Heterogeneous Swarm Formation Control Using Bivariate Normal Functions to Generate Potential Fields

Laura Barnes, Wendy Alvis, MaryAnne Fields*, Kimon Valavanis, Wilfrido Moreno

Dept. of CSE and EE, Center for Robot Assisted Search and Rescue
University of South Florida, Tampa, FL 33620
(*) U.S. Army Research Lab– Aberdeen, MD 21005-5059

Abstract

A novel method is presented for dynamic heterogeneous swarm formation control with potential fields generated from bivariate normal probability density functions (pdfs) used to construct the surface which swarm members move on, controlling swarm geometry, individual member spacing, and managing obstacle avoidance. Limiting functions are defined to provide tighter swarm control by modifying and adjusting a set of control variables forcing the swarm to behave according to set constraints. Bivariate normal functions and limiting functions are combined to guarantee obstacle avoidance and control swarm member orientation and swarm movement as a whole. This approach compared to others, is simple, computationally efficient, scales well to different swarm sizes, to heterogeneous systems, and to both centralized and decentralized swarm models. The method is applied to a simple vehicle model and simulation results are presented for a heterogeneous swarm of ten robot vehicles following line and ellipse formations.

1. Introduction

Formation control of multi-robot systems performing a coordinated task has become a challenging research field. The formation control problem is defined as finding a control algorithm ensuring that multiple autonomous vehicles can uphold a specific formation (or specific formations) while traversing a trajectory and avoiding collisions simultaneously.

The research focuses in *the ability to modify formation of a heterogeneous swarm given a set of parameters and limiting functions*. The swarm should be able to recover and reconfigure itself in the event of loss of a team member, loss of configuration or addition of a new team member.

The main paper contribution is the proposed swarm formation control method that is based on troop movement models [1-2] satisfying scalability (to varying numbers of swarm members) and supporting multiple formations. The central objective is the *overall group formation and behavior*, *not the control of an individual swarm member*.

The proposed solution is based on potential fields generated by bivariate normal functions that are used to control the swarm geometry and the inter-member spacing, as well as manage obstacle avoidance. The rationale behind using potential fields for formation control is that they facilitate in a simple and straightforward way the representation of multiple constraints and goals in a swarm system. Repulsive and attractive forces are utilized to control the overall swarm formation. Bivariate normal functions are used to create the vector fields that control the velocity and heading of robot swarms, with the swarm located on the surface created by the function.

The potential field method does have a problem with local minima. This problem will be considered in future work, but because the vector fields are dynamically changing at each time step in this work, the chances of hitting local minima are lower.

The advantage and difference of this approach compared to others lies at first in the simplicity of the vector generation. Control parameters are easily modified and adjusted to change formation and relative distance between swarm members, while other methods are rigid in formation constraints [13]. The proposed method scales well to different swarm sizes as well as to heterogeneous swarms because the vector field generation is independent of the specific robot vehicle architecture (as opposed to approaches in [18], [19] that are scalable in size but not in heterogeneity).

The robot vehicle controller receives the generated vector as input and translates it to the corresponding motion. For this paper a simple broadcast communication model is implemented. This simple

Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)
0-7695-2589-X/06 $20.00 © 2006 IEEE

model is used because the scope of this paper is not on communication but on the actual control and manipulation of the swarm into formations.

The presented method is demonstrated using physical robot vehicles modeled after an RC-truck with Ackerman steering. A model is derived and simulations using up to ten (heterogeneous) such vehicles are presented, including two formations to demonstrate applicability of the approach.

Section 2 discusses related work. Potential field generation is presented in Section 3. Section 4 presents the robot model with results shown in Section 5. Conclusions and future work are presented in Section 6.

2. Related Work

When looking at robot formation requirements for convoy protection, it is important that the approach be flexible in formation geometry and adaptable to varying team sizes.

Existing methods depend on a central controller [3-8], but dependence on a single entity is prone to failure and it is not ideal for missions where there is a high possibility of failure. Decentralized formation control methods are presented in [9-12, 24]. The decentralized method presented in [10] arranges robots in the spatial pattern of a crystal; the method suffers from heavy computational complexity overhead related to arranging the robots into the specific formation. In [24], a method for swarms of robots to generate patterns based on implicit functions is presented, but this method does not have the ability for dynamic changes. Other methods are rigid in formation constraints giving each robot fixed node assignments or trajectories [13]. Other approaches include graph theoretic ones [14-16], neighbor and leader-following methods [3, 5, 17] and potential functions [18-22].

