

A distributed formation algorithm to organize agents with no coordinate agreement

Gregory Studer¹ and Inman Harvey¹

Centre for Computational Neuroscience and Robotics, University of Sussex

Abstract. In this paper we present an extension of the ShapeBugs distributed formation algorithm which enables 2D mobile agents to agree on a consensus coordinate system starting from no coordinate agreement. The participating agents require only local communication and local distance and motion information. Because this distance and motion information can in many cases be obtained implicitly through software motor approximation and attenuation or time-of-flight in communication, this approach has the potential to globally coordinate general mobile communicating agents without additional sensor requirements. The extended algorithm also remains robust against agent incapacitation and disorientation.

1 Introduction

In the areas of robotics and distributed systems, from many directions, a strong research effort has developed towards controlling autonomous agents with low power requirements and simple sensor capabilities [1][2]. This research has been partially motivated by continuing advances in hardware: microcontrollers and digital communication technology have become cheap, widely available, and in many cases very efficient. The technology enables new applications, and large numbers of simple, self-directed robots show high potential for use in sensor grids [3], resource harvesting [4], and group transport [5][6], amongst many other proposals. As we begin to design robots of smaller size or larger number, however, there are strong technical challenges to our ability to individually control, coordinate, or adequately power them. Centralized supervision of complex agents with virtually unlimited communication range, while successful in many areas, often scales poorly when there are hundreds or thousands of agents to control [7]. Ideally we would like to program global behavior in a distributed way, using assumptions of limited communication and agents acting autonomously. Overcoming the limitations of local communication and independent agent action to achieve global coordination, using minimum hardware, is the defining challenge in making simple agents practical.

The well-studied 2D formation task provides a particularly good starting point for this research, as it is simple to describe with applications in theory and practice. Given a set of points and a set of mobile agents, the formation task is simply to arrange the agents on the points (or perhaps an isometric set of

points)[2]. Real-world tasks for multiple agents often involve forming particular shapes, as in the previous examples of sensor grids and group transport. On the theoretical side, agents which are able to arrange into arbitrary, large shapes have established global information from local interactions [6].

A particularly complete heuristic approach to the formation problem called the “ShapeBugs” algorithm is described by Cheng et al [3], which synchronizes local agent coordinate systems through repeat trilateration. Given a small seed group of initially oriented agents, an arbitrary (though connected) global shape much larger than an individual agent’s communication radius can be formed. The algorithm is robust to sensor and movement error and large numbers of agents. It only requires agents to have local communication, an approximate measure of relative distance, an approximate measure of relative motion, and a global compass. Though technically the goal of the work was to distribute agents through a shape, not place them in particular points on the plane, the algorithm proceeds by synchronizing every agent’s coordinate system, and so implicitly enables agents to form many arbitrary connected formations of any size.

Our work modifies and extends the ShapeBugs algorithm, reducing the agent abilities required to form arbitrary global formations by eliminating the need for a global compass. By continuously calculating the error in local estimation of motion, orientation can be derived from sequential distance measurements. We present here:

- A distributed method capable of organizing many mobile communicating agents into many classes of arbitrary shapes without agent position or orientation agreement.
- An analysis of formation stability using the method with and without orientation agreement.

Without the need for agents to access some shared coordinate system or orientation (except a small number of initially coordinated seed agents, which are not strictly necessary though helpful for control), our algorithm requires simpler agent hardware at the cost of more complex software and higher data broadcast rates. Maintaining agent density to allow effective orientation measurements requires a new distributed growth process (but no new sensors). Because communication often has measurable attenuation or time of flight [8], giving an approximate measure of distance (though inaccuracy can make this unusable), and motor output can be modeled in software, giving an approximate measure of relative motion, this algorithm potentially enables any mobile communicating agent to globally coordinate using only those two defining abilities.

In addition, calculation errors [9] affected the previous data presented in [3], so new ShapeBugs results are presented for comparison with the modified “ShapeBots” version.

