Computing Consensus Curves

L. De La Cruz[1], S. Kobourov[1]*, S. Pupyrev[1,2]⋆, P. Shen[1], and S. Veeramoni[1]⋆

[1] Department of Computer Science, University of Arizona, Tucson, AZ, USA
[2] Institute of Mathematics and Computer Science, Ural Federal University, Ekaterinburg, Russia

Abstract. We study the problem of extracting accurate average ant trajectories from many (inaccurate) input trajectories contributed by citizen scientists. Although there are many generic software tools for motion tracking and specific ones for insect tracking, even untrained humans are better at this task. We consider several local (one ant at a time) and global (all ants together) methods. Our best performing algorithm uses a novel global method, based on finding edge-disjoint paths in a graph constructed from the input trajectories. The underlying optimization problem is a new and interesting network flow variant. Even though the problem is NP-complete, two heuristics work well in practice, outperforming all other approaches, including the best automated system.

1 Introduction

Tracking moving objects in video is a difficult task to automate. Despite advances in machine learning and computer vision, the best way to accomplish such a task still is by hand. At the same time, people spend millions of hours each day playing games like *Solitaire*, *Angry Birds*, and *Farmville* on phones and computers. This presents an opportunity to harness some of the time people spend on online games for more productive but still enjoyable work. In the last few years it has been shown that citizen scientists can contribute to image processing tasks, such as the *Galaxy Zoo* project in which thousands of citizen scientists labeled millions of images of galaxies from the Hubble Deep Sky Survey and *FoldIt* in which online gamers helped to decode the structure of an AIDS protein — a problem which stumped researchers for 15 years.

AngryAnts is our online game available at http://angryants.cs.arizona.edu, which plays videos and allows citizen scientists to build the trajectory of a specified ant via mouse clicks. When we have enough data, we compute a most realistic *consensus* trajectory for each ant. Our motivation comes from biologists who wish to discover longitudinal behavioral patterns in ant colonies. The trajectories of individual ants in a colony extracted from videos are needed to answer questions such as how often do ants communicate, what different roles do ants play in a colony, and how does interaction and communication affect the success or failure of a colony? Existing automated solutions are not good enough, and there is only so much data that even motivated students can annotate in the research lab.

Related Work. The problem of computing the most likely trajectory from a set of given trajectories has been studied in many different contexts and here we mention a few
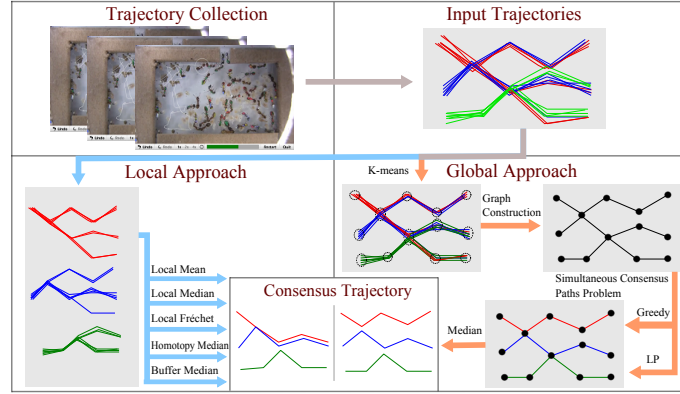
Fig. 1: Overview of consensus trajectory computation via citizen science. Local consensus may route different ants along the same trajectory (red and blue), while the global consensus ensures disjoint trajectories.

examples, which are similar to our approach. Buchin *et al.* [4] look for a representative trajectory for a given set of trajectories and compute a median representative rather than the mean. The Fréchet distance as similarity measure for trajectories is studied by Buchin *et al.* [3], who show how to incorporate time-correspondence and directional constraints. Trajcevski *et al.* [15] use the maximum distance at corresponding times as a measure of similarity between pairs of trajectories, and describe algorithms for matching under rotations and translations.

Yilmaz *et al.* [17] survey the state-of-the-art in object tracking methods. Some of the most recent methods include general approaches for tracking cells undergoing collisions by Nguyen *et al.* [13] and specific approaches for tracking insects by Fletcher *et al.* [7]. Also related are the automatic tracking method for tracking bees by Veeraraghavan *et al.* [16] and cluster-based, data-association approaches for tracking bats in infrared video by Betke *et al.* [2]. Tracking the motion and interaction of ants has also been studied by Khan *et al.* [8], who describe probabilistic methods, and by Maitra *et al.* [11] using computer vision techniques.

Our Contributions. We describe a citizen science approach for extracting consensus trajectories in an online game setting; see Fig. 1 for an overview. Combining human-generated trajectories with a new global approach for computing consensus curves outperforms even the most sophisticated and computationally expensive tracking algorithms [14], even in the most advantageous setting for automated solutions (e.g., high resolution video, sparse ant colony, individually painted ants).

