Using Odometry to Improve Swarm Robot Aggregation

Andrew Vardy

Department of Computer Science
Department of Electrical and Computer Engineering
Memorial University of Newfoundland, St. John's, Canada
(E-mail: av@mun.ca)

Abstract: Aggregation is a useful building block behaviour that can allow a swarm of robots to interact with each other or a user more easily. Previous work on swarm robot aggregation has assumed that the capabilities of individual robots are quite limited. We test whether incorporating odometry as an additional capability is helpful and make the argument that odometry is both realizable and biologically plausible. We propose an algorithm called ODOCLUST which takes inspiration from the BEECLUST algorithm but uses a continuously active odometry-based homing process to achieve more tightly packed robot aggregates more quickly than BEECLUST. Initial results in simulation suggest that high-fidelity odometry is not required in order to see these gains.

Keywords: swarm robotics, aggregation, odometry

1. INTRODUCTION

The ability of a group of robots to aggregate together is useful as a behavioural building block [3]. Robots with limited-range sensing and communication will be able to perceive each other and potentially communicate when in close proximity. An aggregated set of robots can also be more usable. For example, it would be easier to interact with or collect robots after they have aggregated together. So robot aggregation has been studied for these practical reasons, but the problem is also of interest because it is relatively easy to model mathematically [1]. Animals as diverse as bees, cockroaches, fish, and penguins aggregate in a self-organized fashion [4]. Locusts provide an example of a destructive moving aggregate [2].

Within swarm robotics, aggregation is usually studied with the assumption that the individual robots have very limited sensory and computational capabilities. This assumption has allowed the generation of research results which generalize to a wide variety of settings, from current robots measured on the scale of centimetres to future robots at the micro or nano scale. However, for implementation on current robots with an eye towards near-term practical applications there is a need to study and improve performance given the opportunities and constraints of available robots. In this paper we investigate the use of odometry for aggregation. Odometry, also known as *dead reckoning*, *path integration*, and sometimes *inertial navigation*, implies a self-contained system for measuring and integrating a robot's speed and/or acceleration to provide a pose estimate relative to its starting position. It is well understood that odometric systems will incur an inevitable cumulative error that tends to increase with distance travelled. All odometric sensors are *egocentric* in the sense that they measure some internal property of the robot as opposed to *allocentric* sensors which measure external aspects of the environment [13]. The cumulative error of an odometric system can be compensated by reference to allocentric sensors and map-like representations of the environment which are typically brought together through a Bayesian inference process [15].

The scale of expense and accuracy of odometric systems is vast. Components such as wheel encoders and smartphone-grade inertial measurement units (IMUs) cost on the order of tens of dollars or less, although the accuracy of such systems is not typically well-characterized. Integrated inertial navigation systems designed for ships and underwater vehicles cost on the order of hundreds of thousands of dollars and can achieve positional accuracies of less than 0.01% of distance travelled. Clearly for most swarm robotic applications the odometry solution would be based on inexpensive sensors such as encoders and inexpensive IMUs. Kernbach has investigated a novel strategy based on sensing periodic changes in the optical properties of moving gears [8]. Such an approach can provide relative localization for robots where the system's size and computational power are limited. Another possibility is visual odometry which tracks the relative displacement of coherently moving visual features to track a robot's motion [10]. The hardware costs for cameras are low, but computational costs, power consumption, and the need for calibration are considerations.

There is an interesting debate concerning the extent and purpose of the principle of minimalism in swarm robotics [12]. From the perspective of practical applications on current robots, it seems reasonable to employ capabilities that are reasonably inexpensive and readily available. In our opinion, low-end odometric systems fit this criterion. Meanwhile, we focus on systems with very limited allocentric sensing capability. In addition to odometry, the only sensors used in our approach are short-range proximity sensors and single-pixel light sensors which face downwards to detect the region adjacent to the environment's boundary. Except for the addition of odometry, our approach is most directly related to the BEECLUST algorithm which is based on robot-to-robot contact events rather than attempting to estimate the lo-

† Andrew Vardy is the presenter of this paper.

cal object density directly [11]. With sufficient sensing and computational capabilities the overall centroid of the swarm can be determined and the problem becomes trivial. The challenge is to enable aggregation using inexpensive, realizable components. While we expand the 'toolchest' to include odometry, we retain purely local detection of contacts as our only feedback on the positions of other robots. We would direct the reader to [7] for a recent and comprehensive review of work in swarm robot aggregation which pays special consideration to the required sensory and computational requirements of the algorithms proposed.