Potential fields are the basis in many swarm formation models. In [18] and [19] strategies are presented to arrange large scale robot teams in a geometric formation utilizing potential functions; the approaches are scalable in size but not in heterogeneity. In [19] artificial potentials define the interaction control forces between adjacent vehicles and the desired inter-vehicle spacing; to do so, the use of virtual leaders or beacons (not an actual vehicle) is required. In [20], design and implementation of a tool to model and simulate collective behavior and interactions of a group of thousands of robots is presented. Social potential fields are utilized for coordinated group behavior where robots are to stay at a specified distance from each other to achieve optimum coverage of an area; no particular formation is discussed.

Troop movement models are discussed in [1-2]. In [1], reaction diffusion equations (RDEs) are used to describe behavior and control the troop as a group. Troop behavior is described with RDEs, also describing motion and diffusion of the troop. In [2] troop movement model is extended by combining Variable Resolution Terrain (VRT) model [23] and RDEs. VRT was developed to represent the battlefield as a continually differentiable surface. When VRT is combined with RDE, it creates a simulation tool to model troop movement on simulated battlefields.

The foundation for the potential field generation is presented next; details for swarm formation control, obstacle avoidance, swarm orientation and movement are given.

3. Vector Generation

3.1. Description of Functions

Bivariate normal functions may be used to create vector fields that control the velocity and heading of robot swarms, with the swarm located on the surface of the function.

A bivariate normal function with form as given in (1):

$$f(x,y) = e^{-\alpha(x-x_c)^2-\beta(y-y_c)^2} \tag{1}$$

produces an oval/ellipse shaped function. Assuming that the current robot location is at *(x, y)*, the center of the function in (1) is represented by (x_c, y_c) with respect to the world reference frame. The 'control' variables α and β control the strength of the vectors in the x and y directions, respectively, affecting how closely the swarm holds together along x-axis and y-axis, respectively. The x and y partial derivatives create the velocity vectors that are used to determine the heading and velocity of each member of the swarm as shown in (2):

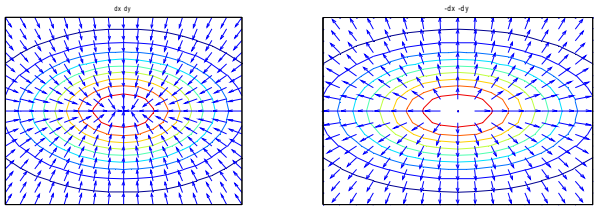
$$\begin{aligned} d_x &= f(x,y)(-2\alpha(x-x_c)) \\ d_y &= f(x,y)(-2\beta(y-y_c)) \end{aligned} \tag{2}$$

Just as with a single robot, the swarm formation, treated as a single shape, has both a local reference and a world reference frame. For the swarm to follow a trajectory in the world reference frame, an axis rotation is required. The heading, ϕ between the swarm formation's x-axis and the center (x_c, y_c) must be found; the rotated coordinates for (x, y) and (x_c, y_c) can be found using (3):

$$\begin{aligned} x_{rot} &= x\cos(\phi) + y\sin(\phi) \\ y_{rot} &= -x\sin(\phi) + y\cos(\phi) \end{aligned} \tag{3}$$

The rotated coordinates are then substituted back into (2) to find d_x and d_y.

Vector calculus dictates that the gradient vector field (d_x, d_y) points in the direction of greatest increase of the function *f(x, y)*, which is towards the center as illustrated in Figure 1a. The gradient vector field –(d_x ,d_y) points away from the center as shown in Figure 1b.



(a) **(b)**

Figure 1. Vector fields directed (a) toward the center and (b) away from the center.

Tighter swarm control may be accomplished when restricting the influence of the vector fields into a smaller region of the *x-y* plane by multiplying each of the fields by a 'limiting function'. This limiting function controls how far from the center the vector fields 'die out' in the *x* and *y* directions.

