Segregating Robots That Do Not Compute

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 reproduce robot aggregation behavior from [1] on a swarm of simulated robots. We then extend the controller to perform 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 the robust segregation is possible, although not guaranteed. Instead, we prove only that one or two kin-robots executing our controller will aggregate.**

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

I. Introduction and Related Work

Several of the methods proposed 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 heterogeneous robots for task allocation [2] [3]. By aggregating, agents can cooperate on tasks, share information, and other higher level swarm behaviors. In nature, we find examples of segregation from cells during embryogenesis to ants sorting their brood [4]. We investigate segregating agents into n clusters, so that agents could perform some collective task within their own cluster.

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 [1], 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 omni-directional 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, [1] considers only 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 [4] 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 [1] 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 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

fitness 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 fitness function can be hard or unreliable. They achieved rendezvous by initializing the policy at generation zero to the policy found in [1] for simple aggregation.

II. Methodology

A. Evolving Segregation

In order to avoid manually spending time picking parameters or spending tremendous compute time using grid-search time find parameters, we use a simple genetic algorithm to evolve controller parameters. In all our experiments, we use 6 parameters for nothing, kin and non-kin, where each parameter ranges from -1 to 1. In our controller, we then scale this parameter to range from -20cm/s to 20cm/s. We treat aggregation as a special case of 1 class segregation, in which case the final two parameters of the controller are never used. 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 most fit individuals (the ones with lowest cost in our formulation), and then forming the next population by crossing the parameters of these two individuals. The original two parents are always kept in the population so that the current fittest individual is never removed until a more fit genome is discovered.

As with any genetic algorithm, it is important to have a fitness function that accurately assigns cost to behaviors. We explore two different cost functions, and we find that while both of them are intuitive they did not behave as we expected. The first is the cluster metric used by [1], 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.

We also propose a new cost function to adress 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. Consider the scenario where different classes of swarm robots needs to be collected after a mission, and the must be sorted into clusters so that they can be air-lifted easily. With this scenario in mind, we propose a new metric which uses the centroids of the robots in each class. We borrow the dispersion metric $u^{(t)}$ from [1], 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:

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

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

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

When we use this cost function in our genetic algorithm, we select fitnessest individuals as those with the lowest cost.

III. Controller Analysis

We found imperially that the parameters [0.5 -0.5 -0.2 0.6 0.6 0.2] successfully achieve segregation. We now analyze this controller formally. First, we prove that a single isolated robot executing this controller will aggregate to a kin robot, or more importantly, a cluster of kin robots. This proof is a variation of the Theorem 5.1 provided in [1]. Consider a robot situated as shown in Figure 1. The controller dictates that since a Kin robot is seen, $I = 1$, so $v_{l_1} = -0.2$ and $v_{r_1} = 0.6$. This will cause the robot to turn with some radius R. Without loss of generality, we define our coordinate system so that $c_i = [0, 0]$ with the robot 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 small 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:

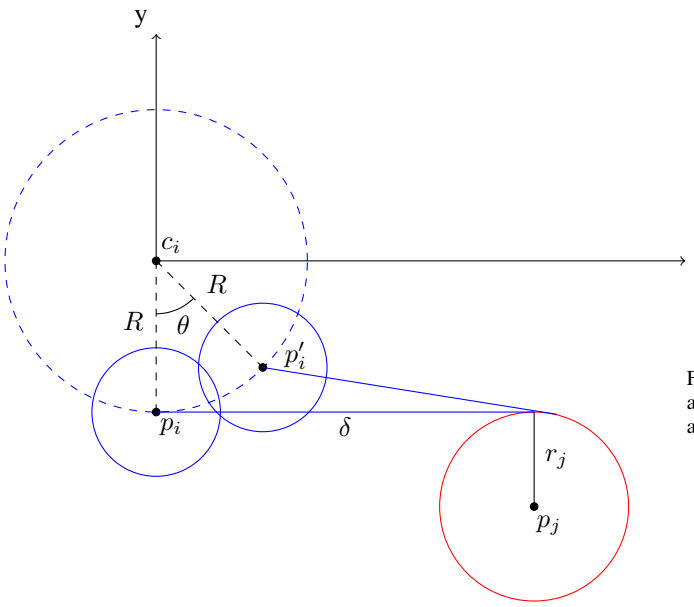

Fig. 1. The robot will always move closer to its Kin.

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

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 and assumptions stated, we can define the coordinate 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 VI-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. 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 VI-B, the change in distance between the robots is equal to

$$2R\big((r_j + R)(1 - \cos(\theta)) - \delta\sin(\theta)\big)$$

In both of these equations the dependant variables θ and R are themselves a function of the inter wheel diameter W and the wheels speeds of the controller v_{l_1} and v_{l_1}, which are shown in Equation 4.