Consider a *trajectory* as a sequence of T pairs $(p_i, t_i), 1 \leq i \leq T$, where $p_i = (x_i, y_i)$ is a point in the plane representing the position of an ant at timestamp t_i. We assume that between timestamps an ant keeps a constant velocity, and therefore, its trajectory is a polyline in 3D (or a possibly self-intersecting polyline in 2D). The input of our problem is a collection $\tau_1, \ldots, \tau_m$ of trajectories. Each trajectory corresponds to one of k ants, and we assume that there exists at least one trajectory for each ant, that is, $m \geq k$. Since the input data comes from the *AngryAnts* game, all trajectories

have the same length (number of points). We also assume that the initial position of each ant is provided by the game (from a different game level called "Count the Ants" where users click on all the ants in the first frame of the video to identify their starting positions). Therefore, the first points of trajectories corresponding to the same ant are identical. Our goal is to compute a *consensus trajectory*, that is, our best guess for the actual route taken, for each of the k ants. While intuitively we are looking for the most probable ant trajectories, it is far from obvious how to measure the quality of a solution.

We designed, implemented, and evaluated several methods for computing accurate consensus trajectories from many (possibly inaccurate) trajectories submitted by citizen scientists. We consider several local strategies (clustering, median trajectories, and Fréchet matching) in one-ant-at-a-time setting. We also designed and implemented a novel global method in the all-ants-together setting. This approach is based on finding edge-disjoint paths in a graph constructed from all input trajectories. The underlying optimization problem is a new and interesting variant of network flow. Even though the problem is NP-complete, our two heuristics work well in practice, outperforming all other approaches including the automated system.

2   The Local Approach

Local Mean. Intuitively, a mean trajectory averages the locations for input trajectories. To compute the mean, we identify all input trajectories $\tau_1, \ldots, \tau_{m_c}$ for a particular ant c. For each timestamp t_i, we query points $p_1 = (x_1, y_1), \ldots, p_{m_c} = (x_{m_c}, y_{m_c})$ corresponding to t_i. The average point is (x_{avg}, y_{avg}), where $x_{avg} = (x_1 + \cdots + x_{m_c})/m_c$ and $y_{avg} = (y_1 + \cdots + y_{m_c})/m_c$ give the location of the mean trajectory at timestamp t_i. The sequence of these average points over time defines the local mean consensus trajectory, which is good when the number of input trajectories is very large or when all input trajectories are very accurate. In reality, however, this is often not the case; a single inaccurate input trajectory may greatly influence the result.

Local Median. The median point is more robust to outliers than the mean. For a set of points $p_1, p_2, \ldots, p_{m_c}$, we choose as median the point (x_{med}, y_{med}), where x_{med} is the median of the array $x_1, \ldots, x_{m_c}$ and y_{med} is the median of the array $y_1, \ldots, y_{m_c}$. The sequence of these median points over time defines the local median consensus trajectory. Note that the median of a set of points is not necessarily a point of the set: it could place an ant at a point that matches none of the input trajectories.

Local Fréchet. Informally, the Fréchet distance between two trajectories can be illustrated as the minimum dog-leash distance that allows a man to walk along one trajectory and his dog along the other, while connected at all times by the leash [1]. Computing the Fréchet distance produces an alignment of the trajectories: at each step, the position of the man is mapped to the position of the dog. Given the Fréchet alignment of two trajectories, we compute their consensus by taking the midpoint of the leash over time. To find the consensus, we repeatedly compute pairwise consensuses until only one trajectory remains. Since the results of the algorithm depends on the order in which the trajectories are merged, we try several different random orders choosing the best result. Note that, by definition, the Fréchet alignment of trajectories uses only the order of points and ignores the timestamps that are an essential feature of our input.

Homotopy Median and Buffer Median. In the above methods, the average of two locations could be an invalid location. In the median trajectory approach, the computed trajectory always lies on segments of input curves. Note that this is more restrictive than the point-based median method described above. We use two algorithms suggested by Buchin *et al.* [4] and Wiratma *et al.* [9]. In the *Homotopy median* algorithm, obstacles are placed in large faces bounded by segments of input trajectories to ensure that the median trajectory is homotopic to the set of trajectories that go around the obstacles. Hence, the median does not miss segments if the input trajectories are self-intersecting. The *Buffer median* (also known as Majority Median) algorithm is a combination of the buffer concept and Dijkstra's shortest path algorithm. A buffer is defined around a segment so that if the segment is a part of the median trajectory, then its buffer intersects all input trajectories. Thus, segments located near the middle of the trajectory have smaller buffer size and are good candidates for the median trajectory. Note again that both the Homotopy median and the Buffer median approaches ignore the timestamps.

