Segregation without Computation

Jordan Burklund
Worcester Polytechnic Institute
Worcester, MA
jsburklund@wpi.edu

Michael Giancola
Worcester Polytechnic Institute
Worcester, MA
mjgiancola@wpi.edu

Peter Mitrano
Worcester Polytechnic Institute
Worcester, MA
pdmitrano@wpi.edu

***Abstract*—In this paper, we demonstrate that memoryless computeless robots are capable of n-class segregation. We evolve controllers for this behavior with a genetic algorithm and perform a grid search over all parameters to understand the full parameter space. We find that robust segregation is possible, although not guaranteed. Instead, we prove that aggregation of kin robots is guaranteed when some reasonable conditions are met.**

***Index Terms*—swarm robotics, robot aggregation, robot segregation**

I. Introduction and Related Work

Many prior methods have demonstrated the ability to aggregate robots and objects in a distributed way, and some have shown the ability to perform such tasks with limited computation and sensing capabilities. Decentralized aggregation is frequently posed as a precursor to other swarm behaviors such as sorting, self-assembly, or coordinated motion [1] [2]. Segregation is a natural extension of aggregation, and so we also desire algorithms for segregating robots into different clusters. In order to make this robust to individual failures and simple to implement, we would also like to achieve segregation with no communication between robots, no complex sensors, no leaders or beacons, and no environmental cues.

In nature, we find many examples of aggregation and segregation. Segregation occurs in cells during embryogenesis, and segregation of objects can be found in ants who sort their brood [3]. Furthermore, segregation is something humans do very quickly, and we can do so almost on reaction rather than carefully planned motion. These examples motivate the study of simple controllers for segregation.

A. Aggregation and Segregation

Robot aggregation is defined as having all robots in the swarm collect at a particular location in a distributed manner. Many swarm aggregation policies require robots to compute bearing and distance to other robots, to sense gradients in the environment, or to otherwise communicate information. However, implementing these communication systems is difficult in practice, so methods that do not require communication or complex sensing are desirable. In [4], the authors propose a new class of policies that require no computation.

The work of [5] does not restrict itself to a computeless or memoryless solution. Instead they develop an aggregation algorithm which enables robots to respond to guidance commands – i.e., a human using hand gestures to indicate which direction the swarm should move. They show that swarms running their algorithm are able to follow guidance commands and stay aggregated without colliding. A simpler controller is used in [6], where robots are equipped with IR distance sensors surrounding the robot, 4 microphones, and an omnidirectional speaker. The policy is a linear combination of the IR sensor values and the intensity values of the microphones. The authors show that a genetic algorithm is able to find weights for the policy such that robots aggregate together. In [7], the authors present a robot aggregation model with very limited sensing and no memory, but allows the robots to perform computations to determine their next position. They prove that their algorithm is correct in theory, then run the algorithm in simulation.

Finally, [4] considers a binary sensor that maps directly to wheel velocities. The authors demonstrate extremely robust aggregation despite these limitations, and they also prove formally that aggregation is guaranteed and find theoretical bounds on aggregation time in simple situations. Similarly, the authors of [8] perform object aggregation with the same restrictions on sensing and control.

Segregation of robots has received little attention in swarm robotics, however many researchers have focused on sorting of objects [9] [10] [11] [12]. The study in [3] shares our objective, however they assume that each class has the same number of robots and that each robot knows the position of all other robots. In contrast, we adhere to the memoryless computeless controller architecture with a simple ternary sensor. [13] also explores segregation robots based on local interactions inspired by gravity, however they require a centralized broadcast or a consensus algorithm to agree on the source or direction of gravity.

B. Robots That Do Not Compute

The memoryless and computeless controllers originally proposed by [4] has been extended to other tasks and been modified in many ways by other researchers.

In [14], the authors propose an aggregation policy roughly based on observations of bees clustering in an optimal temperature location. In their method, robots are only able to distinguish between collisions between another robot or a wall, can only sense the intensity of a light source when they have collided, and do not communicate information. Although the robots have limited sensing and communication capabilities, the authors show that robots can aggregate to an optimal light

Algorithm 1 Controller Design

```
if $I = 0$ then
    set wheel speeds to $v_{l_0}$, $v_{r_0}$
else if $I = 1$ then
    set wheel speeds to $v_{l_1}$, $v_{r_1}$
else
    set wheel speeds to $v_{l_2}$, $v_{r_2}$
end if
```

intensity location, and that the time to converge to the optimal location improves with increasing number of robots in the environment.

Robots that do not compute and are memoryless have also been shown to aggregate around a specific object, circle in a ring around an object, and forage for obstacles [15]. In this work, the authors show one can construct simple cost functions to guide the evolution of controllers to do new, yet interesting tasks. However, they also report that in attempting to evolve a controller to rendezvous the robots around an object, they accidentally and consistently evolved a policy where the robots circled around the target object. This demonstrates that designing a cost function can be hard or unreliable. They achieved rendezvous by initializing the policy at generation zero to the policy found in [4] for simple aggregation.