1.1 Previous Work

The difficulty of solving the formation task is related to agent abilities. Agents generally have arbitrary formation ability when they know their positions in

a unified, global coordinate system (each agent can simply move toward the nearest unfilled formation point, for example). These types of agents will not be discussed in this paper. Formations are not as straightforward if agents use distributed logic and local communication, and the literature contains many proposed solutions [4][7][10][11]. Fujibayashi et al [12] describe a method able to create locally regular formations, though without arbitrary controllability, much like simulated crystals. Another approach by Yamaguchi et al [13] allows a variety of formations from mobile agents in a line using similar virtual constraints. Suzuki and Yamashita [2] mathematically describe a process to create regular polygonal formations, while Ikemoto et al [14] extend this ability using Turing waves to polygonal shapes with an axis of symmetry. Hybrid approaches [8][15] are also possible and widespread in WiFi localization literature which assume only a few agents have extra reference capabilities for global positioning. In general, however, no fully distributed arbitrary formation algorithms for agents with no coordinate agreement (i.e. no global position or orientation information) have been demonstrated thus far for communicating mobile agents, though such algorithms have been proven impossible in non-communicating cases [2][1]. This is an important version of the formation problem, because simple, low power agents may not generally have global positioning or orientation information available, such as in energetic [16], heterogeneous [8], or non-geographic environments. Also, any agents which have some form of coordinate agreement would be able to rely on formation algorithms not requiring agreement as a backup in case of failure or as a complement to their existing methods. Our modified algorithm attempts to achieve connected arbitrary formations in a heuristic manner, by synchronizing agents' local coordinate systems using relative distance information.

2 The “ShapeBots” Algorithm

We simulate simple mobile agents in a 2D, $N \times N$ continuous periodic world, chosen to simplify the simulation by ignoring distant agent aggregation. There are many distributed aggregation techniques described for disconnected agents, [17][18] as examples, but we do not explore those here. Within a range $R_B = 6$ units, agents have the ability to broadcast and receive information, sense neighbor distance, and sense their own motion. Agents also have a simulated repulsion range $R_{Rep} = 4$ and physical collision radius $R_c = 1$, though collision handling becomes unimportant if $R_{rep} \gg R_c$. Agents have a reference direction (a “nose”) which allows them to be oriented in the simulation, and while they have the ability to rotate in place and move in any direction no matter which way they are pointed, agent motion vectors are interpreted as relative to this “forward” direction. After every motion, the reference direction is pointed in the direction of motion.

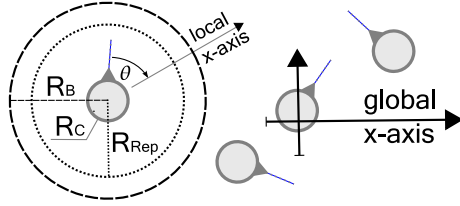


Fig. 1. Sample ShapeBot agent in the 2D plane. The agent’s estimated x-axis at θ from reference direction generally does not correspond to the global observer x-axis.

2.1 Agent Lifecycle and Scheduling

Our algorithm simulates agents as semi-asynchronous, with finite time required for both calculation and movement. Semi-asynchronous is defined here as a variation of asynchronicity as defined by [1], but with a finite upper bound instead of a lower bound on how long an agent may stay in each state of its activation cycle, which we believe more closely models robot hardware. Agents follow a lifecycle of:

$$Wait \rightarrow Sense_1 \rightarrow Compute_1 \rightarrow Move \rightarrow Sense_2 \rightarrow Compute_2 \rightarrow Wait \quad (1)$$

The final $Sense_2$ and $Compute_2$ steps are necessary because agents compare information before and after *Moving* to determine orientation from positioning error. The *Wait*, $Compute_{1,2}$, and *Move* steps each take time bounded by the finite values W_t , C_t , M_t respectively, while the *Sense* time is amortized into the total *Compute* time. After each agent transition, the time until the next transition is drawn uniformly from $[0, T_{max})$, where T_{max} is appropriately W_t , C_t , or M_t . Agents may move $d_m = 1$ or 0 units during each $Move \rightarrow Sense_2$ transition (if they move 0 units, the *Move* step takes no time). In a more realistic system agents could move varying distances over time, but this was not modeled exactly in our simulation. The positioning and orientation algorithms depend largely on the ability of agents to estimate the magnitude of their motions, while the magnitude itself could vary. One agent iteration is defined complete when an agent transitions through all six steps. In our results, $M_t = 500$, $W_t = M_t$, and $C_t = 0.01M_t$.