Vector fields 'moving away' from the center require a limiting function that approaches zero as the distance from the center is increased; such a limiting function is given in (4):

$$S_{out}(\alpha_{out}, x, x_c, y, y_c) = 1 - \frac{1}{1 + e^{-\alpha_{out}(\sqrt{(x-x_c)^2+(y-y_c)^2})}} \quad (4)$$

Gradient vector fields directed towards the center are required to approach zero as the vectors 'move towards' the center; this is achieved using the limiting function in (5):

$$S_{in}(\alpha_{in}, x, x_c, y, y_c) = \frac{1}{1 + e^{-\alpha_{in}(\sqrt{(x-x_c)^2+(y-y_c)^2})}} - \frac{1}{2} \quad (5)$$

Each of the limiting functions in (4) and (5) contains *tuning parameters* that may be used as *gradient vector field control variables*. Functions S_{out} and S_{in} in (4) and (5), respectively, include one tuning parameter each, α_{out} and α_{in}, determining how quickly the function approaches zero.

Functions S_{out} and S_{in} impose additional restrictions and constraints on top of and in addition to the initial swarm function *f(x, y)*. Limiting functions are not a necessity but they provide a much tighter level of control by limiting and restricting where the vector fields begin and end. The limiting functions, along with vector fields created by the bivariate normal function, may be combined to create swarm movement as a group. This forms the velocity and direction of the swarm movement with respect to the center of the swarm, as shown in (6):

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = S_{out} \begin{bmatrix} -d_x \\ -d_y \end{bmatrix} + S_{in} \begin{bmatrix} d_x \\ d_y \end{bmatrix} \quad (6)$$

3.2. Obstacle Avoidance

Normal functions can also be used for obstacle avoidance by creating vectors moving away from the center of the obstacle location (x_{co}, y_{co}). The distance at which a swarm member turns away from the obstacle may be controlled by a limiting function. The same form of limiting function shown in (4) may be used with the normal function as well. Obstacle avoidance is accomplished using equations (7) to (9):

$$S_{avoid}(\alpha_{avoid}, x, x_{co}, y, y_{co}) = 1 - \frac{1}{1 + e^{-\alpha_{avoid}(\sqrt{(x-x_{co})^2+(y-y_{co})^2})}} \quad (7)$$

$$\begin{bmatrix} d_{x\_avoid} \\ d_{y\_avoid} \end{bmatrix} = f_N(x, y) \begin{bmatrix} (-2\omega_{avoid}(x - x_{co}) \\ (-2\omega_{avoid}(y - y_{co}) \end{bmatrix} \quad (8)$$

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = S_{avoid} \begin{bmatrix} d_{x\_avoid} \\ d_{y\_avoid} \end{bmatrix} \quad (9)$$

Avoidance of individual robot swarm members and obstacles is accomplished in two ways: i) by controlling the speed at which robots move away when approaching an avoidance vector field, and, ii) by controlling the range the vector field is allowed to exist. The parameter ω_{avoid} controls the vector strength in the *x* and *y* directions swarm members move around an obstacle centered at (x_{co}, y_{co}). The α_{avoid} parameter in (7) controls how quickly vector fields die out when approaching obstacles. As α_{avoid} decreases, the distance of the avoidance vector field increases.

If the vector's strength was the only factor controlling obstacle avoidance, the magnitude of the vectors would have to be limited corresponding to slowing down 'reaction' by limiting the strength of the avoidance vectors. By including S_{avoid} into the calculation of v_x and v_y, see (9), the vector field may have a large magnitude allowing for quick response, but over much shorter distances. It is important to note, however, that vector fields can be made either strong enough so that they do not die out over a 'reasonable' distance even with the influence of A, or weak enough to die out before reaching the limiting distance of S_{out}. Using (9) and controlling two variables allows for tighter control over the robot's movement.

3.3. Orientation of Swarm Members

A normal function may be also used as an attractive point (x_{atrct}, y_{atrct}) ahead of the swarm to force swarm members to be oriented in the correct direction while heading to the next waypoint. This may be achieved by creating the vectors oriented towards the center of the normal function. Once again, the distance the vector field extends to, may be limited by using a limiting function similar to the one in (5). The attractive point generating function is given by equations (10) to (12):

$$S_{atrct}(\alpha_{atrct}, x, x_{atrct}, y, y_{atrct}) = \frac{1}{1+e^{-\alpha_{atrct}(\sqrt{(x-x_{atrct})^2+(y-y_{atrct})^2}}} - \frac{1}{2} \quad (10)$$

$$\begin{bmatrix} d_{x\_atrct} \\ d_{y\_atrct} \end{bmatrix} = f_N(x,y) \begin{bmatrix} -2\omega_{atrct}(x - x_{atrct}) \\ -2\omega_{atrct}(y - y_{atrct}) \end{bmatrix} \quad (11)$$