Odometry is not only easily realizable from the technical perspective, it is also of fundamental importance to many insects, particular the social insects which have been so inspirational to the field of swarm robotics. Odometry is generally known as path integration within the insect navigation community and is used in combination with visual, olfactory, and other cues which can be treated as landmarks. The relative importance of path integration to central-place foraging animals has been described as follows: "In bees and ants, path integration employing a skylight compass is the predominant mechanism of navigation, but geocentred landmark-based information is used as well." [18]. The classic example of stunningly accurate path integration in insects is the desert ant *Cataglyphis fortis* which embarks upon foraging trips hundreds of metres in length and upon encountering a food item will head straight for the nest—or will head in the corresponding direction in displacement experiments [9]. A more recent review considers the evidence for landmark-based information functioning in the context of a highly capable path integration system [5].

This paper represents a first step in the use of odometry for swarm robot aggregation. The initial plan was to memorize a sequence of contact locations and compute their average as an estimate of an existing aggregate's centroid. However, interesting results were obtained by storing only the last contact location. The ODOCLUST algorithm described below alternates between odometry-based homing to the last contact point and random motion. This is similar in principle to the BEECLUST algorithm except that BEECLUST-controlled robots enter a passive waiting state when another robot is contacted. Our algorithm will react to contact with another robot by actively homing back to the last contact point. Since the contact point shifts from one time-step to the next, the robot oscillates and this oscillation leads to the formation of tighter aggregates. Both BEECLUST and ODOCLUST will stay in the waiting/homing state for a randomly chosen interval and will then continue random motion which allows the robot to potentially reach another aggregate. In both cases a larger aggregate has a higher probability of attracting members, but there is no guarantee of complete convergence.

In the next section we describe the two algorithms tested. We then present experimental results from deploying these algorithms to control a set of robots in simulation. The paper then concludes with conclusions and consideration of future work.

2. METHODS

The BEECLUST algorithm and our odometry-based variant, ODOCLUST, are both well-described as finite state machines. Prior to presenting these state machines we consider the sensory information that guides their behaviour. Our experimental results are based on simulations of Pioneer 3-DX robots. We have adopted the following simplified sensor suite for these robots:

Proximity sensors: The proximity sensors have a range of 20 cm. The radius of the robots is approximately 30 cm. The actual range values from the sensors are not utilized, rather we set Boolean variables to indicate that an object is within sensing range or not. There are front, left, and right sensors with the left and right sensors at an angle of $\pm 40^\circ$ to the robot's forward axis. Within figure 1 the Boolean value *bumped* indicates that one of the three proximity sensors has been tripped. When entering the TURN state, the robot will turn away from the contact (e.g. turn left if the right sensor is tripped).

Floor sensor: The floor sensor is a downward-looking light sensor that can sense the lighter colour of the floor near the boundary walls. This colour is thresholded to yield a Boolean value which is indicated in figure 1 as *boundary*.

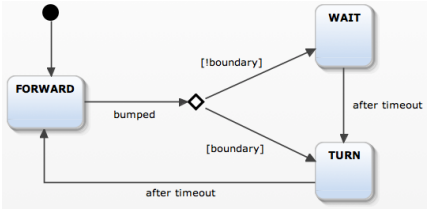
2.1. Finite State Machines

Figure 1 presents the finite state machines that represent our tested algorithms.

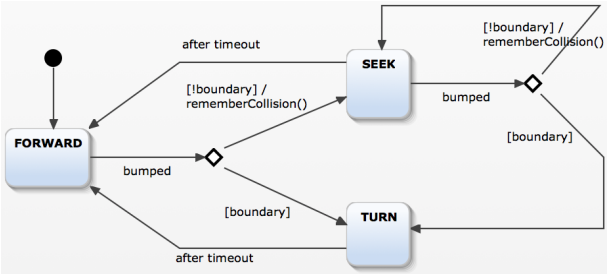
The first algorithm, was modelled after the BEECLUST aggregation strategy [11]. The original BEECLUST was inspired by the tendency of young bees to aggregate together within a warm area of the bees' hive. Thus, it can be seen as solving two problems: aggregation and gradient climbing[1]. We focus on aggregation in the absence of any other distinguishing signals from the environment (other that detecting the boundary area so as to avoid it). Like the original BEECLUST our version (figure 1(a)) has FORWARD and TURN states that cause the robot to travel across the environment, re-bounding after collisions. If a contact occurs in the FORWARD state away from the boundary then the WAIT state is entered under the assumption that the robot has just contacted an aggregate of robots. A random wait time is selected after which the TURN state is entered which causes the robot to turn away from the collision, again by a randomized amount. The main difference from the original BEECLUST is that there is no measure of illumination modulating the probability of entering the WAIT state. WAIT state is only entered after a contact between robots.