The simulation begins with each agent in the *Wait* state, with each agent’s time to *Wait* randomly drawn from $[0, W_t)$. The agents are placed in a priority queue, ordered by soonest update time. The simulation proceeds by removing the agent with the soonest transition from the queue, performing the transition, updating the agent time for the next transition, and finally adding it back into the queue. One simulation iteration is defined complete when a simulation of N agents makes N agent transitions back to the *Wait* state (not all agents may be iterated fully, some may be iterated multiple times). Every agent continuously broadcasts the following information (for both their previous and current iteration):

- estimated (x, y) position and local orientation θ
- computational state (described below)
- local neighbor density ρ
- count of position and orientation updates since oriented

This information is therefore always available for other local agents to *Sense*. While moving, the position is extrapolated from the original position and time, but new estimates are only generated in *Compute* steps. Broadcasted information also contains the relative distance between the agents at time of broadcast.

In addition to the readiness states described above, agents have three computational states, as described in [3], of *Lost*, *OutOfShape*, and *InShape*. An agent in the *Lost* state has no coordinate system, while the *OutOfShape* and *InShape* agents have local coordinate systems (an (x, y) coordinate and an orientation estimate θ). An agent in the *Lost* state will wander randomly through the world until it senses three neighbors with coordinates, which then allows the *Lost* agent to trilaterate a position guess using the neighbor distance measurements and (x, y) positions. Importantly, while trilateration can approximate a consistent coordinate system between neighboring agents, the relative orientation of the agents' reference direction with respect to this coordinate system θ must be calculated using distance measurements after neighbor motions. As an initial orientation guess, a newly un-*Lost* agent's orientation is reset to correspond to a motion directly toward the centroid of the orienting points, though the actual motion may have been less direct.

A newly oriented agent will become *Lost* again immediately if it cannot remain in contact with other oriented agents long enough for several agent orientations to occur, defined in our simulation as a 10 iteration update window for orientation and trilateration. Otherwise poorly oriented agents tend to immediately escape the main formation and form nearby competing formations of opposite orientation. As in [3], once agents have acquired a consensus coordinate system they attempt to fill a formation shape by each agent simulating a virtual gas particle with a repulsion range R_{rep} . Neighboring agents inside the shape with distance $< R_{rep}$ will repel one another, leading to an average distribution of agents throughout the shape. These virtual gas mechanics allow the shapes to be robust against agent addition and death while spreading agents evenly throughout the formation.

2.2 Orienting using Agent Motion

To adjust an agent's perceived reference direction with respect to the trilaterated consensus coordinate system θ , an agent simply observes the error in its coordinate system after motion. Because agents' reference direction is pointed in the last direction of motion, the corrected direction of motion with respect to the trilaterated coordinate system is an estimation of the agent's reference direction in that consensus coordinate system. Along with the simulated error in movement and distance measurement, asynchronous motion limits the agent speed and density required for the agent coordinate systems to converge. Agents may

move simultaneously, adding inaccuracy to their trilaterated coordinate systems due to estimated orientation errors, and the less accurate coordinates can then result in less accurate orientation. Like position updates, the noisy orientation changes calculated each agent motion are averaged over a window of 10 agent iterations to smooth errors.

2.3 Formation Scaling

A major consequence of the co-dependency of orientation and position is that agent coordinate systems will diverge very quickly from one another if agents become separated from a dense group with coordinate agreement. For this reason, agents with coordinate systems try to maintain a target agent density of $\rho = 18$ neighbors by scaling their local copy of the target formation, much like a balloon expands as more gas is added. If many agents are added to the shape, the shape will expand proportionally to maintain ρ , while if many agents are removed or become inoperative, the target shape will shrink to ensure agents stay grouped. If a particular size formation is required, the shape scale could easily be capped at the target size (this is not done in our tests), but if there are too few agents to maintain the target density the agents will be unable to maintain the formation size.

The distributed scaling is achieved by every agent averaging neighbor values of the desired shape scale. An agent broadcasts its desired shape scale (calculated assuming local neighbor density applied throughout the shape) while using itself the average of neighbor scales. There can be a large amount of variability in the number of neighbors each iteration, so the broadcasted shape scale is smoothed using density information from the past 10 agent iterations.