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = S_{atrct} \begin{bmatrix} d_{x\_atrct} \\ d_{y\_atrct} \end{bmatrix} \quad (12)$$

Equations (11) and (12) create the vectors towards the attractive point. In equation (11), ω_{atrct} controls the strength of the vectors towards the attractive point; in equation (12) the vectors are formed limited by function S_{atrct}.

3.4. Control of Swarm Movement

The different vector fields for swarm movement, obstacle avoidance and trajectory following are created independently of one another, but then summed to create a desired swarm movement. The limiting functions, S_{out} and S_{in}, along with the vector associated with the bivariate normal functions are used to control trajectory following and the distance from the center. A point of attraction can force orientation of the swarm members towards a desired waypoint. The obstacle avoidance function can be used to prevent swarm members from colliding with obstacles as well as with each other when following a desired path. A combination of all of the above, along with a shift in the center of the swarm, creates an overall movement of a group of swarm members as a whole. Generation of vector fields for the swarm following a straight trajectory in different formations is presented in Section IV.

The swarm may move from one waypoint to another by moving the center of the swarm (x_c, y_c). However, it is important to force each robot's orientation towards the direction to be traveled. This is achieved by placing a point of attraction just ahead of the swarm. The equation to create vector fields to follow a trajectory with obstacle avoidance by summing equations (6), (9) and (12) is given in (13):

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = S_{out} \begin{bmatrix} -d_x \\ -d_y \end{bmatrix} + S_{atrct} \begin{bmatrix} d_x \\ d_y \end{bmatrix} + \sum_{1}^{\#obstacles} S_{avoid} \begin{bmatrix} d_{x\_avoid} \\ d_{y\_avoid} \end{bmatrix} + S_{atrct} \begin{bmatrix} d_{x\_atrct} \\ d_{y\_atrct} \end{bmatrix} \quad (13)$$

4. Robot Swarm Implementation

A physical robot has much more limited movement than a particle; therefore, a robot model may be implemented with the vector field generator to validate the applicability of the proposed approach for swarm control.

A working model of an *RC-Truck* with Ackerman steering has been derived to demonstrate simple controller design (see Figure 2) used in conjunction with the vector fields. Model simplifications relate to neglecting both the losses in the drive train and motor, as well as slippage of the wheels in the kinematics model.

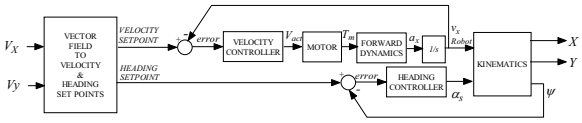


Figure 2. Block diagram of truck model with feedback

4.1. Forward Body Reference Dynamics

The primary force on the truck is the forward motion due to the torque generated by the motor. Other forces acting on the truck such as ground resistance, wind or uneven ground, have not been included in the model. These forces can be overcome by a well designed velocity controller and, therefore, may be neglected in the context of this study. The equation used to calculate the velocity in the forward direction of the robot is given by (14). This represents the mass on wheels portion of the model. The force acting on the vehicle is the torque, $T_m(t)$, produced by the motor increased by the gear ratio between the motor, N_{mw}, the wheels and decreased by the radius of the wheels, r_w, and the mass of the vehicle which is the weight of the vehicle, W, divided by the force of gravity, 32.2 ft s^{-2}.

$$v_{x\_robot}(t) = \int_0^t \frac{N_{mw} T_m(t)}{(W/32.2) r_w} dt \quad (14)$$

Unlike simulated models, robots with the exact same design will not behave identically. In order to

reflect this in simulation, some of the robot's constants relating to the physical characteristics were changed between robots to reflect this. These are given in TABLE I. Notice that the some of the characteristics have been changed to a large enough degree that the swarm better reflects a heterogeneous swarm.

