k-means++ clustering, at a single time step, t. We analyze the clusters created by the data points. The position of the centroids is sensitive to the number of clusters, K. In Fig. 3, we can see how increasing K from 1 to 8 affects the locations of the cluster centroids. However, the problem with this approach is that it is hard to determine the number of

clusters *apriori* for one frame, since ants may intersect with others and reduce the number of the clusters.

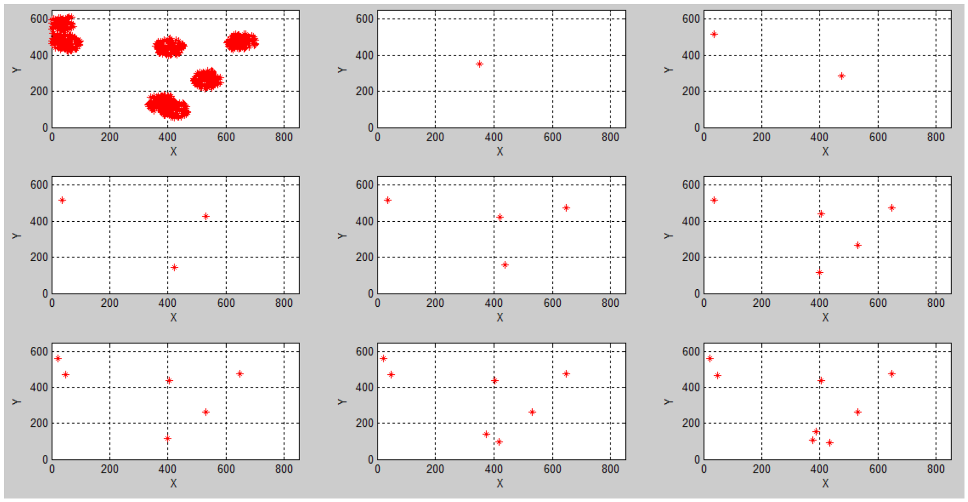


Figure 3: (a) presents all observation points for k-means; altering the number of clusters, K, from 1 to 8, which are presented in $(b) - (i)$.

4.2 SLINK Clustering

As described previously, for a certain time frame it is difficult to determine the exact number of clusters needed. We require an algorithm that can automatically determine the number of clusters. SLINK [12] is an algorithm that solves this problem. It treats each observation as a cluster initially and merges pairs of clusters in each iteration. In our experiment, we use the Euclidean distance as the distance metric and set up a threshold T to stop merging clusters if the distance is larger. Since we know the exact size, R, of each cluster, we compare the results produced by different thresholds, $R/2$, R, and $2R$. As shown in Fig. 4, different threshold values change the total number of clusters. A large threshold value, (d), yields only 4 centroids, while for lower threshold values the number of centroids increases. We notice that a threshold close to R provides the best result.

4.3 Hybrid Clustering

Our goal is to find at most N disks to cover all points at each time step, where N is the number of ants, and maximizes the number of disjoint disks. In this case, we implement a hybrid clustering algorithm that uses both k-means++ and SLINK clustering. We use SLINK to determine the value K, where $K \leq N$, disks which cover all data points. We initially place $K = N$ group centroids into the space using the k-means++ algorithm. We then compare the Euclidean distance between two centroids. If the distance between any pair of centroids is closer than the threshold, we merge these two clusters. This reduces the total number of clusters from K to $K-1$. We repeat this step until we cannot merge anymore clusters. Since we use the k-means++ algorithm to cluster our observations, the solution is guaranteed to be O(log k), competitive with the optimal k-means solution. In our experiment, we fixed the initial K value to 8 and alter the threshold T. From Fig. 5, we see that the cluster merging process reduces the number of clusters K. Although different thresholds may still produce different results, as described in section 4.2, we can set our threshold closer to the actual size of the observation clusters, which can be estimated initially, and thus avoid the threshold problem.

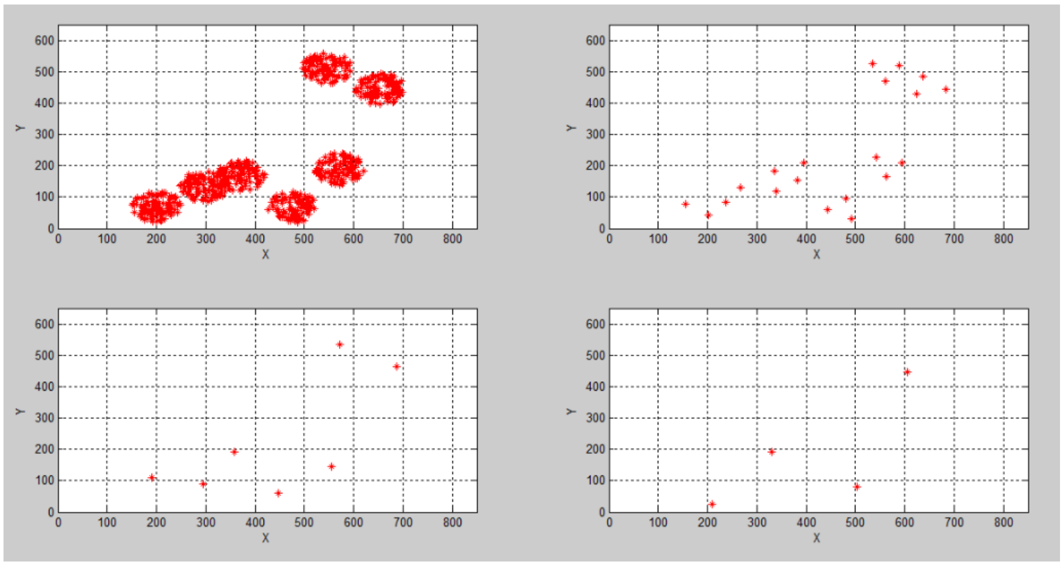


Figure 4: (a) present all observation points for SLINK, (b) presents the clusters found by set threshold $T = R/2$, (c) presents the clusters found by set threshold $T = R$, (d) present the clusters found by set threshold $T = 2R$, where R is the actual radius we used to generate these observations.