3 Formation Control Results

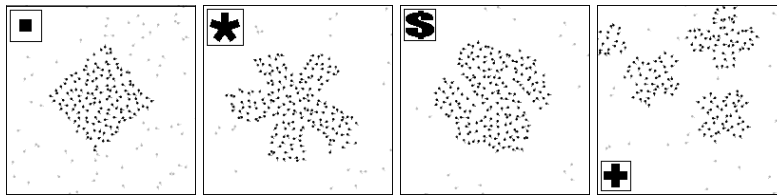


Fig. 2. Sample formations. Dark agents are non-*Lost*, and the formation shapes are specified by the bitmaps in the image corner. Only connected formations are possible using our algorithm, and shapes containing thin portions tend to be harder (or sometimes impossible) to form well. As seen in the **+** formations, if *Lost* agents are allowed to trilaterate with other *Lost* neighbors as well as non-*Lost*, they will spontaneously create many stable formations without seed agents.

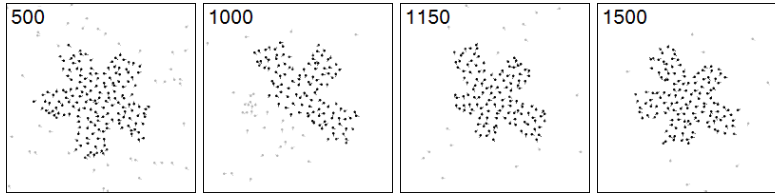


Fig. 3. Even without compasses, formations remain resistant against agent death. At iteration 1000, agents in the lower half of the asterisk formation die. At iteration 1150, new agents have moved into the destroyed area, and by iteration 1500 the formation is rebuilt (at smaller size to maintain density).

To measure the ability of agents to achieve coordinate agreement under varying amounts of distance and motion error, the coordinate variance of 200 agents was calculated (Figure 4) while forming a simple square formation in an 80×80 world. Coordinate variance is defined as the average variance in distance between the non-*Lost* agent origins, using the average orientation of all non-*Lost* agents. Higher variance corresponds to weaker formation control. 12 seed agents were initially placed in a 25×25 square in the center of the world and given consistent coordinates and orientation to start the formation. The distance error e is applied to measurements d by adding a uniformly chosen value from the range $(-de, de)$. Motion error is applied in the same way to each component of a motion vector (m_x, m_y) .

4 Discussion

What a global compass seems to "buy" you, in the context of distributed, arbitrary shape formation algorithms, is the ability for agents to tolerate greater distance and motion sensor error. As seen in Figure 4, 60% distance error with our compass-free agents results in the same coordinate variance as 100% distance error using agents with compasses and scaling. Though less converged, our agent formations without orientation still remain stable until about 80% distance error (when the converged formations may only be temporarily stable). Our modified algorithm seems less susceptible to motion error than the original ShapeBugs work, even in the no-compass case, however this is probably due to a reduced agent speed of 1 unit/iteration instead of the original 2 units/iteration and higher density. Formation instability in our modified algorithm tends to increase slowly until a threshold error level is passed, then quickly jumps, as can be seen in Figure 4. The much higher instability then makes it impossible to maintain a consensus coordinate system, in the case of no compasses.

Our corrected ShapeBugs convergence results match well with the originally reported values, however for low distance error the coordinate variance is reduced. As the problem related to inaccurate initial trilateration calculation, the effect becomes minimal where measurement errors are larger .