3   The Global Approach

In the global approach we consider all input trajectories for all k ants together. The main motivation is that a trajectory corresponding to an ant may contain valid pieces of trajectories for other ants: a citizen scientist may mistakenly switch from tracking an ant x to tracking a different ant y at an intersection point where x and y cross paths. However, even when such mistakes occur, the trace after the intersection point is still useful as it contains a part of the trajectory of ant y. The global approach allows us to retain this possibly useful data as shown by an example in Fig. 1. Given a set of input trajectories, we compute the consensus trajectories in three steps: (1) create a graph G; (2) compute edge-disjoint paths in G; (3) extract consensus trajectories from the paths.

Step 1. We begin by creating a graph that models the interactions between ants in the video. For every timestamp, the graph has at most k vertices, which correspond to the positions of the k ants. If several ants are located close to each other, then we consider them to be at the same vertex. Intuitively, each such vertex is a possible point for a citizen scientist to switch to a wrong ant.

A weighted directed graph $G = (V, E)$ is constructed as follows. For every timestamp t_i, we extract points $p_1, \ldots, p_m$ from the given trajectories, where p_j is the position of trajectory τ_j at t_i. Using a modification of the k-means clustering algorithm [10], we partition the points into $\leq k$ clusters. Our clustering algorithm differs from the classical k-means in that it always merges the points into one cluster located closer than 50 pixels from each other. The vertices of G are the clusters for all timestamps; thus, G has at most kT vertices. We then add edges between vertices in consecutive timestamps. Let V_i represent a set of vertices at timestamp t_i. We add an edge (u, v) between two vertices $u \in V_i$ and $v \in V_{i+1}$ if there is an input trajectory with point p_i belonging to cluster u and point p_{i+1} belonging to cluster v. For each edge, we create k non-negative weights. For each ant $1 \leq x \leq k$ and for each edge $(u, v) \in E, u \in V_i, v \in V_{i+1}$, there is weight $w_{uv}^x \in \mathbb{Z}_{\geq 0}$, which equals to the number of trajectories between timestamps i and $i + 1$ associated with the ant x passing through the clusters u and v.

Note that by construction graph G is acyclic. Let $d^{in}(v)$ and $d^{out}(v)$ denote the indegree and the outdegree of the vertex v. Clearly, the only vertices with $d^{in}(v) = 0$ are in V_1, and the only vertices with $d^{out}(v) = 0$ are in V_T; we call them *source* and *destination* vertices and denote them by s_i and t_i, respectively. We assume that for all *intermediate* vertices $v \in V_i, 1 < i < T$, we have $d^{in}(v) = d^{out}(v)$; this is a realistic assumption because for each intersection point of trajectories, the number of incoming ants equals the number of outgoing ants. We say that a directed graph satisfies the *ant-conservation condition* if (1) the number of outgoing edges from sources and the number of incoming edges to destinations is equal to k, that is, $\sum_i d^{out}(s_i) = \sum_i d^{in}(t_i) = k$, and (2) indegree and outdegree of all its intermediate vertices are the same, that is, $d^{in}(v) = d^{out}(v)$.

Step 2. We compute k edge-disjoint paths in G, corresponding to the most "realistic" trajectories of the ants. The paths connect the k distinct vertices in V_1 to the k distinct vertices in V_T. Since the initial position of each ant is a part of input, we know the starting vertex of each path. However, it is not obvious which of the destination vertices in V_T correspond to each ant. To measure the quality of the resulting ant trajectories, that is, how well the edge-disjoint paths match the input trajectories, we introduce the following optimization problem.

SIMULTANEOUS CONSENSUS PATHS (SCP)

Input: A directed acyclic graph $G = (V, E)$ with k sources $s_1, \ldots, s_k$ and k destinations $t_1, \ldots, t_k$ satisfying the ant-conservation condition. The weight of an edge $e \in E$ for $1 \leq i \leq k$ equals $w_e^i \in \mathbb{Z}_{\geq 0}$.

Task: Find k edge-disjoint paths $P_1, \ldots, P_k$ so that path P_i starts at s_i and ends at t_j for some $1 \leq j \leq k$, and the total $cost = \sum_{i=1}^k \sum_{e \in P_i} w_e^i$ is maximized. Note that the objective is to simultaneously optimize all k disjoint paths. The decision problem is to find k edge-disjoint paths with total $cost \geq c$ for some constant $c \geq 0$.

The problem is related to the integer multi-commodity flow problem, which is known to be NP-hard [6]. In our setting, the weights on edges for different "commodities" are different, and the source-destination pairs are not known in advance. We study the SCP problem in the next section.