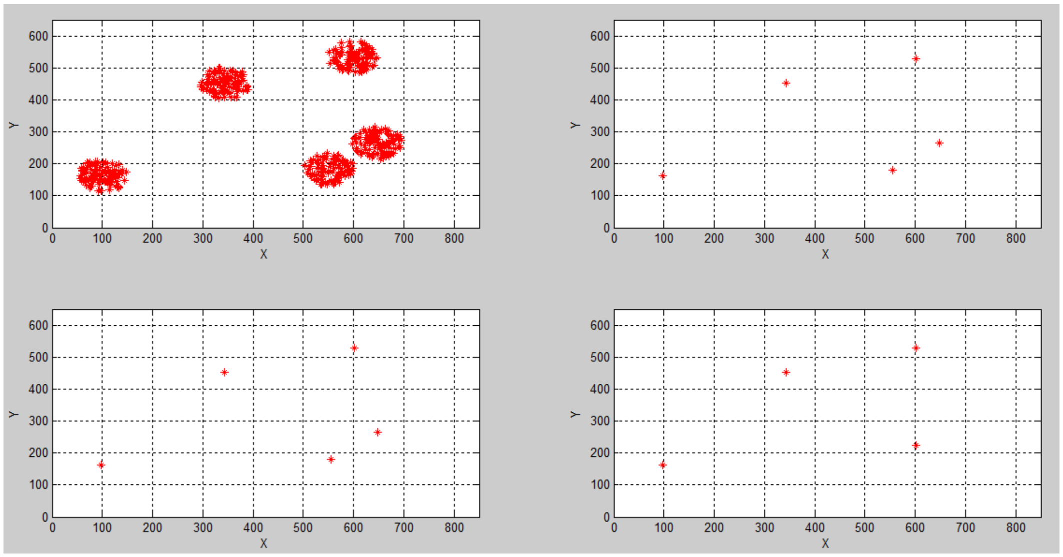


Figure 5: (a) present all observation points for our hybrid algorithm. We set the initial K value to 8: (b) presents the clusters found by set threshold $T = R/2$, (c) presents the clusters found by set threshold $T = R$, (d) present the clusters found by set threshold $T = 2R$, where R is the actual range we used to generate these clusters.

5 Collecting and Generating Data

In order to test the accuracy of the global approach we must first create a trajectory graph, G. We do this in two ways. The first uses simulated data, ie. simulated ant trajectories, and the second uses collected data.

5.1 Dataset

We use a video with about 10,000 frames recorded at 30 frames per second of a *Temnothorax rugatulus* ant colony. In this video, ants are painted individually in order to identify them more easily. An

automated multi-target system has been used to analyze the ant paths. A ground truth trajectory has been created by manually examining and correcting the automated trajectory. We use this ground truth to evaluate our algorithm.
In this data set there are 252 users generating trajectories for 50 ants. Each ant has 2-8 trajectories. We construct the intersection graph G and apply the global greedy algorithm to build the representative trajectories.

5.2 Random Graph Generator

Since intersections are the primary source of error when tracking ants, we randomly generate ant paths but force one intersection at every time step. We do this in order to test the global greedy algorithm on data with many intersections.
The forced-intersection algorithm is written in Matlab. It takes in the number of ants, the number of time frames, the dimensions of the space and the maximum displacement. The output is a text file with the x-coordinate and the y-coordinate for at each time frame for each ant. The path is generated using a 3D random walk. Intersections are forced by selecting an ant at random at each time step, then finding the nearest neighboring ant and determining an intermediate point at which they will meet. We set the x and y coordinates of the two ants accordingly then generate the coordinates for the remaining ants. In this algorithm, ants are equally likely to move up, down, left, right or diagonally but are more likely to remain in their previous position. We generate 3 data sets using this algorithm each with 7 ants, 50 time frames, with a maximum displacement of 50 units.