BEECLUST leads to aggregation because contacts between robots form aggregates and larger aggregates have

[1]It should be noted that BEECLUST employs no comparison of temperatures or direct gradient sensing. Thus, the ability of the swarm to converge near the point of maximum temperature is an emergent phenomenon.



(a) BEECLUST



(b) ODOCLUST

Fig. 1: Finite state machines for BEECLUST (a) and ODOCLUST (b). Not all relevant details are provided. For example, the random selection of timeout values is not given, nor are details on whether the robot turns right or left in the TURN state.

a higher probability of inducing more collisions due to their sheer size.

The ODOCLUST algorithm is based on odometry-based homing to the last point of contact with another robot. Thus, there is no WAIT state. Like BEECLUST the robot begins in the FORWARD state until it encounters another robot (boundary condition false) or the boundary itself. Upon encountering another robot the position at contact is recorded and the robot will apply odometry-based homing to SEEK this position. On its own this behaviour would prevents the robots from exploring in search of other aggregates, so after a fixed timeout period there is a transition back to the FORWARD state.

ODOCLUST leads to aggregation for the same reason as BEECLUST—collisions are opportunities to form or join aggregates and larger aggregates attract more members. However with BEECLUST the shape of an aggregate evolves rather slowly as robots join after a period of random motion (FORWARD/TURN cycles), WAIT motionless within the aggregate for some time, then begin another motion. With ODOCLUST the robots in the SEEK state are constantly shifting and moving towards the last point of contact, allowing many more opportunities for the aggregate to condense.

For both BEECLUST and ODOCLUST a robot in the FORWARD state will very often encounter the environment's boundary and enter the TURN state. This combination of FORWARD and TURN states promotes exploration of the environment. The WAIT and SEEK states encourage the robots to stay within or return to an aggregate.

It is clear that the timeout parameters are important for both BEECLUST and ODOCLUST. When the TURN state is entered the timeout period is set randomly in the range $[0, 20]$ with 20 steps corresponding to an approximately 90° turn. For the timeout period of BEECLUST's WAIT state we tested values in the set $\{200, 400, 600, 800, 1000, 2000\}$ and found 800 to yield the best results. For the timeout period of ODOCLUST's SEEK state we tested values from the same set and again found 800 to yield the best results. Both of these parameter selection tests were run within the $4m \times 4m$ arena using 5 robots (see section 3).

2.2. Odometry

The odometry motion model proposed in [15] was adopted to simulate cumulative odometric error. The robot's most recent and previous poses are denoted (x', y', θ') and (x, y, θ). Note that these are true poses which are available within the simulator. The motion between these two poses can be described by an initial rotation δ_{rot1}, a translation of length δ_{trans}, and a final rotation of angle δ_{rot2}. These three motion parameters are determined as follows:

$$\delta_{rot1} = \text{atan2}(y' - y, x' - x) - \theta \quad (1)$$

$$\delta_{trans} = \sqrt{(x' - x)^2 + (y' - y)^2} \quad (2)$$

$$\delta_{rot2} = \theta' - \theta - \delta_{rot1} \quad (3)$$

The robot's previous odometric pose estimate is $(\bar{x}, \bar{y}, \bar{\theta})$ and the simulation task is to generate the new estimate $(\bar{x}', \bar{y}', \bar{\theta}')$. This is achieved by taking the δ motion parameters and adding appropriate noise terms, then adding this simulated motion to $(\bar{x}, \bar{y}, \bar{\theta})$. Thrun et al. propose the following model for the variances of the Gaussian noise processes for each motion parameter:

$$\sigma^2_{rot1} = \alpha_1 \delta^2_{rot1} + \alpha_2 \delta^2_{trans} \quad (4)$$

$$\sigma^2_{trans} = \alpha_3 \delta^2_{trans} + \alpha_4 (\delta^2_{rot1} + \delta^2_{rot2}) \quad (5)$$

$$\sigma^2_{rot2} = \alpha_1 \delta^2_{rot2} + \alpha_2 \delta^2_{trans} \quad (6)$$