II. Methodology

A. Problem Formulation

We consider a collection of differential drive robots all executing the same controller in a homogeneous, two dimensional, obstacle free environment. The robots are equipped with a single forward facing line of sight sensor which can distinguish between the presence of a kin robot, a non-kin robot, and nothing. This sensor is assumed to have infinite length (we consider non-infinite length in Section V-E). We assign $I = 0$ to the state when no robot is seen, $I = 1$ to the detection of a kin robot, and $I = 2$ to the detection of a non-kin robot. We allow for any number of classes, but the robots need not distinguish between different non-kin classes. They need only to detect if a robot is of the same class or not. The objective is to segregate robots into clusters, ideally such that all the robots of the same class are packed into one cluster with no non-kin robots.

The controllers we use in our experiments are of the same form as in [4]. They simply map each of the three sensor values to a set of wheel speeds. Pseudo-code for the controller is shown in Algorithm 1. This controller has six parameters:

$$[\, v_{l_0}, v_{r_0}, v_{l_1}, v_{r_1}, v_{l_2}, v_{r_2} \,]$$

Keeping with the form used in [4], we let these parameters range from -1 to 1. In simulation, we then scale this parameter to range from $-20\,\mathrm{cm\,s^{-1}}$ to $20\,\mathrm{cm\,s^{-1}}$.

B. Simulation Environment

We use the ARGoS simulation environment to search for controller parameters and evaluate them, which has the advantage of allowing us to run trials much faster than on real robots [16]. In our simulations, we use a range and bearing sensor to implement the theoretical line of sight sensor. In our analysis, we describe the sensor as infinitely thin, but in practice it must have some small finite angle. We explore the performance effect of various beam angles in Section V-D. We note here the detail that we consider robots to be connected in a cluster if the gap between them is $5\,\mathrm{cm}$ or less. We now describe two approaches for finding these parameters, genetic algorithms and grid search.

C. Evolving Segregation

In order to quickly search for parameters to achieve segregation, we use a simple genetic algorithm to evolve controller parameters. The genetic algorithm we used is unmodified from the example MPGA code provided with the ARGoS simulator. The mutation strategy is simply to mutate each of these 6 parameters with some probability p (0.05 in our experiments). If a parameter is selected to be mutated, a uniformly random number from -1 to 1 is picked for the new value. Selection is performed by picking the two lowest cost individuals, and the next population is formed by crossing the parameters of these two individuals. The original two parents are always kept in the population so that the current lowest cost individual is never removed until a lower cost genome is discovered.

D. Cost Functions

As with any optimization algorithm, it is important to have a cost function that accurately assigns cost to behaviors. We explore two different cost functions, and we find that one of them did not behave as we expected (see Section V-A). The first is the cluster metric used by [4], which is the proportion of the number of robots in the largest cluster to the total number of robots. We will apply this for each class of robots and sum up the cluster-metric cost for each class to get the total cost. The negative sign is present because the cluster metric is 0 when no robots are connected and 1 when all robots in a class are connected, so the negative sign assigns more connections a lower cost. This was ultimately the cost function we used in grid search and all following experiments.

$$c_{\text{total}} = \frac{1}{n} \sum_{\text{classes}}^{n} \sum_{t=0}^{T-1} -t c_{\text{gauci}}^{(t)}$$

We also tried a new cost function to address some concerns with the cluster metric. The cluster metric described above is problematic because it considers a straight line of robots to be one cluster. In some scenarios, we care about how tightly the swarm is packed in its clusters. This new metric uses the centroids of the robots in each class. We borrow the dispersion metric $u^{(t)}$ from [4], which essentially computes the sum of distances from each robot to the centroid, but in our case we apply this to each class of robots. Consider $u_i^{(t)}$ to be the dispersion of a particular class i at time t. Given all the centroids $\bar{p}_i^{(t)}$ for each class i, we call the centroid of all of these centroids $\bar{P}^{(t)}$. We call this the centroid-of-centroids cost function, and it can be defined formally as follows:

$$c_{\text{intra}} = \sum_{i=1}^{n} u_i^{(t)}$$

$$c_{\text{inter}} = -\frac{1}{4r^2} \sum_{i=1}^{n} \|\bar{p}_i^{(t)} - \bar{P}_i^{(t)}\|^2$$

$$c_{\text{total}} = \sum_{t=0}^{T-1} t(c_{\text{intra}} + c_{\text{inter}})$$

E. Grid Search

In order to exhaustively search the space of possible controllers, we conduct a grid search of the 6-dimensional parameter space. Due to limited computational resources, we were only able to search with a resolution of 7 values per parameter, which means in total we evaluated $7^6 = 117649$ parameters. For each parameter, we ran 1 trial in 36 different initial configurations, with 180 seconds for each simulation. These configurations consisted of uniformly random placement, clusters, and lines of robots distributed throughout the environment. We chose to include some structure configurations (clusters and lines) because we discovered that they varied significantly in performance from uniform random configurations, and by explicitly evaluating on structure configurations we can speak more confidently about the ability of the controller to succeed in any configuration. Otherwise, we would need to have far more trials with uniformly random robot placement to make the same claims. We were able to parallelize across the 36 configurations and run at 75x real time, so it took approximately 4 days to evaluate all the controllers.