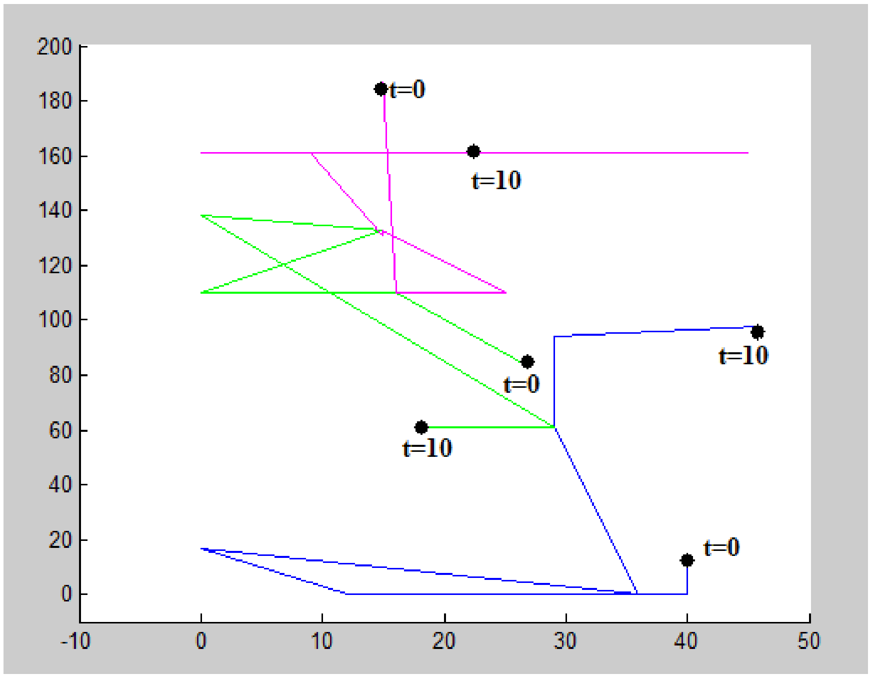


Figure 6: Three randomly generated ant paths for 10 time steps

5.3 Generating User Data

Once the underlying graph is generated, we simulate user data. This algorithm takes in the text files with the generated ant paths and outputs text files with simulated trajectories of 100 users. Each output path represents a user tracking an ant in our base map. To generate this data we assume users always start at the correct point and users will only switch from one path to another when there is an intersection. When an intersection occurs, the user will switch paths with some fixed probability, the default is 0.25; a mistake will occur with probability 0.25 at each intersection point. We generate user data by randomly selecting x and y coordinates within a specified range of the true base map data, the default range is 20 units. We test error probabilities 0.01, 0.05, 0.10, 0.25. See Fig. 7.

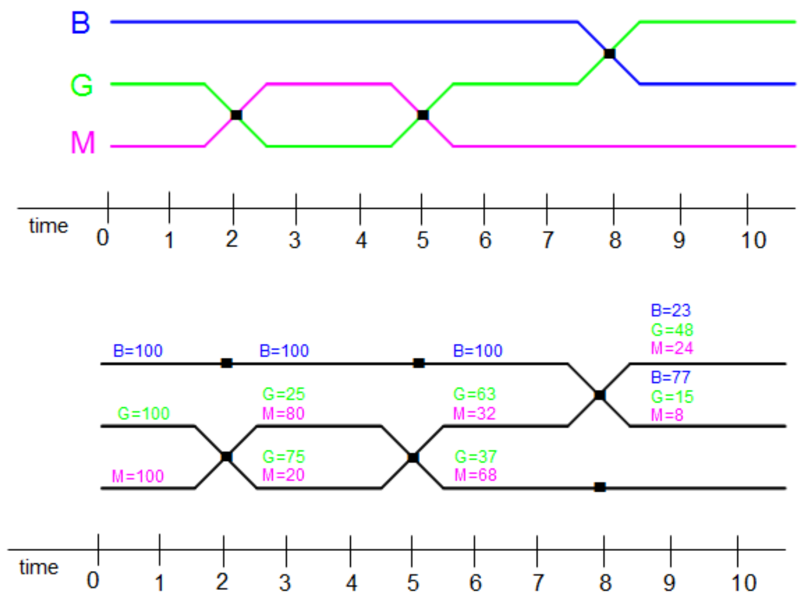


Figure 7: Trajectory graph for generated ant paths and simulated user data

6 Greedy Algorithm

We describe a heuristic for extracting representative trajectories from a collection of trajectories using a global greedy algorithm. We pick an ant at random and begin at its starting vertex in the global graph G. Each edge is weighted by the number of users who tracked that ant across that particular edge. We add an edge to the representative path if it has the maximum weight for that ant. We continue extending the representative path until the last time step is reached. We then remove these edges from G and repeat the process. This continues until all k representative trajectories have been determined. Since ants are chosen at random, we repeat this process and chose the set of representative trajectories with the highest weight over all trajectories.

7 Experiments

We analyze the performance of the global greedy algorithm on two sets of data. The first is using the citizen scientist method on a video of the *Temnothorax rugatulus* ant colony. We collect data from 252 users. We compare to the automated solution. The second is simulated trajectories using the forced-intersection algorithm. We generate 3 data sets using this algorithm each with 7 ants, 50 time frames, with a maximum displacement of 50 units. We vary the probability of error, testing 0.01, 0.05, 0.10 and 0.25. We also vary the clustering threshold, testing $R = 10, 20, 40$. We calculate the variance of the representative trajectories from the true ant paths. We also compare the variance of the global greedy algorithm to the Homotopy Median algorithm. Finally we compare the variance of the homotopy median algorithm to the maximum, minimum and average variance for the global greedy algorithm

8 Results and Discussion

We compute the representative trajectories for each ant using the global greedy algorithm then compare to either the automated trajectory (for the collected data) or the true trajectory (for the generated data). We measure the average variance of the distance between pairs of points of the computed and true trajectories.

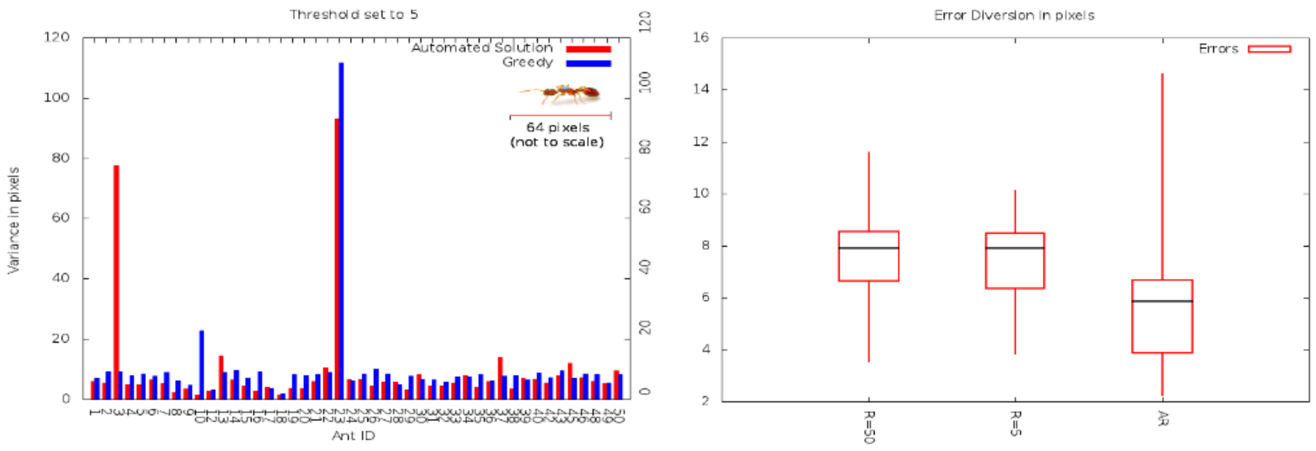


Figure 8: Average variance in pixels of global greedy vs. automated solution with threshold set to 5 pixels. Average error for R=5, R=50 and automated solution