Table 1. Robot Physical Parameters

Robot	N_{mw}	r (ft)	W (lbs)	V_{sp} limit	d_w (ft)
1	10	0.05	4	20	0.5
2	15	0.07	8	20	0.6
3	18	0.09	10	20	0.7
4	20	0.1	14	20	0.8
5	23	0.12	16	20	0.9
6	26	0.15	17	20	1.0
7	29	0.18	20	20	1.1
8	32	0.2	25	20	1.2
9	35	0.22	30	20	1.3
10	40	0.25	33	20	1.4

4.2. Motor Model

Equations (15), (16) and (17) are used to model the electric motor of the RC-truck. The input variable (control variable) is the motor voltage. The output of the motor model is the torque applied to the drive train of the RC-truck model. The constants related to motor specifications are listed in Table II.

Table 2. Motor Parameters

Symbol	*Description*	*Value*
R	Electrical resistance	*0.26* mΩ
L	Electrical inductance	*0.1* uH
K_t	Motor torque constant	*0.000162* Nm/s
b	Motor dampening ratio	*0.0005* Nms
K_v	Motor voltage constant	*11.67* radian/Volt-s
J	Motor inertia	*0.002* kg-m^2/s^2

The motor variables are the current $i(t)$, the angular velocity $w_m(t)$, the input voltage $V_{in}(t)$ and the output torque $T_m(t)$. The output of the motor is converted to lbs-ft to match the units in the summation of forces in the x-direction.

$$\frac{di(t)}{dt} = -\frac{R}{L}i(t) - \frac{1}{K_v L}w_m(t) + \frac{V_{in}(t)}{L} \quad (15)$$

$$\frac{dw_m(t)}{dt} = \frac{K_t}{J}i(t) - \frac{b}{J}w_m(t) \quad (16)$$

$$T_m(t) = K_t i(t) \quad (17)$$

4.3. Kinematic Calculations

Kinematic equations for a bicycle model have been used to convert the motion along the x-body axis to truck's position in the world reference frame. These equations are given in (18), (19) and (20), where $v_{x\_robot}(t)$ is equal to the velocity in the body reference frame, d_w is the distance between the center of the front and back wheels, value given in Table 2, $\alpha_s(t)$ is the steering angle of the front tires, and $\psi(t)$ is the heading in the world reference frame. In addition, the steering angle of the truck has been limited to *+/-30* degrees due to the physical limitations of Ackerman steering:

$$X(t) = \int_0^t v_{x\_robot}(t)\cos(\alpha_s(t))\cos(\psi(t))dt \quad (18)$$

$$Y(t) = \int_0^t v_{x\_robot}(t)\cos(\alpha_s(t))\sin(\psi(t))dt \quad (19)$$

$$\psi(t) = \int_0^t \frac{v_{x\_robot}(t)}{d_w}\sin(\alpha_s(t))dt \quad (20)$$

Control has been implemented by first converting the x and y vector inputs to velocity and heading set points, then implementing feedback with proportional controllers to maintain the set points, see Figure 2.

$$v_{sp}(t) = \kappa\sqrt{v_x^2(t) + v_y^2(t)} \quad (21)$$

Equation (21) is used to calculate the velocity set point. For simplicity of design, a scale factor κ has been set along with a limit given in Table II. The primary objective of calculating the velocity set point from the generated vectors is to slow the movement of the truck as the generated vectors decrease. As the robots approach the way point, $v_x(t)$ and $v_y(t)$ approach zero reducing the velocity set point. The scale factor allows for further control over the velocity of the truck without altering the vector field generation. For the simulations shown in this study, κ = 10. This parameter increased the velocity of the robot by a factor of ten without the necessity of recalculating the vector field tuning parameters.

The heading set point is controlled by calculating the angle between the x and y velocity vectors, $v_x(t)$ and $v_y(t)$, generated by the bivariate and normal functions shown in equation (22):

$$\psi_{sp}(t) = \tan^{-1}(v_y(t)/(v_x(t)) \quad (22)$$

It is important to note that, while the mathematical value for the inverse tan of infinity and negative infinity is 90 degrees and -90 degrees, respectively, some programming languages will not allow this condition. If this is the case, then additional code must be included to check for division by zero.