This model is powerful in that with appropriate values for the α parameters, a wide variety of different patterns of odometric noise can be accommodated. However, an exploration of this 4-dimensional parameter space is cumbersome and we are not focused on modelling the details of a particular system. Instead, our aim is to explore a few coarsely-defined regimes of accuracy of the odometric system. So we define a single α parameter and set the parameters above as follows: $\alpha_1 = 0.01\alpha$, $\alpha_2 = 100\alpha$, $\alpha_3 = \alpha_2$, $\alpha_4 = 0$. The scalar multiples for α_1 and α_2 differ strongly because α_1 governs angular error while α_2 governs translational error. We can then generate Gaussian distributed random numbers with the appropriate σ values defining their standard deviations. These random noise values are added to the three motion parameters. Finally, the new odometric pose estimate is obtained as follows:

$$\bar{x}' = \bar{x} + \delta_{trans} \cos(\bar{\theta} + \delta_{rot1}) \quad (7)$$

$$\bar{y}' = \bar{y} + \delta_{trans} \sin(\bar{\theta} + \delta_{rot1}) \quad (8)$$

$$\bar{\theta}' = \bar{\theta} + \delta_{rot1} + \delta_{rot2} \quad (9)$$

We tested various values for α to obtain a range of odometric inaccuracy. Figure 2 shows scatter plots for a robot that starts at the centre of the arena, facing to the right and simply moves forward. The points plotted are its estimated positions after 60 time steps (approximately 1.08 m of travel). For the experiments below we consider $\alpha = 0$ (no error), $\alpha = 1$ (low error), $\alpha = 2$ (moderate error) and $\alpha = 5$ (extreme error).

It is important to note that our odometry error model does not account for the error potentially induced by collisions with the boundary or with another robot. This error is strongly dependent upon the particular sensors in use. For example, if integrating wheel angles via optical encoders, wheel slippage that occurs during a collision may lead to significant error. If using inertial sensors then it may be possible to filter the acceleration signals at the point of impact, but this is dependent on filter complexity.

This discussion has focused on the simulation of odometry. Given a particular robot, odometry is typically computed by determining δ_{trans}, δ_{rot1}, and δ_{rot2} or equivalent parameters from available sensors and integrating these values into the pose estimate using equations such as (7)-(9).

2.3. Homing

Given an odometric estimate of position $(\bar{x}', \bar{y}', \bar{\theta}')$ we can command the robot to drive to any other position. Of course, the robot's ability to arrive at this position depends on the degree of odometric error. The control law used in our experiments determines the position of the target in the robot's reference frame: (g_x, g_y), where the robot's forwards heading direction is taken as the $x-$axis. For example, if homing to a position directly 10 m ahead of the robot we would have target position $(10, 0)$. The control law is

$$v = K_f g_x \quad (10)$$

$$\omega = K_\omega g_y, \quad (11)$$

where v is the robot's forwards speed, ω is angular speed, and K_f and K_ω are appropriately chosen constants. We also set hard limits on v and ω to keep the robot's speed manageable and prevent overshoot. For ODOCLUST it was found useful to set a minimum magnitude for v so that the robot was constantly moving back and forth around the last point of contact.

3. EXPERIMENT RESULTS

The BEECLUST and ODOCLUST algorithms were implemented and tested using the V-REP robot simulation framework [6]. The robots tested were Pioneer 3-DXs with the simplified sensor suite described in section 2. Parameter selection and the first set of experiments took place within a $4m \times 4m$ square arena using 5 robots. We also report results on a larger $8m \times 8m$ arena with 5 and 20 robots. Figure 3 shows a screenshot of the simulation in progress. A video showing the performance of ODOCLUST in the larger environment with 20 robots is available at `https://youtu.be/QB3MHSvGCkY`.

We utilize two different metrics to judge aggregation performance over time. By defining a particular desired robot-to-robot distance within an aggregate we can determine what pairs of robots are within this distance and thereby produce a set of edges between. Taking the robots as vertices yields a graph. The largest connected component of this graph represents the largest aggregate. We define *percentage completion* (PC) as the size of this largest aggregate divided by the number of robots. The approximate radius of the robots, out to the outer limit of the proximity sensors detection range is 0.5 m. Therefore two robots just barely in sensory contact with each other would be separated by 1.0 m. To introduce some slack, we set the distance threshold for defining edges between robots to 1.5 m. With the exception that the distance threshold is set to a different value, PC is the same or closely related to similar metrics used by others [7, 14].