We can see that for the collected data the global greedy algorithm performs similarly to the automated solution. The average variance of both algorithms generally remains below 20 pixels for both thresholds $R = 5$ and $R = 50$. The box and whisker plot in Figure 10 demonstrates that although the average error for the automated solution is slightly less, about 6 pixels, the variance is quite large, about 13 pixels. For the global greedy algorithm, the average error is about 8 but the variance is only 8 for $R = 50$ and 6 for $R = 5$.

For the generated data we plot the average variance for $R = 10, 20, 40$. Again this is the average variance of each point from the true path. We notice that the variance for all three data sets is between 20-30 pixels. We can see that in general the lower threshold value has a smaller variance but that this is not always the case. For ant 1 in data set 2, the lower threshold (in blue) has a significantly higher variance than the larger threshold values. While for ant 5 in the same data set the lower threshold has a much smaller variance than the other two. Although we can not determine from these experiments the ideal threshold value it is clear that the threshold plays an important role in determining the variance of the representative trajectory.

We then examine the effect of the probability of user error on the average variance of the representative path. See Fig. 11. Here we vary the probability that a user mistakenly switches paths, we test 1%, 5%, 10% and 25% probability of error. For each error probability we also vary the clustering threshold from $R = 10$ to $R = 20$ to $R = 40$.

Note that at 1% probability of error the clustering threshold has little effect of the average variance of the representative path. The variance for all thresholds remains between 15-20 pixels. When the error is increased to 5% we notice that the larger clustering threshold has a lower variance and the variance is now between 15-33 pixels. In the case of 10% user error, the 20 pixel clustering threshold has the largest variance for 6 of the 7 ant paths and the variance is between 2-35 pixels. Finally in the case of 25% user error, the 40 pixel threshold has the largest variance for all 7 ants and the variance is between 10-35 pixels. The relationship between probability of user error and clustering threshold is interesting to note. There is not one single threshold value that is lowest across all probabilities.
Examining the data in a slightly different way, Fig 12, we can see that each clustering threshold minimizes the variance for a different probability of error. For the threshold value of 10, we see that

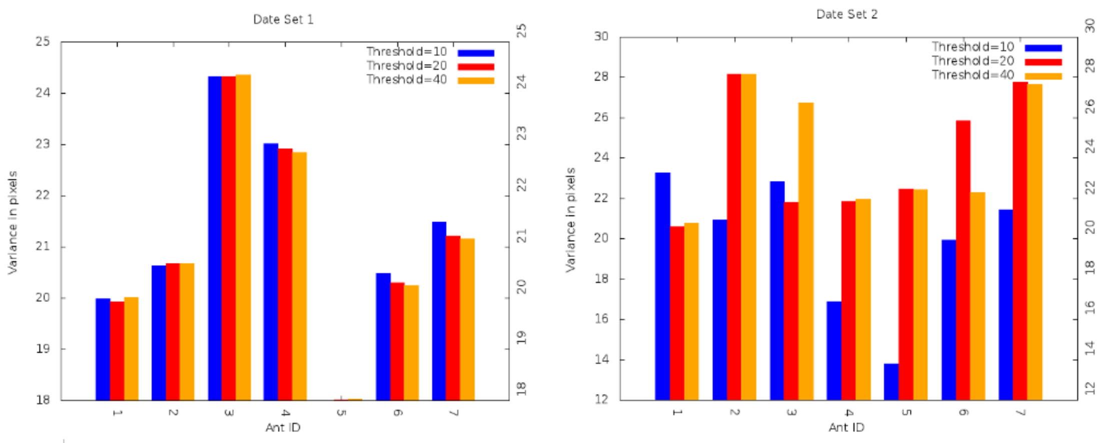


Figure 9: Average variance for $R = 10, 20, 40$ pixels for Data Set 1 and Data Set 2

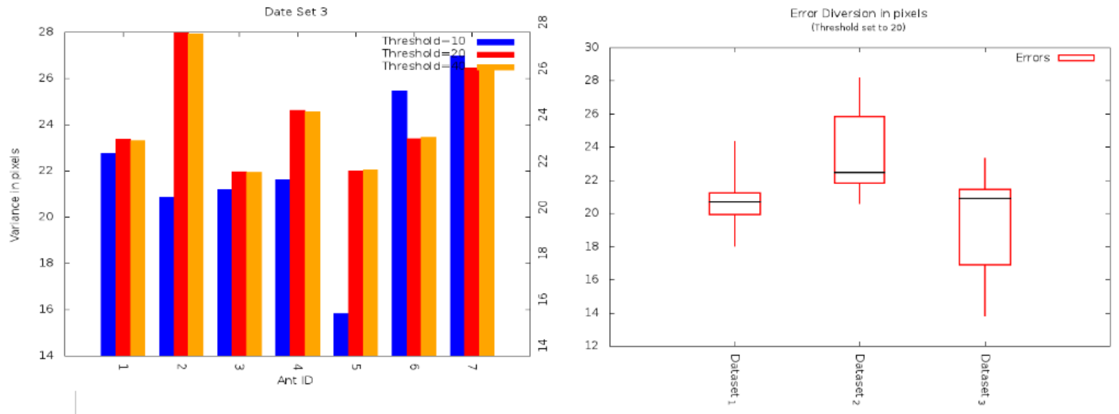


Figure 10: Average variance for $R = 10, 20, 40$ pixels for Data Set 3 and average error plot