The velocity control has been implemented with a proportional controller in the body reference frame. This is given in (23), where K_P is the controller constant, $v_{x\_robot}(t)$ is the velocity in the body reference

frame and $v_{sp}(t)$ is the velocity set point calculated from the generated vector fields:

$$V_{act}(t) = K_P\left(v_{sp}(t) - v_{x\_robot}(t)\right) \quad (23)$$

The steering angle, $\alpha_s(t)$, is set to the difference between the vehicles desired heading, $\psi_{sp}(t)$, and the actual heading of the vehicle, $\psi(t)$, in the world coordinate frame:

$$\alpha_s(t) = \psi_{sp}(t) - \psi(t) \quad (24)$$

Simulation results are presented next.

5. Simulation Results

Simulink has been used to model the robot swarm. Ten robot trucks implementing probability density function generated vector control are used for the simulations. The swarm formation controller is programmed in C and included in the Simulink model along with the robots. Each individual robot's vector generating controller is implemented as a C MEX S-function with the different control parameters fed in as well as a position vector with obstacle locations, including other robots, within a 50 ft radius. This general configuration is displayed in Figure 3. Each of the C MEX S-functions generates the vector fields presented in Section III at each time step. Formation changes are easily made by determining which fields will be included in the final vector summation equation and updating just a few tuning parameters. The center of the swarm and any attractive waypoints are generated outside the Simulink model and sent into the vector generating m-files at the appropriate time steps to generate the desired movement of the swarm as a whole.

For this work, since no sensitivity analysis has been performed, all parameters are selected heuristically. This is not an ideal selection process, but sufficient to demonstrate capabilities of this method.

Assuming a closed world frame, all avoidance vectors for fixed obstacles can be generated before run-time, but in order to accurately represent real world sensors, all obstacles outside of a 50 foot radius are neglected. The robots are not required to have any knowledge about the others location and will spread evenly across the surface of the function. The only obstacles included in these simulations are the other robots. The computational complexity of the vector generation is dependent on the number of obstacles because this is the only factor in the equations which has potential to be continuously growing. The complexity will grow in denser environments as well as when the size of the swarm increases. This complexity is due to the fact that at each time step, an avoidance vector for *n* obstacles and/ or robots within a certain range must be generated. The next two sections demonstrate the swarm following a straight trajectory in line and ellipse formations.

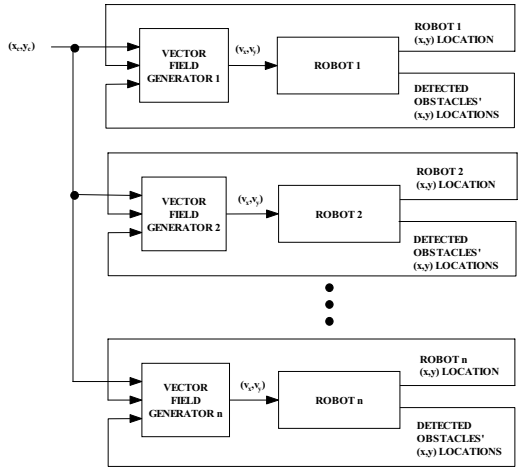


Figure 3. *Matlab Simulink* swarm simulation model with n robots

5.1. Ten Robots in Line Formation

Simulations demonstrating a swarm of ten robots moving into a line formation are presented. In order to force the robots into a line formation α must be very small in comparison to β so the surface of the ellipse function from equation (1) is long and skinny. The fields then force the robots to line up to adhere to the function parameters. The parameters for the swarm function as well as the limiting functions are given in Table 3.

Table 3.
Control Variables with Ten Robots

Control Variables	Line Formation	Ellipse Formation
α	0.095310	2.397895
β	3.091042	3.091042
α_A	20	20
α_C	30	30
α_{avoid}	10	10
ω_{avoid}	.01	.01
α_{atrct}	20	20
ω_{atrct}	.05	.05

The robots form a line avoiding each other and successfully followed a trajectory. All ten robots were slightly different but used the identical vector generation code. The other parameters were fixed dues to the difficulty of parameter selection without tuning. Figure 4 demonstrates the swarm moving into line formation.