The requirement to define such a threshold is one weakness of the percentage completion metric. It also does not reflect the compactness of aggregates which may be a desirable feature. Therefore, inspired by a similar metric [14] we define *total distance* (TD) as the sum of distances between all robot pairs. While PC should ideally reach 100%, TD should ideally reach a minimal value.

Figure 4 shows plots of the PC and TD metrics averaged over 30 runs each lasting 5000 time steps with 5 robots in the $4m \times 4m$ arena. For ODOCLUST we tested α values from the set $\{0, 1, 2, 5\}$. ODOCLUST's performance clearly has some dependency on α but is certainly competitive with BEECLUST for all tested α. In terms of TD, ODOCLUST achieves tighter aggregates

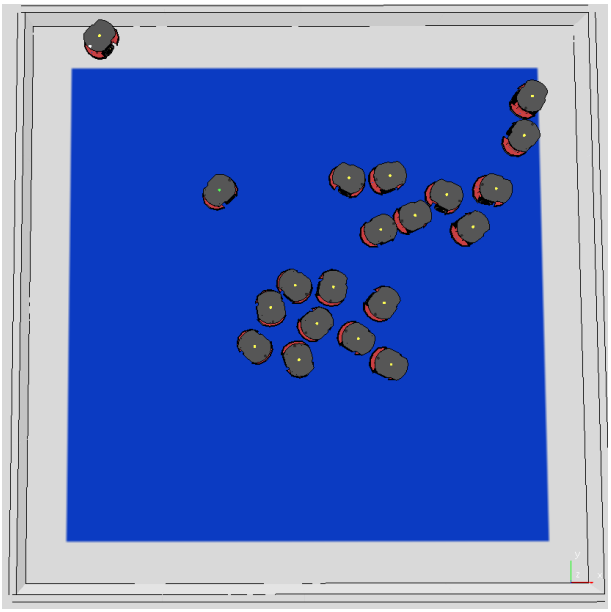


Fig. 3: Screenshot of the V-REP simulation of 20 robots controlled by ODOCLUST in the larger $8m \times 8m$ arena. This image was captured near the end of a 20,000 time step run.

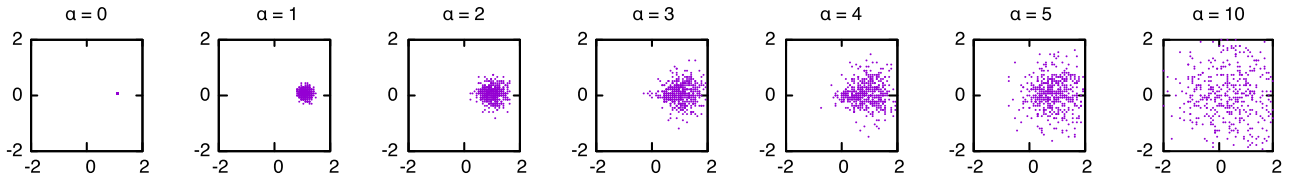


Fig. 2: Scatter plots showing the final position of a robot that begins at position $(0, 0)$ and travels to the right for 60 time steps using different values of the α parameter to simulate odometric noise.

than BEECLUST and suffers from less variability.

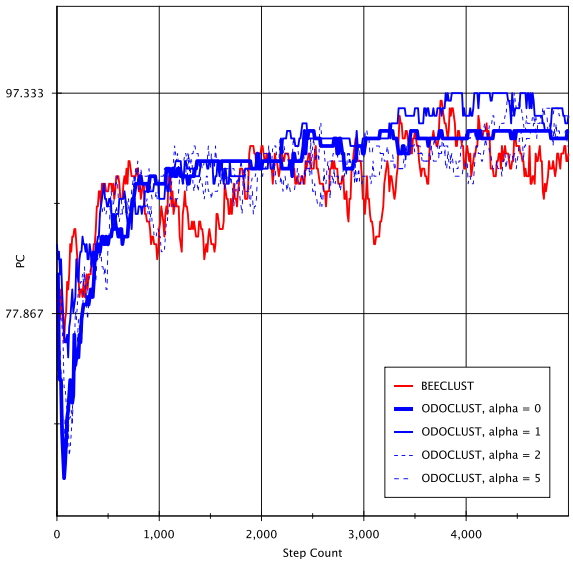
Figure 5 shows similar results for the operation of 5 robots within the larger arena. For clarity we show only the results for $\alpha = 1$ (low noise). The gap in performance is now more pronounced with ODOCLUST achieving higher PC and lower TD.