the variance is low when the probability of error is 1%, 10%, and 25%. But is much higher when the probability of error is 5%. When the threshold is 20, the variance is much larger when the probability of error is 5% and 10%. When the clustering threshold is 40, the variance is larger when the probability of error is 10% and 25%. We can conclude that if there is a low probability of error about 1% then any threshold will produce a representative path with low variance. If the probability of error is about 5% then a high threshold, about 40, will yield a path with low variance. If the probability of error is about 10% then a low threshold, about 10, results in a path with low variance and if the probability of error is 25% then a threshold value between 10-20 will yield a path with low variance.

We can further observe this relationship between probability of user error and clustering threshold using the box and whisker plot in Fig. 13. At 5% user error the average error for all data sets across all threshold values is between 20-25 pixels. All errors lie between 15-35 pixels. At 10% user error, however, the average error for Data Sets 2 and 3 fluctuate depending on the clustering threshold. The minimum error is 2 pixels, while the maximum error is 35 pixels. It is interesting to note that Data Set 1 seems to be invariant with respect to both the clustering threshold and the probability of user error. In fact, even when the user error is 25% the box and whisker plot remains unchanged. It is still unclear what caused this result.

We compare the variance of the global greedy algorithm to the local homotopy median algorithm. See Fig. 14. We use Data Set 1 and compare the average variance of both algorithms. We can see the variance for the local homotopy algorithm is between 40 and 90 pixels while the global greedy algorithm is around 20 pixels.

Since the global greedy algorithm is a randomized algorithm we compare the maximum, minimum

