# Swarm Formation Control with Potential Fields Formed by Bivariate Normal Functions

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

*Abstract*— **A novel method is presented for swarm formation control with potential fields generated from bivariate normal probability density functions (pdfs) that construct the surface the swarm members move upon controlling the swarm geometry and member spacing as well as manage obstacle avoidance. Limiting functions 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. The presented approach, compared to others, is simple, computationally efficient, and scales well to different swarm sizes and swarm models. The method is applied to a simple vehicle model, and simulation results are presented on a homogeneous swarm of ten robot vehicles for different formations.**

***Index Terms: swarm, formation control, potential fields, limiting functions.***

## I. 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 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. It is also imperative that the formation be reconfigurable based on given and dynamically changing parameters. 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*.

This work was supported in part by the U.S. Army Research Office (ARO) Grant Number W91-11NF-06-1-0069.

Laura Barnes and Kimon Valavanis are with the Computer Science Department, University of South Florida, Tampa, FL 33620 USA (e-mail: lbarnes3@ csee.usf.edu, kvalavan@csee.usf.edu).

MaryAnne Fields is with the U.S. Army Research Lab, Aberdeen, MD 21005 USA (e-mail: mafld@arl.army.mil).

Wendy Alvis and Wilfrido Moreno are with the Electrical Engineering Department, University of South Florida, Tampa, FL 33620 USA (e-mail: trietley@eng.usf.edu, moreno@eng.usf.edu).

The proposed solution is based on potential fields generated by bivariate normal functions that are used to control the swarm geometry (in a rather elegant way) 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, but because the vector fields are dynamically changing the chances of hitting a local minima are much smaller.

The advantage and difference of this approach compared to others lies 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]. Further, in the presented approach, since control parameters may be tuned off-line and used on-line, changing formation is computationally inexpensive (compared to, for example, the approach in [10] that suffers from computational complexity).

*The proposed method scales well not only to different swarm sizes but also to heterogeneous systems 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 (A) robot vehicle controller receives the generated vector as input and translates it to the corresponding motion. As swarm members are added, communication grows because each swarm member needs to know the position/location of other swarm members, but this will be solved by deriving a communication model (although beyond the scope of this paper) as part of the overall control model of the swarm. In addition, compared to centralized [3-8] or decentralized [9-12] approaches, the proposed method using bivariate normal pdfs and limiting functions is very general and scales well to both decentralized and centralized swarm models.

A physical robot modeled after an RC-truck with Ackerman steering is used to demonstrate the method. A model is derived and simulations using up to ten (homogeneous) such vehicles are presented. Section II discusses related work. Potential field generation is presented in Section III. Section IV presents the robot

model with results shown in Section V. Conclusions and future work are presented in Section VI.

## II. 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.

## III. Vector generation

### A. 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 within 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/ellipsoid 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 $\alpha$ and $\beta$ control the strength of the vectors in the $x$ and $y$ directions, respectively, affecting how closely the swarm holds together. 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):

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

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. The vector fields $(d_x, -d_y)$ and $(-d_x, d_y)$ are perpendicular to the gradient; Figure 1c. shows such a perpendicular field.

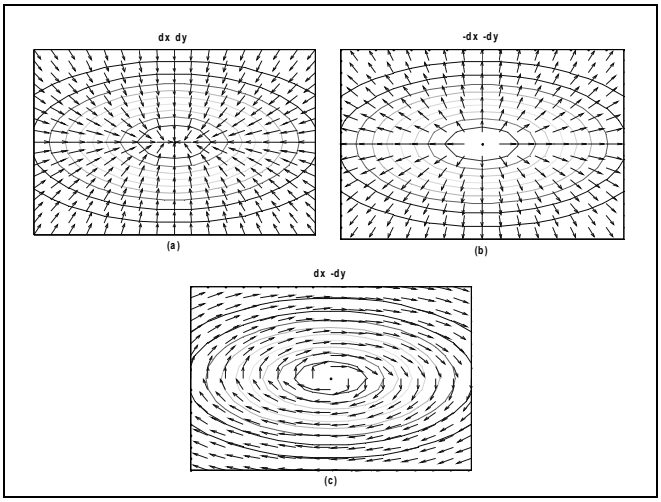


Fig 1. Vector fields directed (a) toward the center and (b) away from the center. (c) perpendicular to the center.