Step 3. We construct consensus trajectories corresponding to a solution of the SCP problem. Let P be a path for an ant x computed in the previous step. For each timestamp t_i, we consider the edge $(u, v) \in P, u \in V_i, v \in V_{i+1}$, and find a set S_{uv} of all input trajectories passing through both clusters u and v. We emphasize here that S_{uv} may contain (and often does contain) trajectories corresponding to more ants than just ant x. Next we compute the median of points of S_{uv} at timestamp t_i; the median is used as the position of ant x at timestamp t_i. The resulting trajectory of x is a polyline connecting its positions for all $t_i, 1 \leq i \leq T$.

4   The SIMULTANEOUS CONSENSUS PATHS Problem

The SCP problem is NP-complete even when restricted to planar graphs by a reduction from a variant of the edge-disjoint path problem [12]. On the other hand, the problem is fixed-parameter tractable in the number of paths. For a fixed constant k, it can be solved optimally via dynamic programming in time $O(|E| + k!|V|)$. The proofs of both claims
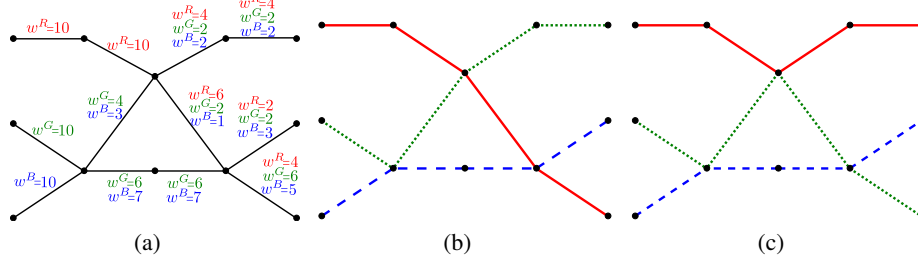
Fig. 2: The greedy algorithm may produce non-optimal solution. (a) The input graph with 3 paths: R, G, and B. (b) The paths computed by the greedy algorithm with $cost = 75$ (R is shown solid, G – dotted, B – dashed). (c) The optimal solution with $cost = 77$.

are in the full version [5]. Next we suggest a greedy heuristic and provide an integer linear programming (ILP) formulation for SCP.

The Greedy Algorithm. The algorithm finds the longest path among all source-destination pairs s_i, t_j. The length of a path starting at s_i is the sum of w_{uv}^i for all edges (u, v) of the path. Since G is an acyclic directed graph, the longest path for the specified pair s_i, t_j can be computed in time $O(|E| + |V|)$ via dynamic programming. Once the longest path is found, we remove all edges of the path from G and proceed with the next longest path. The algorithm finds at most k paths on each iteration and the number of iterations is k; hence, the overall running time is $O(k^2(|V| + |E|))$.

The algorithm always yields a solution with k disjoint paths. Initially, G satisfies the ant-conservation condition: $d^{in}(v) = d^{out}(v)$ for all intermediate vertices v. Since G is connected, there exists a source-destination path. After removing the longest path, G may be disconnected, but the ant-conservation condition still holds for every connected component. The number of outgoing edges from sources and the number of incoming edges to destinations are equal for every connected component. Thus, the greedy algorithm produces a feasible solution but not necessarily the optimal one; see Fig. 2.

Linear Programming Formulation. Let P_i denote the path in G from s_i to t_j for some j (the path of the i-th ant). For each P_i and each edge e, we introduce a binary variable x_e^i, which indicates whether path P_i passes through the edge e. The SCP problem can be formulated as the following ILP:

$$
\begin{aligned}
\textbf{maximize}\quad & \sum_e \sum_i w_e^i x_e^i \\
\textbf{subject to}\quad & \sum_i x_e^i = 1 & \forall e \in E & \quad (1) \\
& \sum_{uv} x_{uv}^i = \sum_{vw} x_{vw}^i & \forall v \in V \setminus \{s_1, t_1, \ldots, s_k, t_k\}, 1 \le i \le k & \quad (2) \\
& \sum_v x_{s_i v}^i = 1 & \forall 1 \le i \le k & \quad (3) \\
& x_e^i \in \{0, 1\} & \forall e \in E, 1 \le i \le k & \quad (4)
\end{aligned}
$$