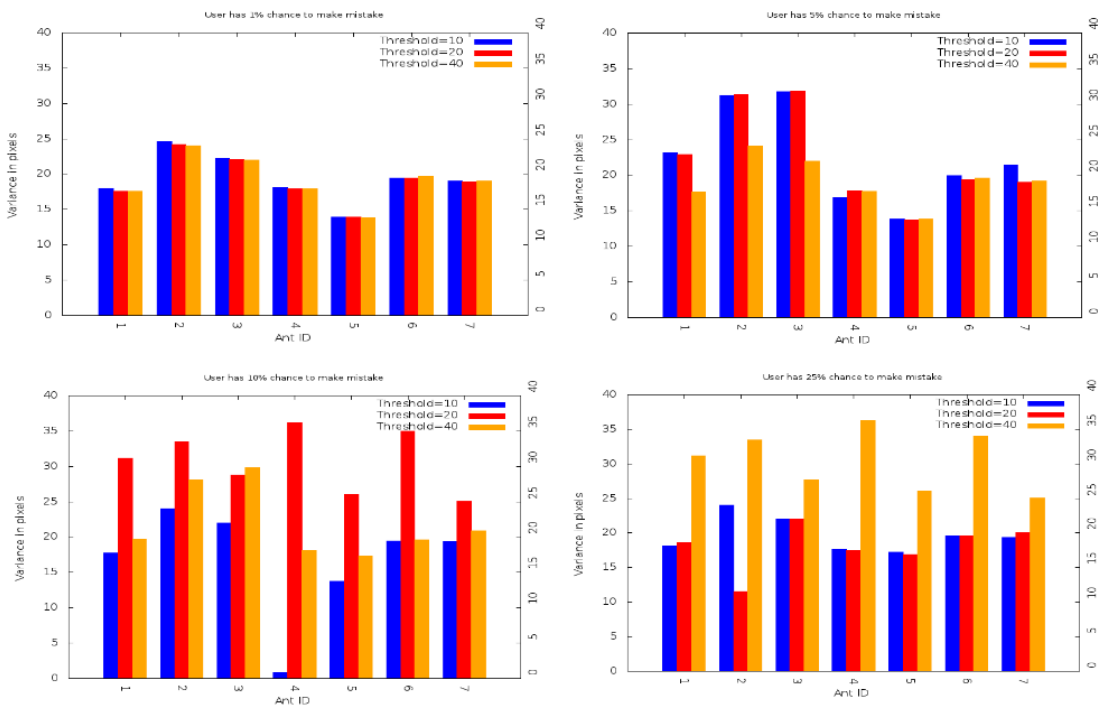


Figure 11: Data Set 2

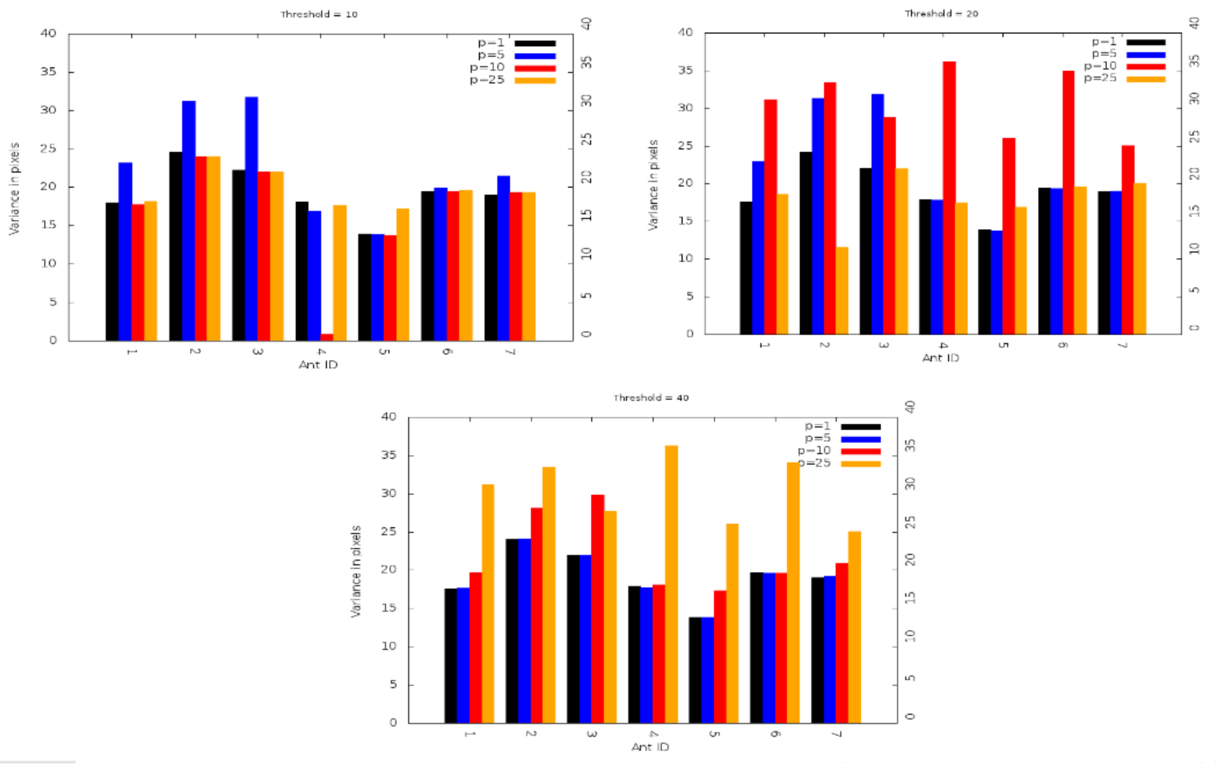


Figure 12: Data Set 2

and average variance of the global algorithm to the variance of the homotopy median algorithm. See Fig. 14. We see that in the best and average cases, the global greedy algorithm has a lower variance than the local homotopy algorithm. In the worst case, however, it is no better than the local algorithm. More plots can be found in the appendix.

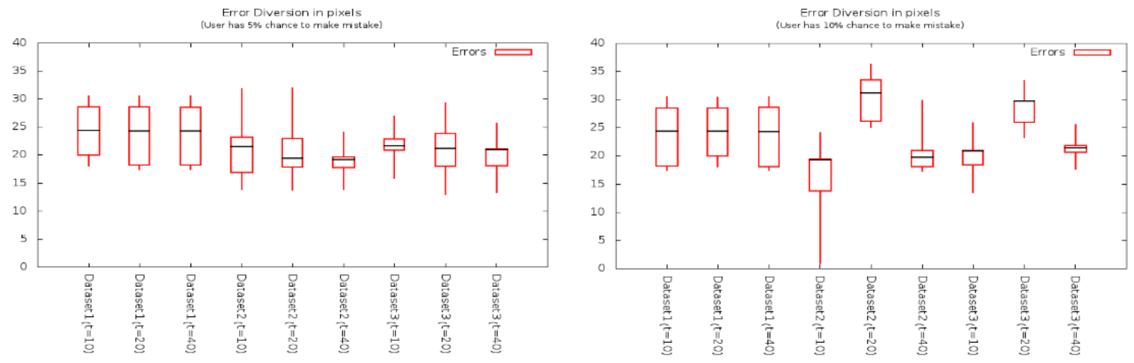


Figure 13: Average error for probability of user error 5% and 10%

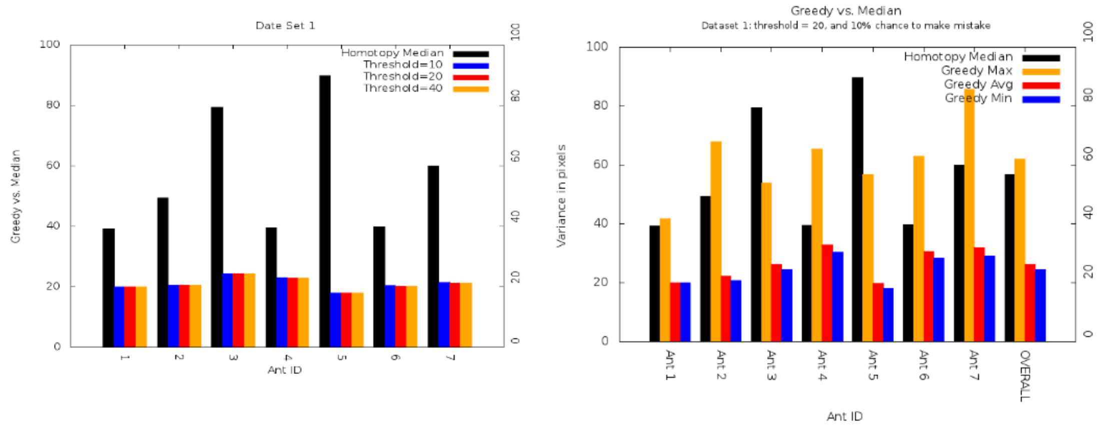


Figure 14: Global greedy vs. local homotopy median

9 Conclusions and Future Work

We explored a global greedy algorithm for extracting representative trajectories from a large number of input trajectories. When analyzing user data collected by 'citizen scientists' we found that the global greedy algorithm was comparable to the automated solution. We demonstrated that the global greedy algorithm is sensitive to changes in the user probability of error and the clustering threshold. We found that the global greedy algorithm out-performs the local homotopy median algorithm in both the best and the average cases, but not in the worst case.