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 (3):

$$S_{out}(\alpha_{out}, x, x_c, y, y_c) = 1 - \frac{1}{1 + e^{-\alpha_{out}(\sqrt{(x-x_c)^2+(y-y_c)^2})}} \tag{3}$$

Perpendicular fields must be held within a narrow region at some distance from the swarm center, so that members are kept at a specified distance from the center. Therefore, the gradient vector field must approach zero as it 'moves towards' the center from one side of the narrow region and as it 'moves away' from the center of the other side of the region. A limiting function accomplishing that is given in (4):

$$N_{\perp}(\alpha_{\perp}, x, x_c, y, y_c) = e^{-\alpha_{\perp}(\sqrt{(x-x_c)^2+(y-y_c)^2}-d_C)} \tag{4}$$

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):

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

Each of the limiting functions in (3) to (5) contains *tuning parameters* that may be used as *gradient vector field control variables*. Functions $S_{out}$ *and* $S_{in}$ in (3) and (5), respectively, include one tuning parameter each, $\alpha_{out}$ and $\alpha_{in}$, determining how quickly the function approaches zero. Function $N_{\perp}$ in (4) includes two tuning parameters, $\alpha_{\perp}$ and $d_c$. The distance of the strongest vector field from the center of the swarm is controlled by $d_c$; the distance at which the fields die out on either side of this perpendicular path is controlled by $\alpha_{\perp}$.

Functions $S_{out}$, $N_{\perp}$ 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 added in different combinations to create swarm movement as a group. Different combinations of $S_{out}$, $N_{\perp}$ and $S_{in}$ create different forms of movement and when equations (2) to (5) are combined they form the velocity and direction of the swarm movement with respect to the center of the swarm, as shown in (6), where $_{\perp}$ denotes perpendicular:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = S_{out}\begin{bmatrix} -d_x \\ -d_y \end{bmatrix} + N_{\perp}\begin{bmatrix} d_x \\ -d_y \end{bmatrix}_{\perp} + S_{in}\begin{bmatrix} d_x \\ d_y \end{bmatrix} \tag{6}$$

A symmetric normal function as in equation (7):

$$f_N(x, y) = e^{-\omega[(x-x_c)+(y-y_c)]} \tag{7}$$

may be used in a similar fashion as the bivariate function in (1) to create vector fields as shown in (8):

$$\begin{bmatrix} d_x \\ d_y \end{bmatrix} = f(x, y)\begin{bmatrix} (-2\omega(x-x_c) \\ (-2\omega(y-y_c) \end{bmatrix} \tag{8}$$

The primary difference is that (7) creates a circle rather than an oval. While this limits control over the shape of the created vector fields, it does have the advantage of having only one tuning parameter that controls the vector field strength, making the design of the field simpler.

### *B. Obstacle Avoidance*

Normal functions may 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, too. The same form of limiting function shown in (3) may be used with the normal function as well. Obstacle avoidance is accomplished using equations (9) to (11):

$$S_{avoid}(\alpha_{avoid}, x, x_{co}, y, y_{co}) = 1 - \frac{1}{1+e^{-\alpha_{avoid}(\sqrt{(x-x_{co})^2+(y-y_{co})^2}}} \tag{9}$$

$$\begin{bmatrix} d_{x\_avoid} \\ d_{y\_avoid} \end{bmatrix} = f_N(x, y)\begin{bmatrix} (-2\omega_{avoid}(x-x_{co}) \\ (-2\omega_{avoid}(y-y_{co}) \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = S_{avoid}\begin{bmatrix} d_{x\_avoid} \\ d_{y\_avoid} \end{bmatrix} \tag{11}$$

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 $\omega_{avoid}$ controls the vector strength in the *x* and *y* directions swarm members move around an obstacle centered at *($x_{co}$, $y_{co}$)*. The $\alpha_{avoid}$ parameter in (9) controls how quickly vector fields die out when approaching obstacles. As $\alpha_{avoid}$ 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 (11), 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 $S_{avoid}$, or weak enough to die out before reaching the limiting distance of $S_{avoid}$. Using (11) and controlling two variables allows for tighter control over the robot's movement.