Finally, figure 6 presents the results for 20 robots operating within the larger arena, again with $\alpha = 1$. Increasing the number of robots has clearly broadened the performance gap. The performance of BEECLUST actually decreases with time according to both metrics while the improvement in ODOCLUST is close to monotonic. Figure 7 shows the final positions of all robots over 30 runs for each algorithm. The final aggregates formed by ODOCLUST incorporate the majority of robots and are much more compact than the aggregates formed by BEECLUST.

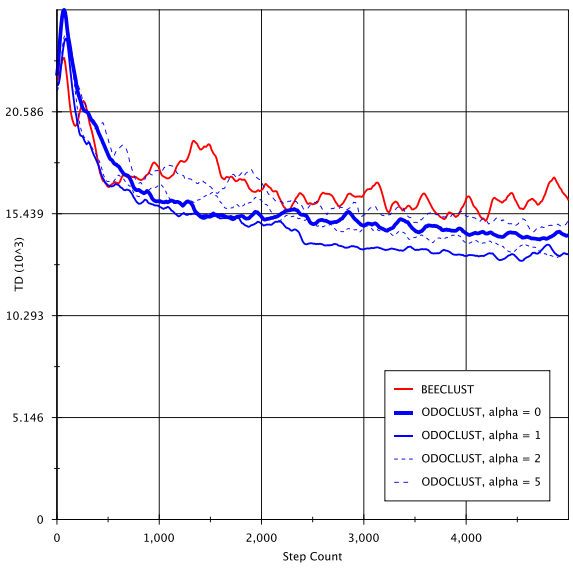
4. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed ODOCLUST, a novel robot aggregation method and compared its performance to the well-known BEECLUST method. ODOCLUST has been found to achieve superior performance, particularly within the larger arena with more robots. Our tests also indicate that the algorithm remains competitive with BEECLUST even for very inaccurate odometry. However, we should caution that these findings will need to be re-assessed after performing more extended trials and testing the algorithm on actual mobile robots.

We have proposed to include odometry within the set of capabilities for swarm robotic systems because of the power, affordability, and biological relevance of odometry. This work is part of a larger research program that assesses the benefits and costs of introducing increased spatial awareness in otherwise simplistic robot collectives. Our previous work on distributed object sorting supposes that robots can store sensory snapshots which can be used to return to previously visited clusters [17]. This capability is utilized by our *cache consensus* algorithm to compare currently viewed clusters with remembered clusters and retain only the larger cluster of each object type in memory. This allows the robots to come to consensus and quickly form one homogeneous cluster of each type, even without any communication. The required navigational competence of returning to previously visited places can be achieved by the increasingly mature methods of Simultaneous Localization and Mapping (SLAM) [15], through bio-inspired visual homing



(a) Percentage Completion



(b) Total Distance

Fig. 4: Plots of PC and TD for the $4m \times 4m$ arena and 5 robots. Note that the two solid lines represent the two tested algorithms without odometric noise. Dashed lines represent ODOCLUST with various levels of α. These plots are averaged across 30 runs.

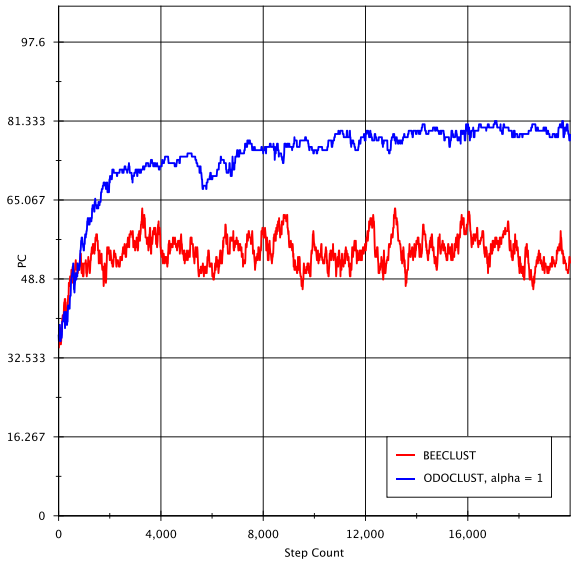
[16], or various other means. This represents one avenue of future work where we extend beyond the relative localization offered by odometry to the absolute localization offered by these techniques. We are also interested in going in the other direction and using pure odometry for object sorting.

Odometry may be helpful in other tasks where the instantaneous sensory information is ambiguous but can be made coherent by comparing information at multiple nearby places. Examples of such tasks include dispersion (the inverse of aggregation), chemotaxis (moving to align with chemical gradients), pattern formation, foraging, self-assembly, collective exploration, and collective transport [3].