Because the search space is 6 dimensional, we choose to visualize it by plotting every pair of parameters against each other. For example, we consider how the cost changes as v_{l_0} and v_{r_0} change. We note that these graphs, shown in Figure 1, show very similar patterns to the same plots presented by [4]. As an example of reading these plots, we can tell from the plot of parameters 2 and 3 ($I = 1$) that there were no good controllers where the left and right wheel speeds were equal and negative (dark squares in the upper left), and that the best controllers had slightly unequal values close to 1 (lightest squares in the bottom left). These plots also visualize how there are some sharp discontinuities where performance changes dramatically.

III. The Emergent Behavior

After running 10 generations with 10 genomes per generation, the lowest cost controller we found was [0.5, -0.5, -0.2, 0.6, 0.6, 0.2]. Similarly, after running the grid search the best controller was [1, -2/3, 1/3, 1, 1, 0]. For both of these

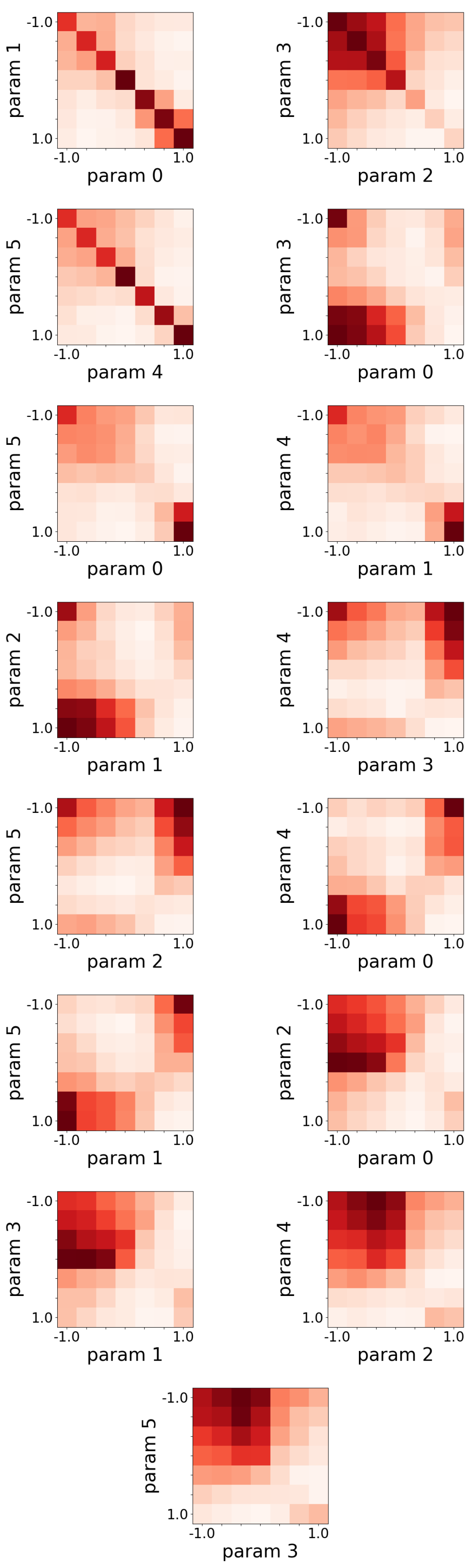

Fig. 1. Each grid cell shows the cost of the best controller with the x-axis and y-axis parameters at that given cell value.

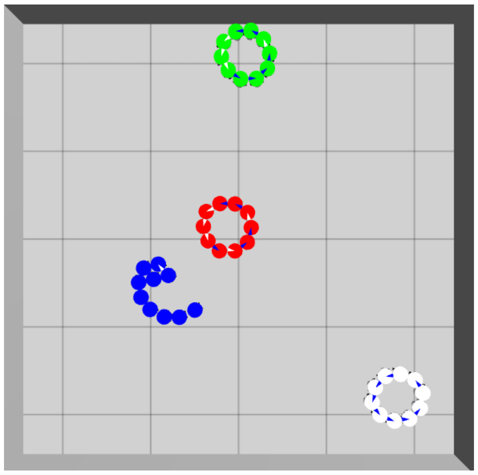

Fig. 2. the emergent behavior is rings of kin. These rings sometimes expand or rotate around the environment.

controllers, the robots policy is to turn away from kin but turn the opposite way when they see nothing or non-kin. This cause the kin robots to zig-zag in a line towards their kin, and when multiple robots execute this behavior the kin robots form rings. An example of this can be seen below in Figure 2. Rarely, they also for these shrimp-like shapes which can also be seen in Figure 2.

The basic description "they form rings" does not do justice to the complexity of the emergent behavior. We strongly recommend you view the supplementary videos in Section C to appreciate it fully. For example, we notice that these rings expand over time, and that if they are disturbed by a non-kin robot trying to pass through, they break and reform in a fascinating snake-like manner.

IV. Controller Analysis

Now that we have found parameters that achieve what we consider segregation, we attempt to prove that aggregation between kin is guaranteed. In all of these proofs we will consider the controller found via grid search since it is the best controller.