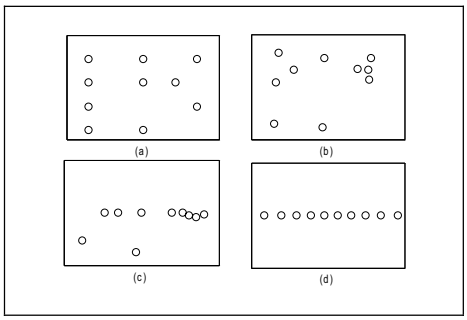


Figure 4. *Line formation with 10* robots at (a) t=1. (b) t=25. (c) t=50. (d) t=100

5.2. Ten Robots in Ellipse Formation

A second case study demonstrates control of ten robots following a straight trajectory and a sine trajectory. In this case, $\beta > \alpha$ to force a narrower ellipse configuration along the path being followed. Each robot avoided other robots in the swarm and generated an avoidance field at the correct location to avoid a collision between swarm members. The center of the swarm was used as waypoints and it was incremented as the members were in a certain threshold of each waypoint. An attractive point was also set as the next center so the robots would always orient correctly along the trajectory. All ten robots used the same vector generation presented in equation (13) and the control variables as in Table III.

Figure 5 demonstrates the swarm following a straight trajectory along the *x*-axis at different time steps. The robots started in random positions and aligned themselves onto the band of the ellipse. The robots take longer to get into the ellipse formation versus the line but successfully maintain the formation while following a trajectory.

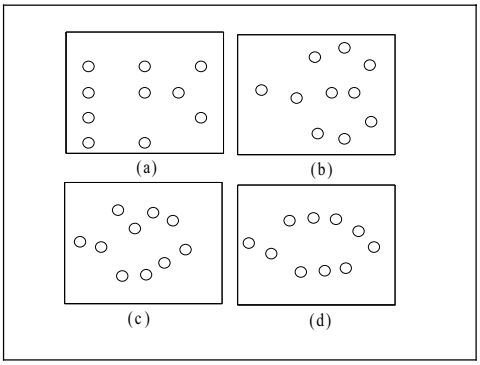


Figure 5. *Ellipse formation with 10* robots (a) t=1. (b) t=50. (c) t=100. (d) t=200

Figure 6 demonstrates the robots making an ellipse formation while following a sine wave trajectory. The formation is still tight even when the robots are forced to reorient constantly. Figure 7 shows the trajectories of each of the robots along the center. The robots exhibit the oscillating pattern because they do not all reach the center at the same time, and they rotate around the center area until a new center and attractive point are received.

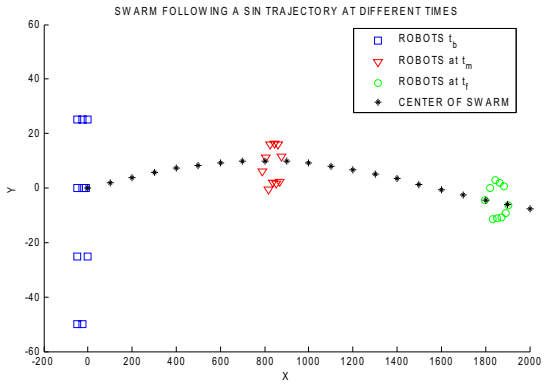


Figure 6. *Ellipse formation with 10* robots at different times

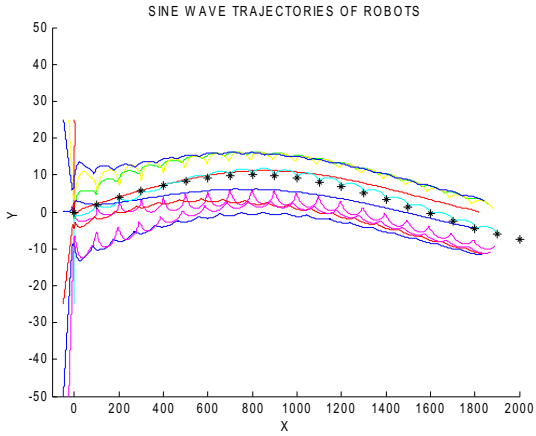


Figure 7. Sine wave trajectories with 10 robots for ellipse formation

6. Conclusions and Discussion

The paper has demonstrated that bivariate normal functions together with limiting functions can be successfully utilized to control robot swarm formation, obstacle avoidance and the overall swarm movement.

The presented method supports scalability, different swarm sizes, multiple formations, heterogeneous swarm member teams, centralized and decentralized formation control; since parameters are tuned off-line and used on line, the method is computationally inexpensive.

In the future, the work will be expanded to include more formations. Swarm member communication will also be improved using an ad-hoc network design for communication between members. A sensitivity

analysis of the control variables for the swarms function as well as the limiting functions will be performed with an automated tuner through a method such as neural networks or genetic algorithms to make the model more dynamic and scalable.