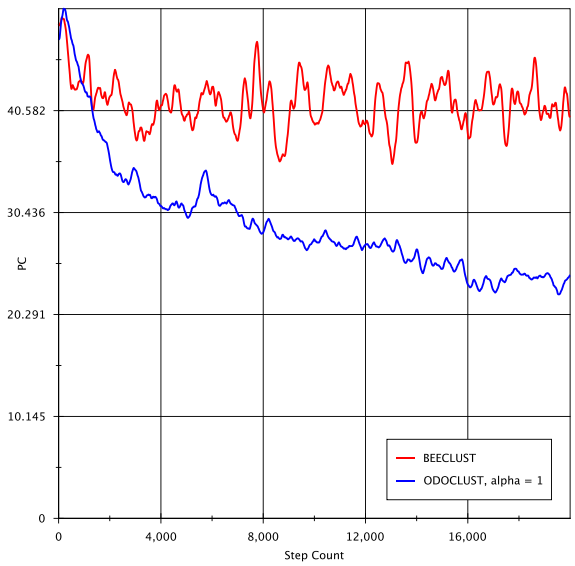
The case could be made that adding odometric information is going against the principle of minimalism that guides so much work in swarm robotics. Yet the importance of odometry in animals, particularly in social insects such as bees and ants suggests that incorporating it is beneficial and may strengthen the often fruitful correspondence between biological models of behaviour and robotics.

REFERENCES

[1] L. Bayindir and E. Sahin. Modeling self-organized aggregation in swarm robotic systems. In *Swarm*

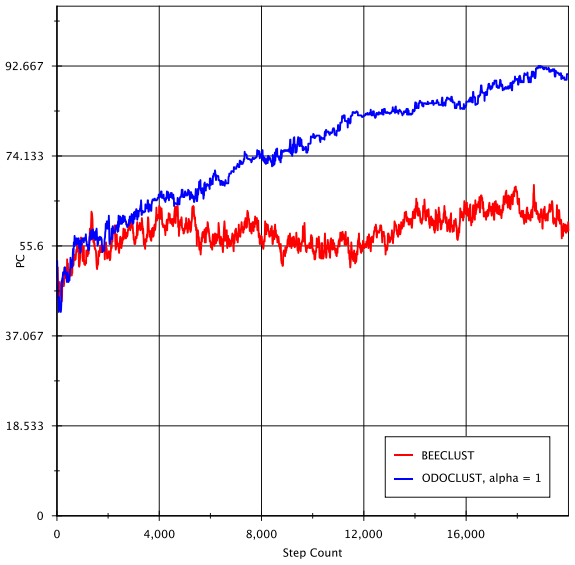


(a) Percentage Completion

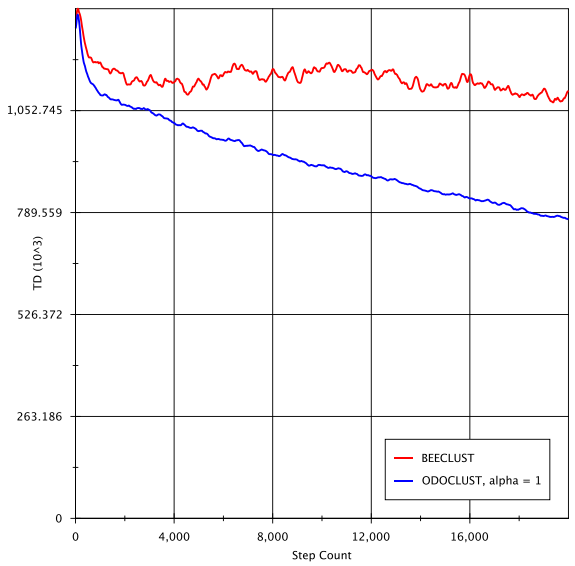


(b) Total Distance

Fig. 5: Plots of PC and TD for the $8m \times 8m$ arena and 5 robots. These plots are averaged across 30 runs.



(a) Percentage Completion



(b) Total Distance

Fig. 6: Plots of PC and TD for the $8m \times 8m$ arena and 20 robots. These plots are averaged across 30 runs.

Intelligence Symposium, 2009. SIS'09. IEEE, pages 88–95. IEEE, 2009.