It would be interesting to continue exploring the effect of user error and the clustering threshold on representative trajectories, especially considering the results of Data Set 1. It would also be interesting to determine the average probability of error from user data. We test a range of probabilities here but have no analysis to indicate what the observed value actually is.

We also hope to perform a similar analysis on other global algorithms to see the sensitivity of other algorithms to the probability of error and the clustering threshold.

10 Appendix

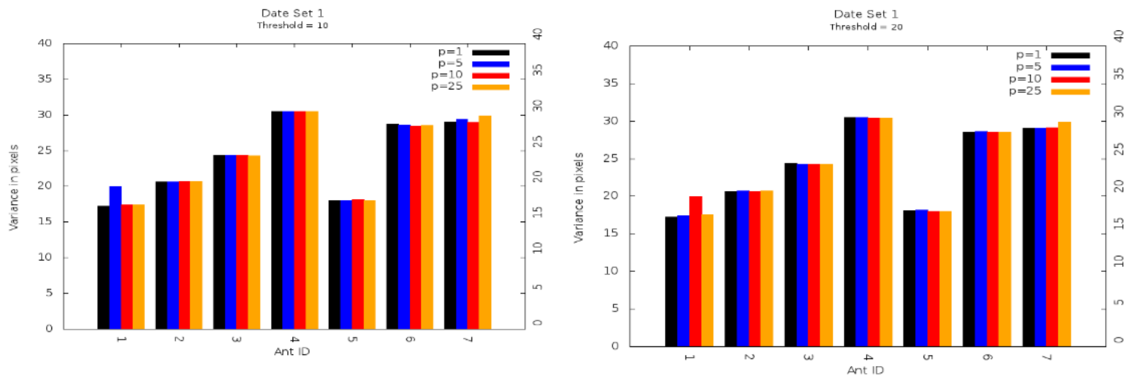


Figure 15: Data Set 1 with threshold $R = 10, 20$

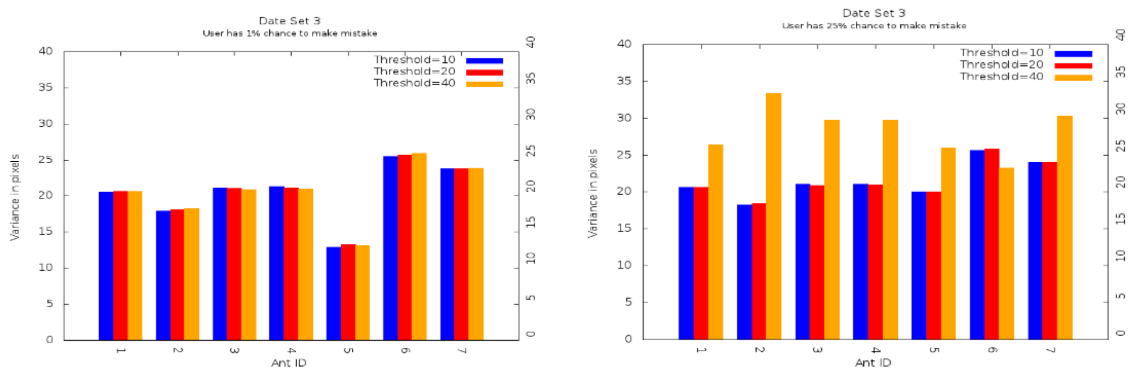


Figure 16: Data Set 3 with probability of error 1% and 25%

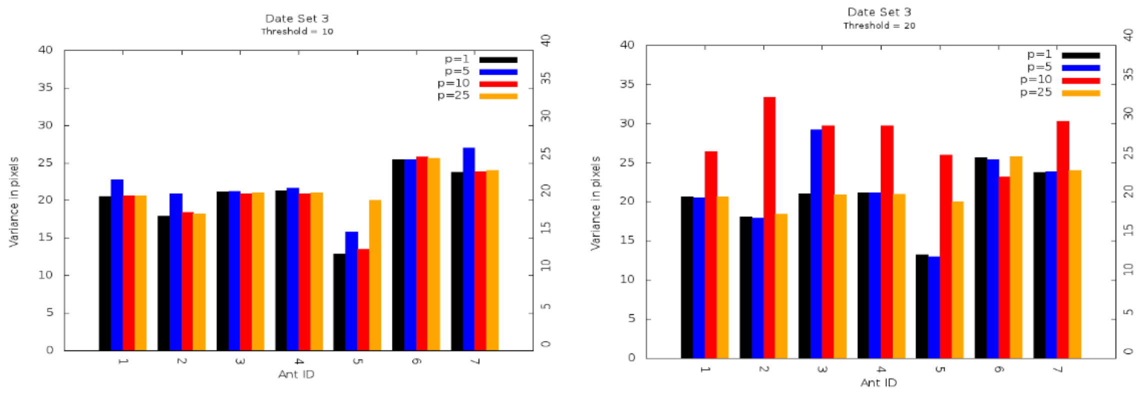
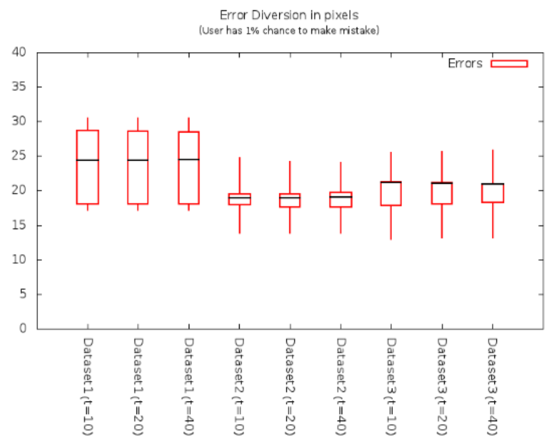