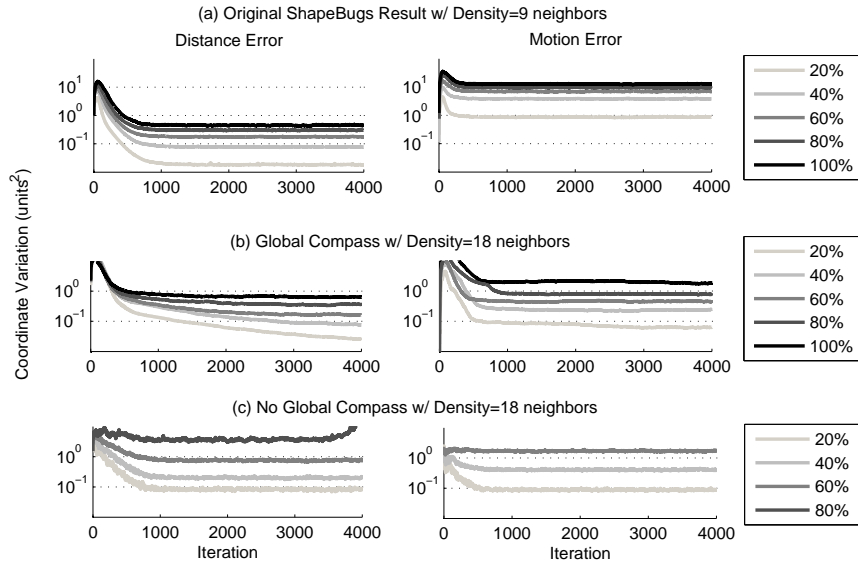


Fig. 4. Average coordinate variance under distance and motion error (averaged over 90 simulations). The variance using the original ShapeBugs algorithm from [3] is shown in (a). Though variance is low, as distance error increases, often seeded simulations will fail to converge on a target shape, this happens 6%-80% of the time for 20%-100% distance error and 11%-45% of the time for 20%-100% motion error. In (b) our modified algorithm results using formation scaling are presented. At maximum, the simulations fail to converge only 13% of the time, this is largely due to formations growing too quickly (but is unchanged to better compare with (c)). Simulation data using formation scaling without compasses is presented as (c), all simulations achieve convergence except at distance error 80% (formations may not be stable long-term). At higher error, no simulations converge.

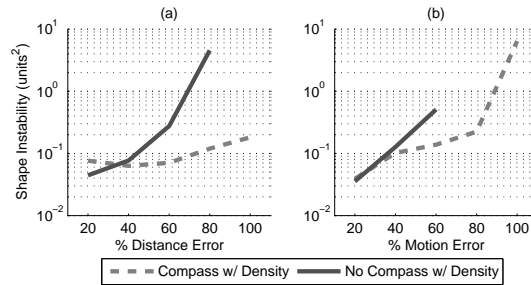


Fig. 5. Formation instability, measured as the average standard deviation over time of the stabilized coordinate variance from $t = [2000, 4000]$, is lowest when distance and motion error is low.

4.1 Conclusion

The local communication, asynchronous update, and short burst motion schedule make the ShapeBots algorithm amenable to many different types of agents, though higher density and lower speed (equivalent to more frequent updates) to mitigate larger measurement errors are required to reach comparable coordinate convergence. For limited-power devices, this may mean the speed of formation is lowered as compared to the ShapeBugs algorithm. Alternately, devices could use higher transmission power to increase the agent range, and so increase the formation density. Stable formations tend to converge more consistently using gradual scaling, this may be desirable despite the other limitations.

In a situation where agents can extract reasonably accurate distance and motion data from communication and motion input and can tolerate small variance in the global formation, large numbers of mobile agents containing only communication hardware can organize in arbitrary patterns using the ShapeBots algorithm. The new algorithm shares much of the resilience of the original ShapeBugs approach, as the repeated coordinate averaging and virtual gas motion was retained from the original algorithm. The presented method of synchronizing scale across the formation to maintain this density works even with large numbers of agents, but provides an additional source of formation shape error (though *not* coordinate error). If the target shapes have very thin sections, where agents are prone to escape and density measurements become especially variable (because there cannot be many surrounding positions), even rescaling may fail and the formation can be distorted or leak agents.

In future work we hope to address these issues and extend the algorithm to approximate more intricate shapes. A major limiting factor is the rapid divergence of coordinates and orientation once agents become disconnected. Scaling agent motion proportional to their confidence in position in the formation may be one useful approach, as slower motions could allow converged agents to stabilize the coordinate system more exactly. Over time the agent formation would also tend to slow down and stop, ideal behavior for energy-limited agents. With larger numbers of agents, though, intricate shapes become less of a problem because the shape scale grows and narrow portions become wider. Another approach would simply be to grow the shape a filled square, then transitioning to the actual shape when agents agree the scale is large enough.