Here constraint (1) guarantees that the paths are disjoint: there is exactly one path passing through every edge. Constraint (2) enforces consistency of the paths at every intermediate vertex: if a vertex v is contained in a path, then the path passes through an edge (u, v) and an edge (v, w) for some $u, w \in V$. In constraint (3) we sum over
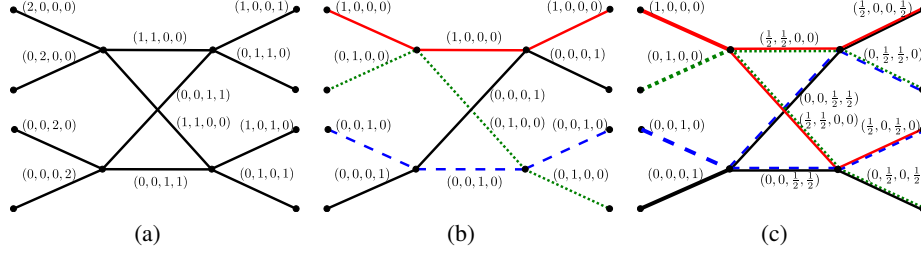
Fig. 3: Graph in which a fractional solution has cost greater than the cost of any integer solution. (a) The input graph with 4 paths. The vector on the edge e corresponds to the weights $(w_e^1, w_e^2, w_e^3, w_e^4)$ for the paths on e. (b) The optimum integer solution with $cost = 15$. The vector on the edge e corresponds to the solution $(x_e^1, x_e^2, x_e^3, x_e^4)$ on e. (c) A fractional solution with $cost = 16$. The vector on the edge e corresponds to the solution $(x_e^1, x_e^2, x_e^3, x_e^4)$ on e.

vertices v with $(s_i, v) \in E$; it implies that the i-th path starts at source s_i. If we relax the integrality constraint (4) by $0 \leq x_e^i \leq 1$, we have a fractional LP formulation for the SCP problem, which can be solved in polynomial-time. However, the solution does not have a natural interpretation in the context of ants (fractional ants do not make sense in the biological problem). Further, we found an example for which the best fractional solution has cost strictly greater than the cost of any integer solution; see Fig. 3.

We can convert an optimal fractional LP solution x^* into a feasible integer solution as follows. Randomly pick an ant $1 \leq i \leq k$, with probability of choosing the i-th ant proportional to its weight $\sum_e w_e^i x^{*i}_e$ in the fractional solution. We then consider the graph with modified edge weights in which the weight of an edge $e \in E$ is $w_e^i x^{*i}_e$. We look for the longest path starting at source s_i in this graph, assign the path to the i-th ant, and remove this path from the graph. We then rerun the LP to find a fractional solution on the smaller instance. Our experiments suggest that this rounding scheme yields an integer solution that is close to optimal.

5   Experimental Results

We use a machine with Intel i5 3.2GHz, 8GB RAM and CPLEX Optimizer.

Real-World Dataset. Here we consider a real-world scenario and compare ground truth data to seven different consensus algorithms described in this paper, along with an automated solution. To evaluate our various algorithms, we work with a video of a *Temnothorax rugatulus* ant colony containing $10,000$ frames, recorded at 30 frames per second. This particular video contains ants that are individually painted and has been analyzed with the state-of-the-art automated multi-target tracking system of Poff *et al.* [14]. The method required about 160 minutes to analyze the video. To evaluate the automated system, they create a *ground truth* trajectory for each ant, by manually examining *every ant in every 100th frame* of the automated output and correcting when necessary. We use this ground truth data to evaluate the efficiency of the algorithms considered and the automated system. Note that just by the way the ground truth is generated, results are inherently biased in favor of the automated solution.

Table 1: Average and worst root-mean-square error (in pixels) computed for proposed algorithms.

Algorithm	Worst RMSE	Average RMSE	Runtime
Automated Solution	95.308	9.593	160 min
Local Mean	105.271	12.531	< 100 ms
Local Median	112.741	9.801	< 100 ms
Local Fréchet	127.104	15.562	1.2 sec
Homotopy Median	146.267	20.244	8.2 sec
Buffer Median	171.556	23.998	9.7 sec
Global ILP	20.588	8.716	34 sec
Global Greedy	24.820	8.900	0.2 sec

Our dataset consists of 252 citizen scientist generated trajectories for 50 ants, with between 2 and 8 trajectories per ant. To compute the ant trajectories, we construct the interaction graph G, as described in Section 3. For our dataset, the graph contains $4,246$ vertices and $10,494$ edges. We apply the five local methods and two of the global methods (greedy and integer linear programming) to build consensus trajectories.

We computed seven different consensus trajectories for each ant: five based on the local algorithms, and two based on the global algorithms. An overview of results is in Table 1 and in Fig. 4(a). We compare all seven, as well as the trajectories computed by the automated system, by measuring average and worst root-mean-square error of the Euclidean distance between pairs of points of computed and ground truth trajectories. We notice that the approximate dimensions of an ant in our video are 60×15 pixels.

Among the local approaches, the local median is best. The local mean is negatively impacted by the outliers in the data. The Fréchet approach, Homotopy median, and Buffer median perform poorly. This could be due to the very self-intersecting trajectories making these algorithms miss entire pieces. We used the default values for all the parameters in these algorithms; a careful tuning will likely improve accuracy. In the Fréchet approach we tried 50 different random orders of merging trajectories.