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

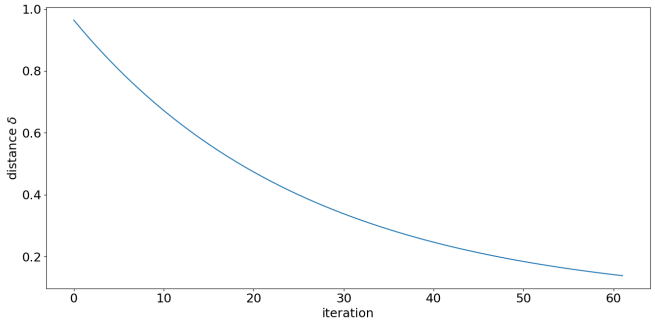

Fig. 2. The distance between a robot and its kin after each step as it aggregates. We use $W = 0.14$, $r_j = 0.14$, $\Delta t = 0.1$. In this example, aggregation took 60 iterations.

$$\begin{aligned} \theta &= \Delta t\omega = \Delta t\frac{v_{r_1} - v_{l_1}}{W} = \Delta t\frac{0.6 - (-0.2)}{W} = \frac{0.8\Delta t}{W} \\ R &= \frac{W}{2}\left(\frac{v_{r_1} + v_{l_1}}{v_{r_1} - v_{l_1}}\right) = \frac{W}{2}\left(\frac{0.6 + (-0.2)}{0.6 - (-0.2)}\right) = \frac{W}{4} \end{aligned} \tag{4}$$

Combining 4 with 3, we get a useful expression that tells us the conditions under which aggregation is guaranteed (III). 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 $1\,\mathrm{s}$ then aggregation is likely not guaranteed. We can also determine exactly how much closer the robots get each time, and since this is a function of δ we can plot how the distance between the robots changes over time. We show an example of this for our simulated robot in Figure 2

$$\delta > \left(\frac{W}{4} + r_j\right)\tan\left(\frac{0.8\Delta t}{2W}\right)$$

We can now consider an example with the Khephera IV robot. The inner wheel distance for the Khephera IV is $W = 0.14\,\mathrm{m}$, and in our simulations we use a time step of $\Delta t = 0.1s$. Therefore $\theta = 0.5714$ rad and $R = 0.035$, and so aggregation is guaranteed when the following is true.

$$\begin{aligned} \delta &> \left(0.035 + r_j\right)\tan\frac{0.5714}{2 * 0.14} \\ \delta &> -1.96904r_j - 0.06892 \end{aligned} \tag{5}$$

Note the largest possible value on the right hand side is achieved when $r_j = 0$, at which point it equals -0.06892. However, since the distance between two robots δ must always be positive it will always be greater than -0.06892, we can say that it is guaranteed that Equation 5 holds true for any r_j.

IV. Experimental Results

A. Simulation Environment

We first extended the binary sensor presented in [1] to be able to detect multiple types of robots. In our simulations, We use the range and bearing sensor with the FootBot robots and the ARGoS simulator, to detect whether a robot is in line of

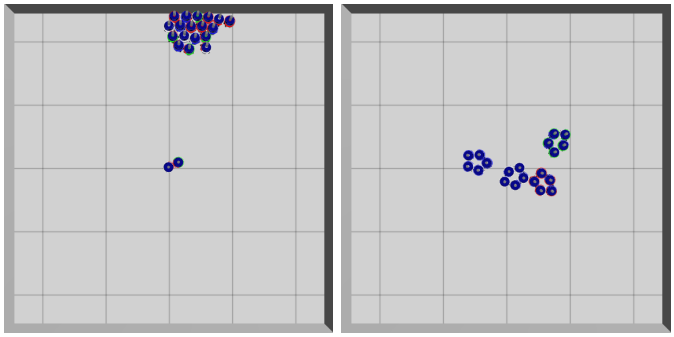

Fig. 3. The left picture was given lower cost than the right, which is not desirable.

sight. If there is no robot, the sensor returns 0. If there is a robot, then the sensor returns 1 if the robot is kin (in the same class), and 2 if the robot is non-kin (in any other class).

B. Evolving Aggregation

C. 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 3. 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.

D. Grid Search

E. The effect of Beam Angle

F. The effect of Beam Length

V. Future Work

References

[1] 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.

[2] 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.

[3] 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 <Emphasis Type="Italic">SwarmBot</Emphasis>. *Autonomous Robots*, 17(2-3):223–245, September 2004.

[4] 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.

[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).

VI. 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

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