A. Aggregation with Static Kin

We found from grid search that the parameters [1, -0.6667, 0.3333, 1.0, 1.0, 0.0] best achieve segregation as defined by our cluster metric cost function. We now analyze this controller formally. First, we prove that a single isolated robot executing this controller will aggregate to a static kin robot, or more importantly, a static cluster of kin robots. This proof is a variation of the Theorem 5.1 provided in [4]. Consider a robot situated as shown in Figure 3. The controller dictates that since a kin robot is seen, $I = 1$, so $v_{l_1} = 0.3333$ and $v_{r_1} = 1$. This will cause the robot to turn counter-clockwise with some radius R. Without loss of generality, we define our coordinate system so that $c_i = [0, 0]$ with the robot i facing the $+x$ axis. For this analysis, we assume that the robot has a fixed, positive inter wheel distance W, and that the controller is executed in

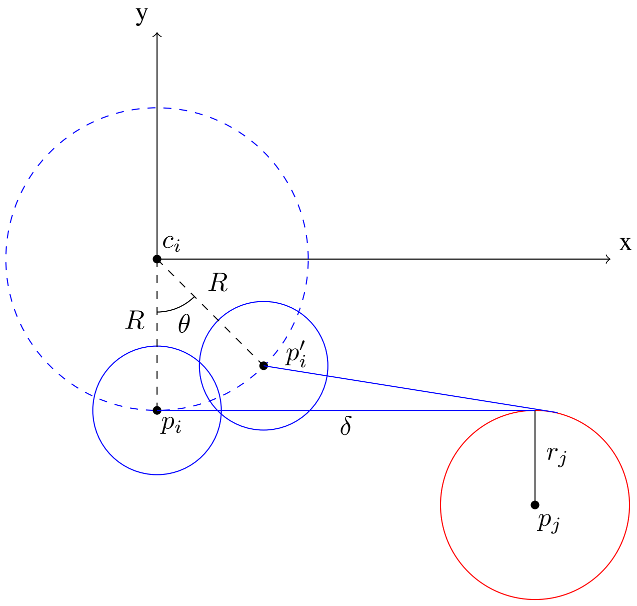

Fig. 3. The robot will always move closer to its Kin.

finite time steps. Our goal is to show that the distance between the position of the robot after executing our controller for one time step p_i' and the kin robot p_j is less than the distance between the initial position of the robot p_i and the kin robot p_j. Formally, this is expressed by the following relationship:

$$\|p_j - p_i'\| < \|p_j - p_i\| \tag{1}$$

We emphasize that r_j could be the radius of a single kin robot or the radius of a large cluster of kin robots. With this coordinate system defined and assumptions stated, we can define the coordinates of p_i, p_j, and p_i'.

$$\begin{aligned} p_i &= \begin{bmatrix} 0 \\ -R \end{bmatrix} \\ p_j &= \begin{bmatrix} \delta \\ -(R + r_j) \end{bmatrix} \\ p_i' &= \begin{bmatrix} R\sin(\theta) \\ -R\cos(\theta) \end{bmatrix} \end{aligned} \tag{2}$$

We then substitute these variables into our inequality (Equation 1), and the result is shown in Equation 3. For the full derivation, see Appendix A. In short, the distance of the robot to its kin is guaranteed to decrease until a point where the sensor ray distance δ is not greater than some simple function of θ and R.

$$\delta > (R + r_j)\tan\left(\frac{\theta}{2}\right) \tag{3}$$

We can further calculate exactly how much closer the robot i will be to robot j after executing the $I = 1$ state for one time step. As we show in Appendix B, the change in distance between the robots is exactly equal to Equation 4.

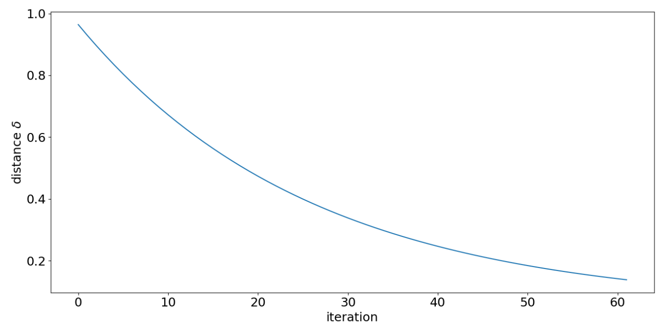

Fig. 4. The distance between a robot and its kin after each step as it aggregates. We use $W = 0.14$, $r_j = 0.14$, $\Delta t = 0.1$. In this example, aggregation took 60 iterations.

We can use this equation to plot how the distance between the robots changes over time. We show an example of this for our simulated robot in Figure 4.

$$2R\big((r_j + R)(1 - \cos(\theta)) - \delta \sin(\theta)\big) \tag{4}$$

In both of these equations the dependent variables θ and R are themselves a function of Δt, the inter wheel diameter W, and the wheels speeds of the controller v_{l_1} and v_{r_1}, which are shown in Equation 5.

$$\begin{aligned} \theta &= \Delta t \omega = \Delta t \frac{v_{r_1} - v_{l_1}}{W} = \Delta t \frac{1 - (0.3333)}{W} = \frac{2\Delta t}{3W} \\ R &= \frac{W}{2}\left(\frac{v_{r_1} + v_{l_1}}{v_{r_1} - v_{l_1}}\right) = \frac{W}{2}\left(\frac{1 + (0.3333)}{1 - (0.3333)}\right) = \frac{W}{4} \end{aligned} \tag{5}$$

Combining 5 with 3 gives us 6, which tells us the conditions under which aggregation is guaranteed. We emphasize that this does not mean aggregation to a kin is guaranteed for all values of Δt, r_j, and W. For example, if Δt is increased to 1 s then aggregation is likely not guaranteed, simply because robot i may have driven so far in its circle that it is further from p_j than before.