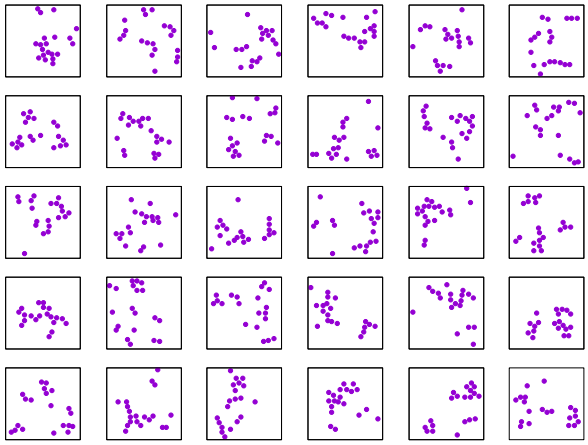
[2] M. Beekman, G. A. Sword, and S. J. Simpson. Biological foundations of swarm intelligence. In C. Blum and D. Merkle, editors, *Swarm Intelligence*, Natural Computing Series, pages 3–41. Springer Berlin / Heidelberg, 2008.

[3] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

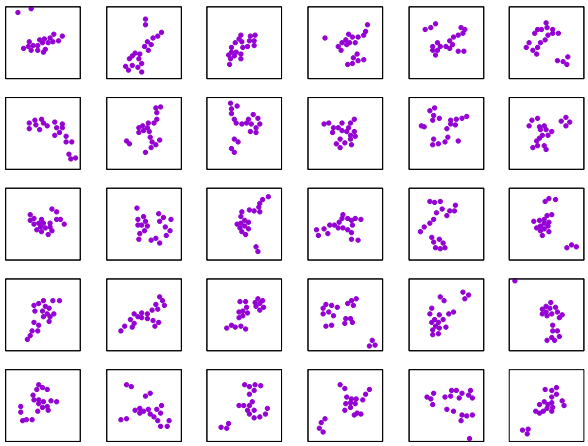
[4] S. Camazine. *Self-organization in biological systems*. Princeton University Press, 2003.

[5] T. Collett and M. Collett. Memory use in insect visual navigation. *Nature Reviews Neuroscience*, 3:542–552, 2002.

[6] M. Freese, S. Singh, F. Ozaki, and N. Matsuhira. Virtual robot experimentation platform v-rep: a versatile 3d robot simulator. In *Simulation, Modeling, and Programming for Autonomous Robots*, pages 51–62. Springer, 2010.



(a) BEECLUST



(b) ODOCLUST

Fig. 7: Snapshots of the final positions of all robots over 30 runs for the $8m \times 8m$ arena and 20 robots.

[7] M. Gauci, , J. Chen, W. Li, T. J. Dodd, and R. Groß. Self-organised aggregation without computation. *The International Journal of Robotics Research*, 33(9):1145–1161, 2014.

[8] S. Kernbach. Encoder-free odometric system for autonomous microrobots. *Mechatronics*, 22(6):870–880, 2012.

[9] M. Müller and R. Wehner. Path integration in desert ants, cataglyphis fortis. *Proceedings of the National Academy of Sciences*, 85(14):5287–5290, 1988.

[10] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–652. IEEE, 2004.

[11] T. Schmickl, R. Thenius, C. Moeslinger, G. Radspieler, S. Kernbach, M. Szymanski, and K. Crailsheim. Get in touch: cooperative decision making based on robot-to-robot collisions. *Autonomous Agents and Multi-Agent Systems*, 18(1):133–155, 2009.

[12] A. J. Sharkey. Swarm robotics and minimalism. *Connection Science*, 19(3):245–260, 2007.

[13] R. Siegwart, I. Nourbakhsh, and D. Scaramuzza. *Introduction to Autonomous Mobile Robots*. MIT Press, Cambridge, MA, 2nd edition, 2011.

[14] O. Soysal, E. Bahçeci, and E. ŞahİN. Aggregation in swarm robotic systems: Evolution and probabilistic control. *Turkish Journal of Electrical Engineering & Computer Sciences*, 15(2):199–225, 2007.

[15] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[16] A. Vardy and R. Möller. Biologically plausible visual homing methods based on optical flow techniques. *Connection Science*, 17(1/2):47–90, 2005.

[17] A. Vardy, G. Vorobyev, and W. Banzhaf. Cache consensus: Rapid object sorting by a robotic swarm. *Swarm Intelligence*, 8(1):61–87, 2014.

[18] R. Wehner, B. Michel, and P. Antonsen. Visual navigation in insects: Coupling of egocentric and geocentric information. *Journal of Experimental Biology*, 199:129–140, 1996.