### *C. 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 (12) to (14):

$$S_{atrct}(\alpha_{atrct}, x, x_{atrct}, y, y_{atrct}) = \frac{1}{1+e^{-\alpha_{atrct}(\sqrt{(x-x_{atrct})^2+(y-y_{atrct})^2}}} - \frac{1}{2} \quad (12)$$

$$\begin{bmatrix} d_{x\_atrct} \\ d_{y\_atrct} \end{bmatrix} = f_N(x,y) \begin{bmatrix} -2\omega_{atrct}(x-x_{atrct}) \\ -2\omega_{atrct}(y-y_{atrct}) \end{bmatrix} \quad (13)$$

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = S_{atrct} \begin{bmatrix} d_{x\_atrct} \\ d_{y\_atrct} \end{bmatrix} \quad (14)$$

Equations (13) and (14) create the vectors towards the attractive point. In equation (13), $\omega_{atrct}$ controls the strength of the vectors towards the attractive point; in equation (14) the vectors are formed limited by function $S_{in}$.

### D. 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, $S_{out}$, $N_{\perp}$ and $S_{in}$, along with the vector associated with the bivariate normal functions are used to control circling around a point 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 circling a point and following a straight trajectory is presented in Section IV.

The swarm may be forced to circle around a point by defining the swarm center as the point to be circled. In this case, no attractive point ahead of the swarm is given, and the center of the swarm is held stationary. Avoidance fields may be generated for the other members of the swarm and any obstacle in the path of the swarm. The separate vector fields from equations (3) to (5) as well as (10) are summed to create trajectories as shown in (15):

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = S_{out} \begin{bmatrix} -d_x \\ -d_y \end{bmatrix} + N_{\perp} \begin{bmatrix} d_x \\ d_y \end{bmatrix}_{\perp} + S_{in} \begin{bmatrix} d_x \\ d_y \end{bmatrix} + \sum_{1}^{\#obstacles} S_{avoid} \begin{bmatrix} d_{x\_avoid} \\ d_{y\_avoid} \end{bmatrix} \quad (15)$$

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 (5), (10) and (14) is given in (16):

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \mathrm{S}_{\mathrm{out}} \begin{bmatrix} -d_x \\ -d_y \end{bmatrix} + \mathrm{S}_{\mathrm{atrct}} \begin{bmatrix} d_x \\ d_y \end{bmatrix} + \sum_{1}^{\#obstacles} \mathrm{S}_{avoid} \begin{bmatrix} d_{x\_avoid} \\ d_{y\_avoid} \end{bmatrix} + \mathrm{S}_{atrct} \begin{bmatrix} d_{x\_atrct} \\ d_{y\_atrct} \end{bmatrix} \quad (16)$$

Notice that the perpendicular field is not used, since the swarm is forced to move along a trajectory heading towards the next waypoint rather than circle.

## IV. Implementation on a Robot Swarm

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 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.

### A. 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 is given by (17), where $T_m(t)$ is the torque produced by the motor, $N_{mw}$ equal to 28.91 is the motor to wheel ratio, $r_w$ is the radius of the tire, $r_w$ equal to *0.3* ft, and $W$ is the weight of the truck, equal to *14.875* lbs.

$$v_{x\_robot}(t) = \int_0^t \frac{N_{mw} T_m(t)}{(W/32.2) r_w} dt \quad (17)$$

### B. Motor Model

Equations (18), (19) and (20) 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 as follows; $R$ is the electrical resistance equal to *0.26* mΩ, $L$ is the electrical inductance equal to *0.1* uH, $K_t$ is the motor torque constant equal to *0.000162* Nm/s, $b$ is the motor dampening ratio equal to *0.0005* Nms, $K_v$ is the motor voltage constant equal to *11.67* radian/Volt-s, and $J$ is the motor inertia equal to *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.