The approach will also be expanded to unmanned aerial vehicles (UAVs) by making the formation function a trivariate normal.

7. References

[1] M. Fields, "Modeling large scale troop movement using reaction diffusion equations," *ARL-TR-200*, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, September 1993.

[2] M. Fields, "Preliminary applications of the variable resolution terrain model to a troop movement model," ARL-*TR-842*, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, August 1995.

[3] M. Egerstedt and X. Hu. "Formation constrained multi-agent control," *IEEE Transactions on Robotics and Automation*, 17(6), 2001.

[4] R. O. Saber, W. B. Dunbar, and R. M. Murray, "Cooperative control of multi-vehicle systems using cost graphs and optimization," in *Proceedings of the American Control Conference*, Vol. 3, June 2003, pp. 2217–2222.

[5] S. Zilinski, T. Koo, and S. Sastry, "Optimization-based formation reconfiguration planning for autonomous vehicles," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2003.

[6] L. Chaimowicz and V. Kumar, "Coordination among UAVs and Swarm Robots," *International Symposium on Distributed Autonomous Robotic Systems*, 2004.

[7] W. Kowalczyk. "Target assignment strategy for scattered robots building formation," in *Proceedings of International Workshop on Robot Motion and Control*, 2002.

[8] Z. Cao, M. Tan, S. Wang, Y. Fan, and M. Zhang, "The optimization research of formation control for multiple mobile robots," in *Proceedings of the World Congress on Intelligent Control and Automation*, 2003.

[9] J. Fredslund and M. J. Mataric, "A general algorithm for robot formations using local sensing and minimal communication," *IEEE Transactions on Robotics and Automation*, 18(5), 2002.

[10] K. Fujibayashi, S. Murata, K. Sugawara, and M. Yamamura, "Self- organizing formation algorithm for active elements," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2000.

[11] K. Sugihara and I. Suzuki, "Distributed algorithms for formation of geometric patterns with many mobile robots," *Robot Systems*, 13(3), 1996.

[12] H. Yamaguchi, T. Arai, and G. Beni, "A distributed control scheme for multiple robotic vehicles to make group formations," *Robotics and Autonomous Systems*, 36:125-147, 2001.

[13] W. Kang, N. Xi, Y. Zhao, J. Tan, and Y. Wang, "Formation control of multiple autonomous vehicles: Theory and experimentation," in *Proceeding,. IFAC 15th Triennial World Congress*, 2002, pp. 1155–1160.

[14] R. Olfati-Saber and R. Murray, "Graph rigidity and distributed formation stabilization of multi-vehicle systems," *Proceedings, Conference on Decision and Control (CDC)*, 2002.

[15] J. P. Desai, "Modeling multiple teams of mobile robots: a graph theoretic approach," in *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, 2001.

[16] R. Fierro and A. K. Das, "A modular architecture for formation control", in *International Workshop on Robot Motion and Control*, 2002.

[17] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*. 48(6), June 2003.

[18] T. Balch and M. Hybinette, "Social potentials for scalable multi-robot formations", *Proceedings of IEEE International Conference on Robotics and Automation*, 2000.

[19] N. Leonard, E. Fiorelli, "Leaders, artificial potentials and coordinated control of groups," *Conference on Decision and Control,* Florida, 2001.

[20] D. D Dudenhoeffer and M. P. Jones, "A formation behavior for large-scale micro-robot force deployment," in *Simulation Conference Proceedings*, 2000.

[21] R. Olfati-Saber and R. Murray, "Distributed cooperative control of multiple vehicle formations using structural potential fields," *IFAC World Congress*, Barcelona, Spain, 2002.

[22] F. E. Schneider and D. Wildermuth, "A potential field based approach to multi robot formation navigation," in *Proceedings of International Conference on Robotics, Intelligent Systems and Signal Processing*, October 2003.

[23] J. K. Wald and C. J. Patterson, "A variable resolution terrain model for combat simulation", *BRL-TR-3374*, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, October 1994.

[24] L. Chaimowicz , N. Michael, and V. Kumar., "Controlling swarms of robots using interpolated implicit functions", in *Proceedings of International Conference on Robotics and Automation*, 2005.