The two global approaches perform similarly with respect to quality. There are only few segments of trajectories in which the results differ. It is important to emphasize that the global approaches outperform the automated solution for the cases where many citizen scientists make the same mistake; see Fig. 4(a). Such cases do happen in practice; see the full version [5] for an example. In this case, none of the local algorithms have a chance to recover a correct trajectory. Only the global approaches allow us to identify the correct ant and produce accurate results. On the other hand, the automated system is more accurate for "simpler" ants. We stress again that the ground truth data is inherently biased towards the automated solution because it was obtained by modifying the trajectories obtained from the automated solution.

A big challenge for the existing automated systems is tracking ants in long videos. For long videos (e.g., hundreds or even thousands of hours), automated tracking methods are not reliable. Whenever such algorithms loose tracking, the error quickly accumulates and a trajectory often is not recovered. Our global approaches naturally resolve this problem. We evaluate tracking accuracy as the percentage of ants correctly "tracked" at a given timestamp; here, we consider the ant correctly tracked if the dis-

(a)   (b)
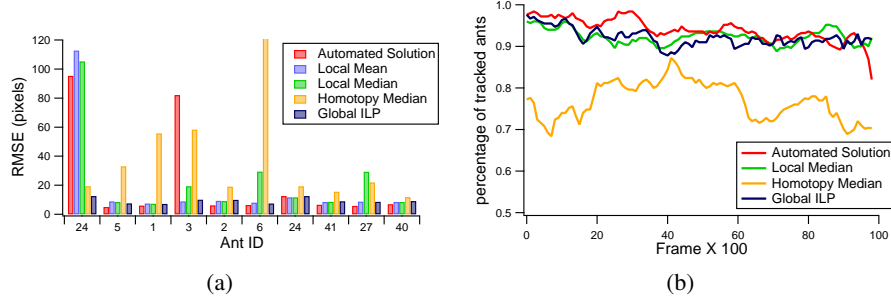
Fig. 4: (a) Root-mean-square error for the 10 "most movable" ants (according to the ground truth). (b) Comparison of tracking accuracy between automated solution and proposed algorithms.

tance between the ground truth and our trajectory is less than 15 pixels (typical width of ant head). The accuracy of the automated solution decreases over time, and by the end of the 5-minute video it is below 87% accuracy; see Fig. 4(b). Our algorithms are steadily over 90% accuracy over the entire video.

Synthetic Dataset. In order to validate our global approach on a larger dataset, we generate a collection of synthetic graphs. Here we test our algorithms for the SCP problem, that is, the algorithms for computing disjoint paths on graphs, rather than for extracting optimal trajectories. We construct a set of directed acyclic graphs having approximately the same characteristics as the interaction graph computed for the real-world dataset. The graph construction follows the same pipeline as described in Section 3; every graph is generated for $k \leq 50$ ants and $T \leq 100$ timestamps. The ants form k vertices for the initial timestamp; on every subsequent timestamp pairs of ants meet with probability 0.4 (the constant estimated for the real-world graph). Thus, for every timestamp, we have from $k/2$ to k vertices. The pairs of ants meeting at a timestamp are chosen randomly with the restriction that indegrees and outdegrees of every vertex in a graph are equal and at most 2. By construction of the graphs, we naturally get "ground truth" paths for all ants. Next we generate citizen scientist trajectories in two scenarios: $2 - 8$ trajectories (as in the real-world dataset) and $15 - 20$ trajectories per ant. A citizen scientist tracking an ant is modeled by a path starting at the source of the ant. At every junction vertex (with outdegree 2) there is fixed probability $P(error)$ of making a mistake by switching from tracking the current ant to the other one. If $P(error) = 0$, then user trajectories always coincide with the ground truth paths; if $P(error) = 0.5$, the trajectories may be considered as random walks on the graph.

We evaluate the greedy heuristic (GREEDY) and the linear program with rounding (LP+ROUNDING). As the latter heuristic is a randomized algorithm, we report the best result over 5 runs for a given input. For small instances, we also compute an exact solution using the integer linear program (ILP). We analyze the precision of the algorithms under various parameters. For every edge of the graph G, we say that it is correctly identified if both the algorithm and the ground truth assign the edge to the same path. The precision is measured as the fraction of correctly identified edges in G: a value of 1 means that all paths are correct. As in the real-world dataset, we consider a sce-
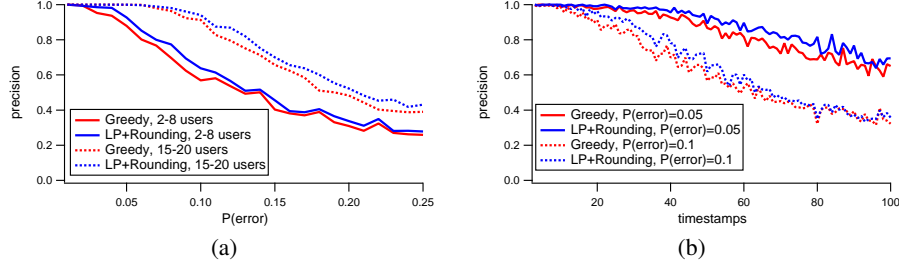
Fig. 5: Precision of the algorithms on the synthetic dataset. Solid lines represent average values over 5 runs of the algorithms for a given error/timestamp. (a) Results for $k = 50$ ants, $T = 100$ timestamps (5-minute video segment), and various values of $P(error)$. (b) Results for $k = 50$ ants, $2 − 8$ user trajectories per ant, and various number of timestamps.