$$\frac{di(t)}{dt} = -\frac{R}{L}i(t) - \frac{1}{K_v L}w_m(t) + \frac{V_{in}(t)}{L} \tag{18}$$

$$\frac{dw_m(t)}{dt} = \frac{K_t}{J}i(t) - \frac{b}{J}w_m(t) \tag{19}$$

$$T_m(t) = K_t i(t) \tag{20}$$

### C. 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 (21), (22) and (23), 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 equal to *0.5* ft, $\alpha_s(t)$ 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:

$$X(t) = \int_0^t v_{x\_robot}(t)\cos(\alpha_s(t))\cos(\psi(t))dt \tag{21}$$

$$Y(t) = \int_0^t v_{x\_robot}(t)\cos(\alpha_s(t))\sin(\psi(t))dt \tag{22}$$

$$\psi(t) = \int_0^t \frac{v_{x\_robot}(t)}{d_w}\sin(\alpha_s(t))dt \tag{23}$$

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.

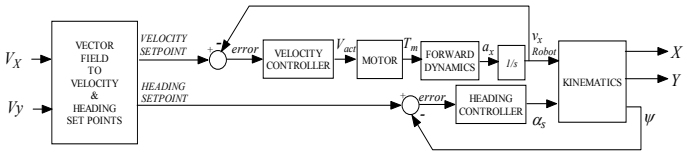


Fig 2. Block diagram of truck model with feedback

Equation:

$$v_{sp}(t) = \kappa\sqrt{v_x^{\,2}(t) + v_y^{\,2}(t)} \tag{24}$$

is used to calculate the velocity set point. For simplicity of design, a scale factor $\kappa$ has been set along with a limit of *20* ft/sec. 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, $\kappa = 10$. 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 (25):

$$\psi_{sp}(t) = \tan^{-1}(v_y(t)/(v_x(t)) \tag{25}$$

The velocity control has been implemented with a proportional controller in the body reference frame. This is given in (26), where $K_P$ is the controller constant, $V_x(t)$ is the velocity in the body reference frame and $v_{sp}(t)$ is the velocity set point calculated from the generated vector fields:

$$V_{act}(t) = K_P\left(v_{sp}(t) - v_{x\_robot}(t)\right) \tag{26}$$

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:

$$\alpha_s(t) = \psi_{sp}(t) - \psi(t) \tag{27}$$

Simulation results are presented next.

## V. Simulation Results

*Simulink* has been used to model swarms of four to ten robot trucks implementing probability density function generated vector control. The swarm formation is modeled in *Simulink*. Each individual robot is represented as a C MEX *S*-function with the different control parameters fed in as well as a position vector with the other robots' locations as shown in Figure 3.

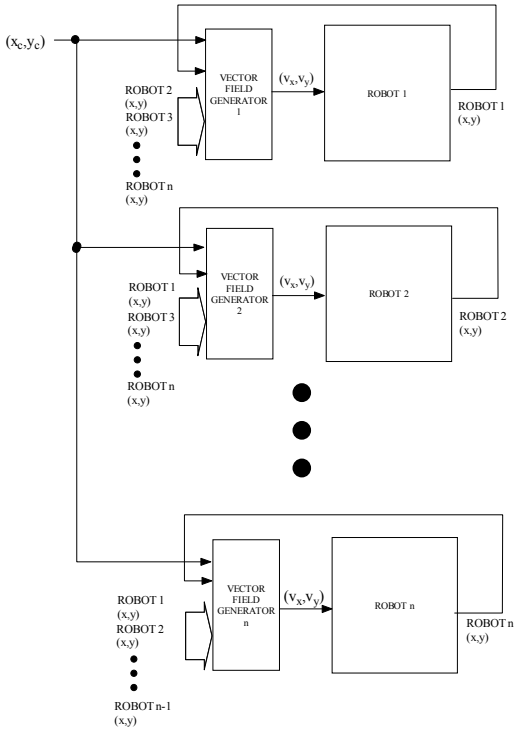


Fig 2. *Matlab Simulink* swarm simulation model with n robots

The *S*-function for each robot generates the vector fields at each time step. Each of these C MEX *S-functions*

included code to generate each of the vector fields presented in Section III. Formation changes are easily made by changing just a few tuning parameters and deciding about which fields will be included in the final vector summation equation. 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. Assuming a closed world frame, all avoidance vectors for fixed obstacles can be generated before run-time. The only dynamic obstacles at this state are assumed to be the other robots within the swarm. Without a sensor model, the computational complexity of the vector generation is *O(n)* where *n* is the number of swarm members. This complexity is due to the fact that at each time step, an avoidance vector for *n-1* robots must be generated. With a good sensor model this would be minimized or done only if necessary. The next two sections demonstrate the swarm circling a point and following a straight trajectory.