$$\delta > \left(\frac{W}{4} + r_j\right) \tan\left(\frac{\Delta t}{3W}\right) \tag{6}$$

We can now consider an example with the FootBot robot. The inner wheel distance for the FootBot is $W = 0.14$m, and in our simulations we use a time step of $\Delta t = 0.1s$. Therefore $\theta = 0.4761$ rad and $R = 0.035$, and so aggregation is guaranteed when the following is true.

$$\begin{aligned} &\delta > \Big(0.035 + r_j\Big) \tan \frac{0.4761}{2} \\ &\delta > 0.2427 r_j + 0.0085 \end{aligned} \tag{7}$$

What we would like is for the right hand side of the equation to always be less than the distance at which the robots are considered aggregated. In other words, we want aggregation to be guaranteed until the sensor ray distance is small enough that aggregation has been achieved. We defined aggregation as when the distance between the robots $d \leq r_j + r_i$. Since the sensor ray length δ is always less than that distance, we can simply show that the right hand side of Equation 6 is always less than $r_i + r_j$. Again we must consider an example since aggregation cannot be guaranteed in all scenarios, so going back to our example with the FootBot, we note that $0.2427 r_j + 0.0085 < r_j + r_i$ is always true as long as $r_j > 0$ and $r_i \geq 0.0085$.

We examined a wide range of possible values for our parameters Δt, r_i, W, and r_j, and we find that as long as Δt is small (less than 0.25s), then aggregation is guaranteed for reasonable values of W (less than 0.5m).

V. Experimental Results

A. Evaluating the centroid-of-centroids cost function

When we implemented the centroid-of-centroids style cost function, we quickly found examples of configurations ranked in ways we did not like. One example is shown below in Figure 5. The behavior that looks like aggregation was given lower cost of -8e9, versus the behavior that looks like segregation was given a cost of -5e9.

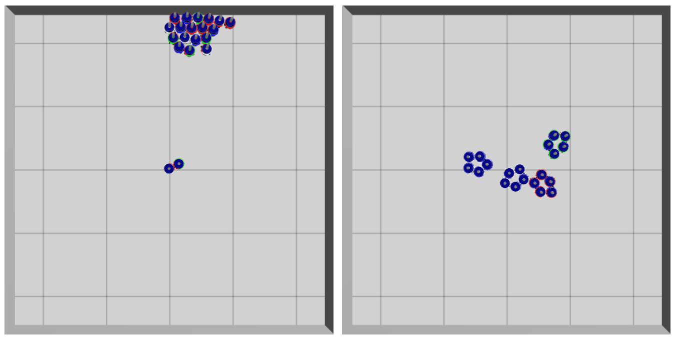

Fig. 5. The left picture was given lower cost than the right, which is not desirable.

B. Scalability Study

In this experiment we investigate how segregation behavior scales with the number of classes and number of robots in the environment. We varied the number of classes from 1 to 25 and ran 100 trials of robots randomly distributed. Because our cost function is independent of the number of classes, we first considered having 10 robots for each class. However, this also means that in the trial with 25 classes there are 25 times the total number of robots than in the 1 class trial. This means that occlusion of robots is more likely so we are not surprised to find that cost increases with more classes. The results of this are plotted in Figure 7. Another scenario is to consider a fixed number of robots and split them into more and more classes. We choose 100 robots because we are still able to get 4 robots per class at 25 classes. As you can see in Figure 6, the cost no longer increases as the number of classes increases. However, the cost for just a few classes is much higher. This is expected, because when there are many robots of a single class, our controller forms very large sparse rings where robots are too far to be considered clustered.

Ultimately, we can say that our controller scales well to many classes with few robots, but not well to many robots of the same class.

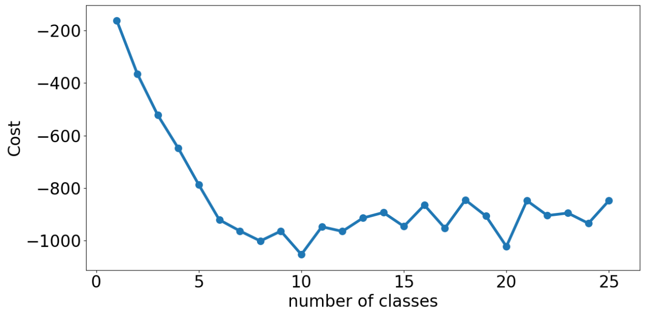

Fig. 6. The average cost with 100 robots divided into N classes. More classes are lower cost partially because kin robots stay close enough to be considered clusters.

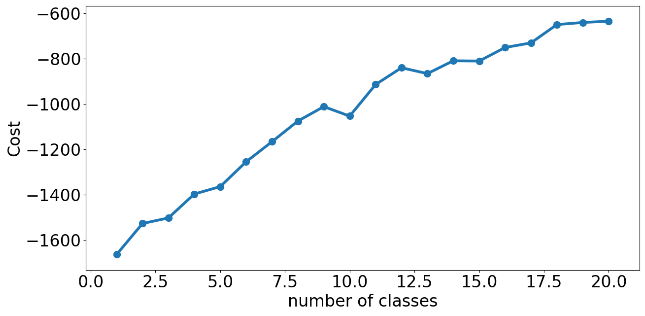

Fig. 7. The average cost with N classes, 10 robots per class. More classes are higher cost because other robots obstruct your view making it difficult to find kin.