Figure 17: Data Set 3 with threshold $R = 10, 20$



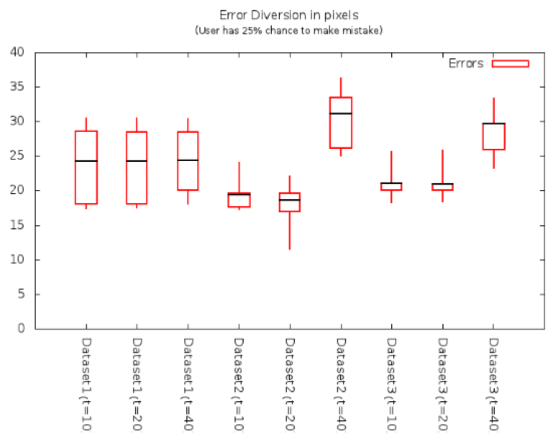
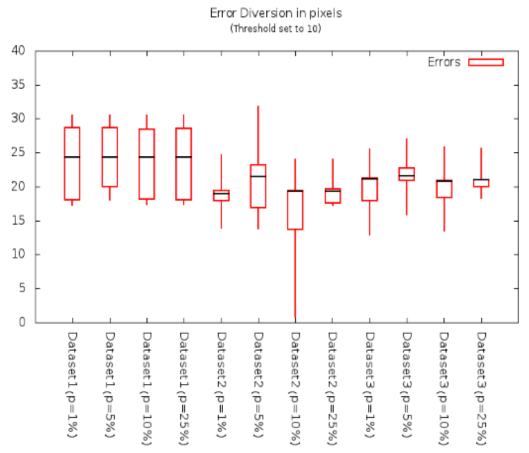


Figure 18: Error for 1% probability and 25% probability



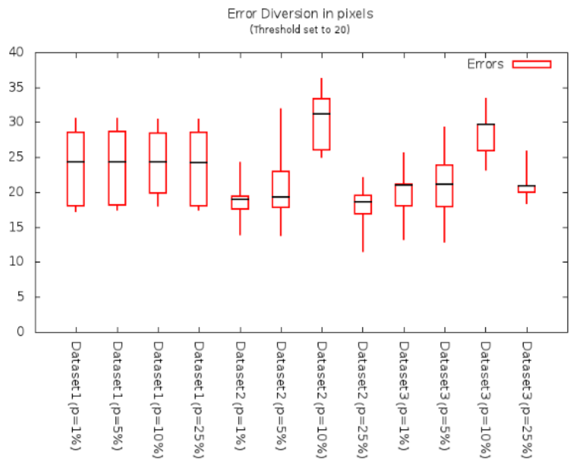


Figure 19: Error for threshold $R = 10, 20$

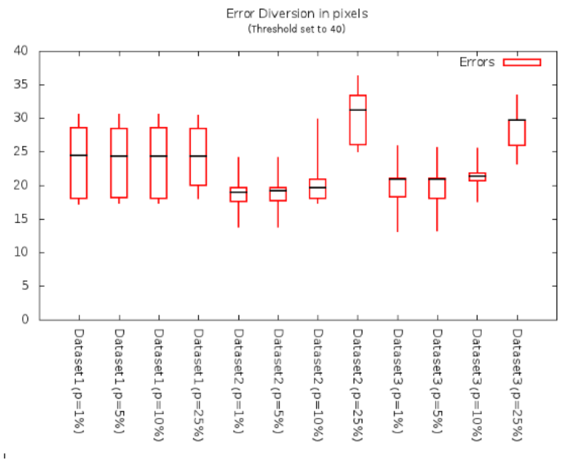


Figure 20: Error for threshold $R = 40$

References

[1] Helmut Alt, Alon Efrat, Gunter Rote, and Carola Wenk. Matching planar maps, 2003.

[2] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[3] Kevin Buchin, Maike Buchin, Marc Van Kreveld, Maarten Löffler, Rodrigo I. Silveira, Carola Wenk, and Lionov Wiratma. Median trajectories. In *Proceedings of the 18th annual European conference on Algorithms: Part I*, ESA'10, pages 463–474, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] Kevin Buchin, Maike Buchin, and Yusu Wang. Exact algorithm for partial curve matching via the frchet distance. In *Proc. 20th ACM-SIAM Symposium on Discrete Algorithms*, 2009.

[5] Joy Chen, Ryan Compton, Aprana Das, Yi Huang, Stephen Kobourov, Paul Shen, Sankar Veeramoni, and Yunhao Yu. Angry ants: A citizen science approach to computing accurate averge trajectories. *arXiv: 1212.0935vl [cs.CG]*, 2012.

[6] Livio De La Cruz, Stephen Kobourov, Sergey Pupyrev, Paul Shen, and Sankar Veeramoni. Angry ants: An approach for the accurate average trajectories using citizen science. In *Submitted to European Symposium on Algorithms ESA 2013*, 2013.

[7] Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the fréchet distance for realistic curves in near linear time. *CoRR*, abs/1003.0460, 2010.

[8] M. Fletcher, A. Dornhaus, and M.C. Shin. Multiple ant tracking with global foreground maximization and variable target proposal distribution. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 570 –576, jan. 2011.

[9] P. Maitra, S. Schneider, and M.C. Shin. Robust bee tracking with adaptive appearance template and geometry-constrained resampling. In *Applications of Computer Vision (WACV), 2009 Workshop on*, pages 1 –6, dec. 2009.

[10] Scott Morris and Kobus Barnard. Finding trails.

[11] C. Poff, H. Nguyen, T. Kang, and M.C. Shin. Efficient tracking of ants in long video with gpu and interaction. In *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*, pages 57 –62, jan. 2012.

[12] Robin Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973.

[13] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), December 2006.

[14] I. Zuriarrain, F. Lerasle, N. Arana, and M. Devy. An mcmc-based particle filter for multiple person tracking. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1 –4, dec. 2008.