nario with $k = 50$ ants. As expected, increasing the probability of making a mistake decreases the quality of the solution; see Fig. 5(a). However, increasing the number of user trajectories does help. Both algorithms recover all paths correctly if the number of trajectories for each ant is more than 15, even in a case with $P(error) = 0.1$. On the other hand, with $P(error) > 0.2$ the precision drops to under 50%. Although we cannot definitively measure the accuracy of all citizen scientists, empirical evidence from our experiment indicates that the error rate was very low: $P(error) = 0.02$. This is an order of magnitude lower than the upper limit on errors that our algorithms can handle. We also consider the impact of video length on the precision of the algorithms; see Fig. 5(b). Not surprisingly, precision is higher for short videos: for 2-minute segments (40 timestamps) and $P(error) = 0.05$, LP+ROUNDING produces the correct paths, while for 5-minute segments (100 timestamps) only 60% of paths are correct.

We also analyze the effectiveness of the GREEDY and LP+ROUNDING algorithms for the SCP problem with $P(error) = 0.2$ and $2 − 8$ trajectories per ant; see Fig. 6. To normalize the results, values are given as a percentage of the *cost* of an optimal fractional solution (FLP) for the SCP problem. Note that the ILP results are in the range$[0.98, 1.0]$, which means that an optimal integer solution is always very close to the optimal fractional solution. Both GREEDY and LP+ROUNDING perform very well, achieving ≈ 0.85 of the optimal solution. These two algorithms produce similar results, with LP+ROUNDING usually outperforming GREEDY.

Running times are shown in Fig. 7. As expected, the greedy algorithm, with complexity dependent linearly on the size of the graph, finishes in under few milliseconds. The ILP is also relatively quick on the real-world dataset, computing the optimal solution within a minute. On synthetic data and more erroneous real-world data the ILP approach is applicable only when the number of ants is small, e.g., $k < 25$. For larger values of k, the computation of optimal disjoint paths takes hours. On the other hand, LP+ROUNDING is fast: the real-world instances with $k = 50$ ants and $T = 100$ timestamps are processed within a minute. GREEDY takes only $2 − 3$ seconds on the largest instances. Hence, the running times of our algorithms (except ILP) are practical.
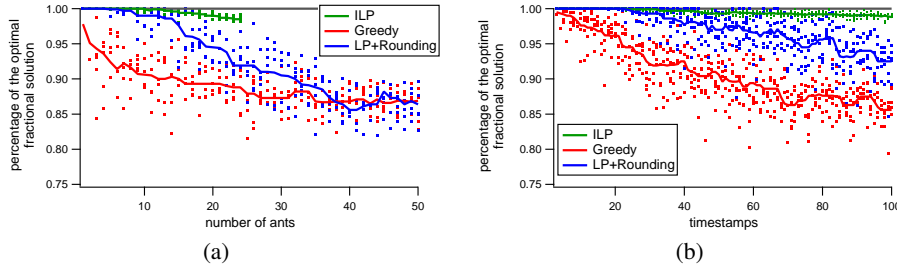
Fig. 6: Quality of the algorithms: ratio between the *cost* of obtained solution and the optimal (fractional) *cost*. Results for single instances are depicted by dots and solid curves show the average values over 5 runs for a given number of ants/timestamps. Note that the y-axis starts at value 0.75. (a) Results for $T = 100$ timestamps, $P(error) = 0.2$, and $2 - 8$ user trajectories per ant. (b) Results for $k = 20$ ants, $P(error) = 0.2$, and $2 - 8$ user trajectories per ant.

6 Conclusions and Future Work

We described a system for computing consensus trajectories from a large number of input trajectories, contributed by untrained citizen scientists. We proposed a new global approach for computing consensus trajectories and experimentally demonstrated its effectiveness. In particular, the global approach outperforms the state-of-the-art in computer vision tools, even in their most advantageous setting (high resolution video, sparse ant colony, individually painted ants). In reality, there are hundreds of thousands of hours of video in settings that are much more difficult for the computer vision tools and where we expect our citizen science approach to compare even more favorably.