C. The Effect of Implementation Details of the sensor

So far we have been vague on how exactly we implement this theoretical line-of-sight sensor. We found in our experiments that the implementation details have a tremendous effect on the performance or behavior of the controller. This is unfortunate because one of the main motivations of the binary or ternary line of sight sensors is that they can be easily transferred between real and simulated implementations.

Initially, our method for determining sensor state from our simulated range-and-bearing sensors was to consider all the robots within some small angle in front of the robot and pick the closest one. This is very similar to what you would presumably get from a real-world camera implementation that uses colored skirts on each robot and picks the largest blob as the robot to be detected. This sensor implementation works well and was used in all our genetic algorithm and grid search experiments. However, we found later that if instead the robots always prefer to react to kin over non-kin, you can form larger rings more quickly and robustly. For example, if there are two robots within the imagery beam of your sensor and the non-kin robot is closer, you will ignore it and execute the $I = 1$ state which will drive you towards the farther away kin robot. Exploring exactly which of these implementation details have what effect on cost is left for future work.

D. The Effect of Beam Angle

In practice, there must be some finite beam angle to the theoretically line of sight sensor. We ran 100 trials in simulation with uniformly random initial distributions of 40 robots with various half beam angles. Figure 8 shows the results, as well as a diagram showing how we define half beam angle. The best half beam angle we tested was $15°$, and angles smaller or larger got progressively worse. We found that at lower beam angles, it was possible for a robot to become stuck in groups of two or three where the robots spent all their time looking at each other and not peeking around them to find kin. At larger angles, we believe the behavior fails because larger beam angles cause the rings to enlarge faster, which in turn causes the rings to be so large that they are not considered a cluster anymore and so the cost rises.

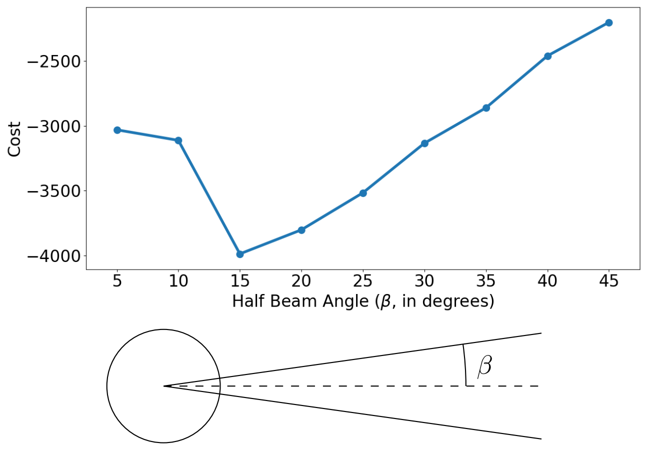

Fig. 8. A $15°$ degree half beam angle is best for segregation. Lower cost is better.

E. The Effect of Beam Length

We also consider what happens if our theoretically infinite sensor now has finite range. We use $15°$ half beam angle and the same experimental setup as with the beam angle experiments. Like in [4], we consider the maximum range of the sensor as the diagonal length of the square in which the robot are initially distributed. In all our experiments, this box was $5\,\mathrm{m}$, so we consider a range of $7.07\,\mathrm{m}$ to be effectively unlimited. We report the costs for beam lengths as a fraction of this maximum range. As you can see in Figure 9, in practice a beam length of only 35% of the theoretical maximum performs just as well. Below this, the performance degrades. However, even a beam length of just 7% of unlimited is more effective than zero length at segregation. In our experiment watching many simulations, beam length should also be scaled with the total number of robots as well as the initial distance between robots. When the beam length is short, there are many cases where two rings of kin robots form at different points in the

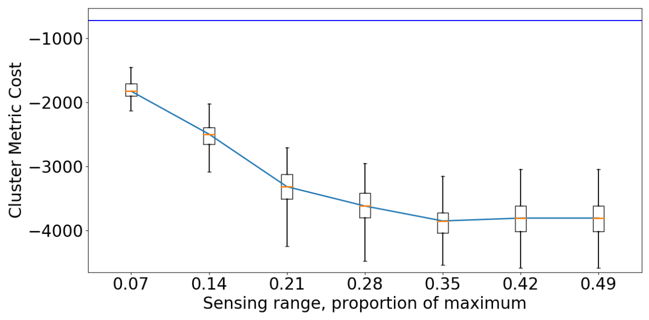

Fig. 9. Segregation is robust to very finite sensor beam lengths. The blue bar indicates the worst (highest) attainable cost in which all robots are isolated from any kin.

environment, and in order for these two rings to join the robots must have sufficiently long beam length in order to detect each other.

VI. Conclusion

In this paper, we demonstrate that memoryless, computeless robots are capable of n-class segregation. We use a simple controller design consisting of a 6-tuple. This controller is invariant to the number of classes, so any given controller can work for any number of classes. To quickly find a quality controller, we evolved one using a basic genetic algorithm, but we also performed a grid search to make claims about the full parameter space. We investigated the effect of sensor implementation details and the number of robots and classes on performance. We find that robust segregation is possible, although not guaranteed. Instead, we prove that aggregation of kin robots is guaranteed when reasonable conditions on controller frequency and inner wheel distance are met.