The ShapeBots simulation code, implemented in the MASON multi-agent simulation framework [19], is available online at:
<http://www.informatics.sussex.ac.uk/users/gms21/shapebot>

4.2 Acknowledgments

We thank Simon McGregor for his comments on the work-in-progress.

References

1. Principe, G., Santoro, N.: Distributed algorithms for autonomous mobile robots. In: Proc. of 5th IFIP Intl. Conf. on Theoretical Computer Science. (2006)

2. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing* **28**(4) (1999) 1347–1363
3. Cheng, J., Cheng, W., Nagpal, R.: Robust and self-repairing formation control for swarms of mobile agents. In: Proc. of the 20th Natl. Conf. on Artificial Intelligence, Menlo Park, California, AAAI Press (2005) 59–64
4. Dudek, G., Jenkin, M., Milios, E., Wilkes, D.: A taxonomy for multi-agent robotics. *Autonomous Robots* **3** (1996) 375–397
5. Sahin, E., Labella, T., Trianni, V., Deneubourg, J.L., Rasse, P., Floreano, D., Gambardella, L., Mondada, F., Nolfi, S., Dorigo, M.: SWARM-BOTS: Pattern formation in a swarm of self-assembling mobile robots. In: Proc. of the IEEE Intl. Conf. on Systems, Man and Cybernetics, Hammamet, Tunisia, IEEE Press (2002)
6. Cao, Y.U., Fukunaga, A.S., Kahng, A.B.: Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots* **4**(1) (1997) 7–23
7. Bahceci, E., Soysal, O., Sahin, E.: A review: Pattern formation and adaptation in multi-robot systems. Technical Report CMU-RI-TR-03-43, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA (2003)
8. Niculescu, D., Nath, B.: Position and orientation in ad hoc networks. *Ad Hoc Networks* **2** (2004) 133–151
9. Nagpal, R. Personal communication (2007)
10. Defago, X., Konagaya, A.: Circle formation for oblivious anonymous mobile robots with no common sense of orientation. In: Proc. of the 2nd ACM Intl. Workshop on Principles of Mobile Computing, New York, NY, USA, ACM Press (2002) 97–104
11. Howard, A., Mataric, M., Sukhatme, G.: Putting the 'i' in team: an ego-centric approach to cooperative localization. In: Proc. of the IEEE Intl. Conf. of Robotics and Automation, Taipei, Taiwan (2003)
12. Fujibayashi, K., Murata, S., Sugawara, K., Yamamura, M.: Self-organizing formation algorithm for active elements. In: Proc. of the 21st IEEE Symposium on Reliable Distributed Systems, Washington, DC, USA, IEEE Computer Society (2002) 416
13. Yamaguchi, H., Arai, T., Beni, G.: A distributed control scheme for multiple robotic vehicles to make group formations. *Robotics and Autonomous Systems* **36**(4) (2001) 125–147
14. Ikemoto, Y., Hasegawa, Y., Fukuda, T., Matsuda, K.: Gradual spatial pattern formation of homogeneous robot group. *Information Sciences - Informatics and Computer Science* **171**(4) (2005) 431–445
15. Mihaylova, L., Angelova, D., Canagarajah, C., Bull, D.: Algorithms for mobile nodes self-localisation in wireless ad hoc networks. In: 9th Intl. Conf. on Information Fusion, Florence, Italy (2006)
16. Souissi, S., Defago, X., Yamashita, M.: Using eventually consistent compasses to gather oblivious mobile robots with limited visibility. In: Proc. of the 8th Intl. Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS). (2006) 471 – 487
17. Trianni, V., Gross, R., Labella, T., Sahin, E., Rasse, P., Deneubourg, J., Dorigo, M.: Evolving aggregation behaviors in a swarm of robots. Technical Report TR/IRIDIA/2003-07, IRIDIA, Universite Libre de Bruxelles, Bruxelles, Belgium (2003)
18. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science* **337**(1-3) (2005) 147–168
19. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K.: Mason: A new multi-agent simulation toolkit. In: Proc. of the 2004 SwarmFest Workshop. (2004)