A great deal of challenging problems remain. Arguably, the best method would be to track "easy ants" and/or "easy trajectory segments" automatically, while asking citizen scientists to solve the hard ants and hard ant trajectory segments. In such a scenario, every input trajectory will describe a part of the complete ant trajectory, which would require stitching together many short pieces of overlapping trajectories.

References

1. Alt, H., Godau, M.: Computing the Fréchet distance between two polygonal curves. Internat. J. Comput. Geom. Appl. 5(1), 75–91 (1995)
2. Betke, M., Hirsh, D., Bagchi, A., Hristov, N., Makris, N., Kunz, T.: Tracking large variable numbers of objects in clutter. In: CVPR. pp. 1–8. IEEE Computer Society, Washington (2007)
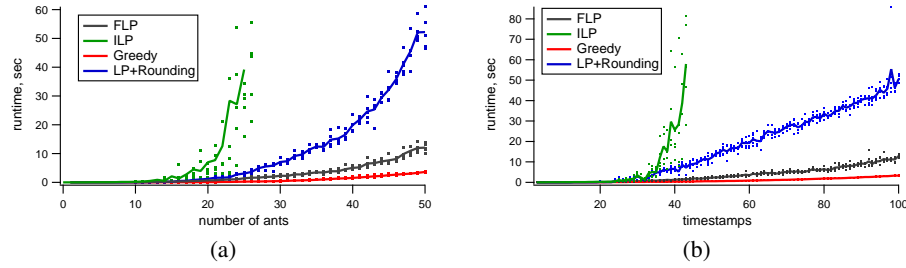
Fig. 7: Running time of the algorithms with $P(error) = 0.2$ and $2 - 8$ trajectories per ant. The results for single instances are depicted by dots, while solid lines represent average values over 5 runs for a given number of ants/timestamps. (a) Results for $T = 100$ timestamps and various number of ants. (b) Results for $k = 50$ ants and various length of a video.

3. Buchin, K., Buchin, M., Gudmundsson, J.: Constrained free space diagrams: a tool for trajectory analysis. Int. J. Geogr. Inf. Sci. 24(7), 1101–1125 (2010)
4. Buchin, K., Buchin, M., Kreveld, M., Lffler, M., Silveira, R., Wenk, C., Wiratma, L.: Median trajectories. Algorithmica 66(3), 595–614 (2013)
5. De La Cruz, L., Kobourov, S., Pupyrev, S., Shen, P., Veeramoni, S.: Computing consensus curves. Arxiv report http://arxiv.org/abs/1212.0935 (2013)
6. Even, S., Itai, A., Shamir, A.: On the complexity of timetable and multicommodity flow problems. SIAM J. Comput. 5(4), 691–703 (1976)
7. Fletcher, M., Dornhaus, A., Shin, M.: Multiple ant tracking with global foreground maximization and variable target proposal distribution. In: WACV. pp. 570–576. IEEE Computer Society, Washington (2011)
8. Khan, Z., Balch, T., Dellaert, F.: MCMC-based particle filtering for tracking a variable number of interacting targets. IEEE TPAMI 27(11), 1805–1819 (2005)
9. van Kreveld, M., Wiratma, L.: Median trajectories using well-visited regions and shortest paths. In: GIS. pp. 241–250. ACM, New York (2011)
10. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory 28(2), 129–137 (1982)
11. Maitra, P., Schneider, S., Shin, M.: Robust bee tracking with adaptive appearance template and geometry-constrained resampling. In: WACV. pp. 1–6. IEEE Computer Society, Washington (2009)
12. Marx, D.: Eulerian disjoint paths problem in grid graphs is NP-complete. Discrete Appl. Math. 143(1-3), 336–341 (2004)
13. Nguyen, N., Keller, S., Norris, E., Huynh, T., Clemens, M., Shin, M.: Tracking colliding cells in vivo microscopy. IEEE Trans. Biomed. Eng. 58(8), 2391–2400 (2011)
14. Poff, C., Hoan, N., Kang, T., Shin, M.: Efficient tracking of ants in long video with GPU and interaction. In: WACV. pp. 57–62. IEEE Computer Society, Washington (2012)
15. Trajcevski, G., Ding, H., Scheuermann, P., Tamassia, R., Vaccaro, D.: Dynamics-aware similarity of moving objects trajectories. In: GIS. pp. 1–8. ACM, New York (2007)
16. Veeraraghavan, A., Chellappa, R., Srinivasan, M.: Shape-and-behavior encoded tracking of bee dances. IEEE TPAMI 30(3), 463–476 (2008)
17. Yilmaz, A., Javed, O., Shah, M.: Object tracking: a survey. ACM Comput. Surv. 38, 1–45 (2006)