References

[1] Melvin Gauci, Jianing Chen, Tony J. Dodd, and Roderich Gro\ss. Evolving Aggregation Behaviors in Multi-Robot Systems with Binary Sensors. In *Distributed Autonomous Robotic Systems*, Springer Tracts in Advanced Robotics, pages 355–367. Springer, Berlin, Heidelberg, 2014.

[2] Marco Dorigo, Vito Trianni, Erol ahin, Roderich Gro, Thomas H. Labella, Gianluca Baldassarre, Stefano Nolfi, Jean-Louis Deneubourg, Francesco Mondada, Dario Floreano, and Luca M. Gambardella. Evolving Self-Organizing Behaviors for a Swarm-Bot. *Autonomous Robots*, 17(2-3):223–245, September 2004.

[3] V. G. Santos, L. C. A. Pimenta, and L. Chaimowicz. Segregation of multiple heterogeneous units in a robotic swarm. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1112–1117, May 2014.

[4] Melvin Gauci, Jianing Chen, Wei Li, Tony J. Dodd, and Roderich Gro\ss. Self-organized Aggregation Without Computation. *Int. J. Rob. Res.*, 33(8):1145–1161, July 2014.

[5] A. Gasparri, A. Priolo, and G. Ulivi. A swarm aggregation algorithm for multi-robot systems based on local interaction. In *2012 IEEE International Conference on Control Applications*, pages 1497–1502, October 2012.

[6] E. Bahgeci and E. Sahin. Evolving aggregation behaviors for swarm robotic systems: a systematic case study. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, pages 333–340, June 2005.

[7] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, 15(5):818–828, October 1999.

[8] Melvin Gauci, Jianing Chen, Wei Li, Tony J. Dodd, and Roderich Gross. Clustering Objects with Robots That Do Not Compute. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pages 421–428, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[9] A. Vardy. Accelerated Patch Sorting by a Robotic Swarm. In *2012 Ninth Conference on Computer and Robot Vision*, pages 314–321, May 2012.

[10] Owen Holland and Steve Hoddell. Collective sorting and segregation in robots with minimal sensing, 1998.

[11] Tao Wang and Hong Zhang. Collective Sorting with Multiple Robots. pages 716–720. IEEE, 2004.

[12] Owen Holland and Chris Melhuish. Stigmergy, Self-organization, and Sorting in Collective Robotics. *Artif. Life*, 5(2):173–202, April 1999.

[13] R. Gro, S. Magnenat, and F. Mondada. Segregation in swarms of mobile robots based on the Brazil nut effect. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4349–4356, October 2009.

[14] Serge Kernbach, Ronald Thenius, Olga Kernbach, and Thomas Schmickl. Re-embodiment of Honeybee Aggregation Behavior in an Artificial Micro-Robotic System. *Adaptive Behavior*, 17(3):237–259, June 2009.

[15] Matthew Johnson and Daniel Brown. Evolving and Controlling Perimeter, Rendezvous, and Foraging Behaviors in a Computation-Free Robot Swarm. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (Formerly BIONETICS)*, BICT'15, pages 311–314, ICST, Brussels, Belgium, Belgium, 2016. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[16] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, Mauro Birattari, Luca Maria Gambardella, and Marco Dorigo. ARGoS: a Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems. *Swarm Intelligence*, 6(4):271–295, 2012.

Appendix

A. *Proof of Theorem 1*

$$\begin{aligned}
\|p_j - p_i'\| &< \|p_j - p_i\| \\
\sqrt{(p_{j_x} - p'_{i_x})^2 + (p_{j_y} - p'_{i_y})^2} &< \sqrt{(p_{j_x} - p_{i_x})^2 + (p_{j_y} - p_{i_y})^2} \\
(p_{j_x} - p'_{i_x})^2 + (p_{j_y} - p'_{i_y})^2 &< (p_{j_x} - p_{i_x})^2 + (p_{j_y} - p_{i_y})^2 \\
(\delta - R\sin(\theta))^2 + (-(R + r_j) + R\cos(\theta))^2 &< \delta^2 + (-(R + r_j) + R)^2 \\
(\delta - R\sin(\theta))^2 + (-(R + r_j) + R\cos(\theta))^2 &< \delta^2 + r_j^2 \\
\delta^2 - 2R\delta\sin(\theta) + R^2\sin(\theta)^2 + (r_j + R(1 - \cos(\theta))^2 &< \delta^2 + r_j^2 \\
\delta^2 - 2R\delta\sin(\theta) + R^2\sin(\theta)^2 + r_j^2 + 2Rr_j(1 - \cos(\theta)) + R^2(1 - \cos(\theta))^2 &< \delta^2 + r_j^2 \\
\delta^2 - 2R\delta\sin(\theta) + R^2\sin(\theta)^2 + r_j^2 + 2Rr_j - 2Rr_j\cos(\theta) + R^2(1 - \cos(\theta))^2 &< \delta^2 + r_j^2 \\
\delta^2 - 2R\delta\sin(\theta) + R^2\sin(\theta)^2 + r_j^2 + 2Rr_j - 2Rr_j\cos(\theta) + R^2(1 - 2\cos(\theta) + \cos(\theta)^2) &< \delta^2 + r_j^2 \\
\delta^2 - 2R\delta\sin(\theta) + R^2\sin(\theta)^2 + r_j^2 + 2Rr_j - 2Rr_j\cos(\theta) + R^2 - 2R^2\cos(\theta) + R^2\cos(\theta)^2 &< \delta^2 + r_j^2 \\
\delta^2 - 2R\delta\sin(\theta) + R^2(\sin(\theta)^2 + cos(\theta)^2) + r_j^2 + 2Rr_j - 2Rr_j\cos(\theta) + R^2 - 2R^2\cos(\theta) &< \delta^2 + r_j^2 \\
\cancel{\delta^2} - 2R\delta\sin(\theta) + 2R^2 + \cancel{r_j^2} + 2Rr_j - 2Rr_j\cos(\theta) - 2R^2\cos(\theta) &< \cancel{\delta^2} + \cancel{r_j^2} \\
-2R\delta\sin(\theta) + 2R^2 + 2Rr_j - 2Rr_j\cos(\theta) - 2R^2\cos(\theta) &< 0 \\
-\delta\sin(\theta) + R + r_j - r_j\cos(\theta) - R\cos(\theta) &< 0 \qquad \text{assuming } R > 0 \\
R + r_j - r_j\cos(\theta) - R\cos(\theta) &< \delta\sin(\theta) \\
r_j(1 - \cos(\theta)) + R(1 - \cos(\theta)) &< \delta\sin(\theta) \\
(r_j + R)(1 - \cos(\theta)) &< \delta\sin(\theta) \\
(R + r_j)\left(\frac{1 - \cos(\theta)}{\sin(\theta)}\right) &< \delta \\
(R + r_j)\tan\frac{\theta}{2} &< \delta
\end{aligned}$$

B. *Proof of Theorem 2*

$$\begin{aligned}
\Delta d &= d' - d \\
&= \|p_j - p'_i\| - \|p_j - p_i\| \\
&= \sqrt{(p_{j_x} - p'_{i_x})^2 + (p_{j_y} - p'_{i_y})^2} - \sqrt{(p_{j_x} - p_{i_x})^2 + (p_{j_y} - p_{i_y})^2} \\
&= (p_{j_x} - p'_{i_x})^2 + (p_{j_y} - p'_{i_y})^2 - (p_{j_x} - p_{i_x})^2 + (p_{j_y} - p_{i_y})^2 \\
&= (\delta - R\sin(\theta))^2 + (-(R + r_j) + R\cos(\theta))^2 - \delta^2 + (-(R + r_j) + R)^2 \\
&= (\delta - R\sin(\theta))^2 + (-(R + r_j) + R\cos(\theta))^2 - \delta^2 + r_j^2 \\
&= \delta^2 - 2R\delta\sin(\theta) + R^2\sin(\theta)^2 + (r_j + R(1 - \cos(\theta))^2 - \delta^2 + r_j^2 \\
&= \delta^2 - 2R\delta\sin(\theta) + R^2\sin(\theta)^2 + r_j^2 + 2Rr_j(1 - \cos(\theta)) + R^2(1 - \cos(\theta))^2 - \delta^2 + r_j^2 \\
&= \delta^2 - 2R\delta\sin(\theta) + R^2\sin(\theta)^2 + r_j^2 + 2Rr_j - 2Rr_j\cos(\theta) + R^2(1 - \cos(\theta))^2 - \delta^2 + r_j^2 \\
&= \delta^2 - 2R\delta\sin(\theta) + R^2\sin(\theta)^2 + r_j^2 + 2Rr_j - 2Rr_j\cos(\theta) + R^2(1 - 2\cos(\theta) + \cos(\theta)^2) - \delta^2 + r_j^2 \\
&= \delta^2 - 2R\delta\sin(\theta) + R^2\sin(\theta)^2 + r_j^2 + 2Rr_j - 2Rr_j\cos(\theta) + R^2 - 2R^2\cos(\theta) + R^2\cos(\theta)^2 - \delta^2 + r_j^2 \\
&= \delta^2 - 2R\delta\sin(\theta) + R^2(\sin(\theta)^2 + cos(\theta)^2) + r_j^2 + 2Rr_j - 2Rr_j\cos(\theta) + R^2 - 2R^2\cos(\theta) - \delta^2 + r_j^2 \\
&= \cancel{\delta^2} - 2R\delta\sin(\theta) + 2R^2 + \cancel{r_j^2} + 2Rr_j - 2Rr_j\cos(\theta) - 2R^2\cos(\theta) - \cancel{\delta^2} + \cancel{r_j^2} \\
&= -2R\delta\sin(\theta) + 2R^2 + 2Rr_j - 2Rr_j\cos(\theta) - 2R^2\cos(\theta) \\
&= 2R\big(- \delta\sin(\theta) + R + r_j - r_j\cos(\theta) - R\cos(\theta)\big) \\
&= 2R\big((r_j + R)(1 - \cos(\theta)) - \delta\sin(\theta)\big)
\end{aligned}$$

C. *Supplementary Videos*

This YouTube playlist contains videos of the controller found with grid search:
https://goo.gl/z8UAuB