### *A. Robot Swarm of 10 Robots Circling Point*

Simulation results are also presented with ten robots for the circle trajectory. Table I summarizes the values used for the control variables. Figure 4 shows the 10 robots following a circle trajectory around a central point.

TABLE I
Control Variables with Ten Robots

| Control Variables | Circling Center Point | Following Trajectory |
|---|---|---|
| $\alpha$ | 3.0910 | 2.3979 |
| $\beta$ | 3.0910 | 0.0953 |
| $d_c$ | 10 | - |
| $\alpha_A$ | 0.2 | - |
| $\alpha_B$ | 0.001 | - |
| $\alpha_C$ | 1 | 30 |
| $\alpha_{avoid}$ | 8 | 15 |
| $\omega_{avoid}$ | 0.005 | 0.01 |
| $\alpha_{atrct}$ | 20 | 20 |
| $\omega_{atrct}$ | 0.05 | 0.05 |

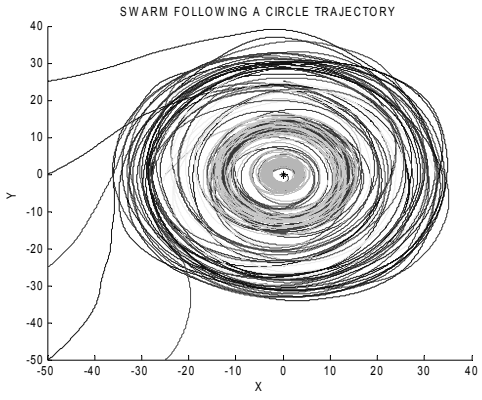


Fig 4. Swarm of 10 robots following a circle trajectory

### *B. Robot Swarm of 10 Robots Following Trajectory*

The last case study relates to implementing control of ten robots following a straight trajectory with different parameters. These simulations will also show the necessity of the limiting functions. The robots start initially at spread distances and approach the center, coming together into formation with different distances apart depending on the heuristically selected parameters.

Figure 5 shows the ten robots following a straight trajectory at different time steps, $t_b$ at the initial position, $t_m$ at the central time, and $t_f$ at the final state. The robots avoid each other and follow the bivariate function with center $x_c$ and $y_c$ given at different time steps. The robot formation becomes tighter adhering to the control parameters as time progresses, avoiding each other while traversing the waypoints in formation.

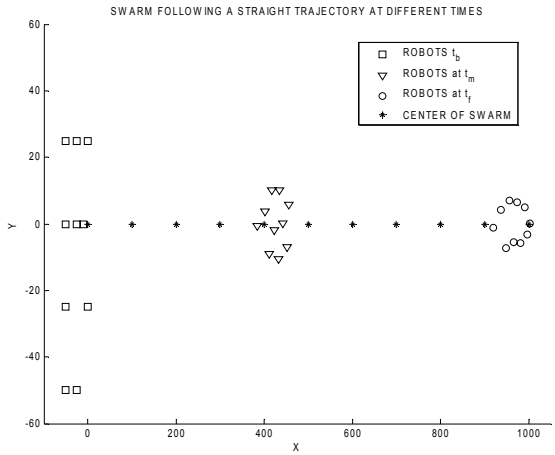


Fig 5. Swarm of 10 robots following a straight trajectory with formation shown at different times utilizing limiting functions

To demonstrate the necessity for the limiting functions, one simulation is run without using $S_{in}$ and $S_{out}$. Figure 6 shows the swarm configuration at different time steps, $t_b$ at the initial position, $t_m$ at the central time, and $t_f$ at the final state. The robots still avoid each other but they 'clump' together instead of adhering to the outer band of the ellipse.

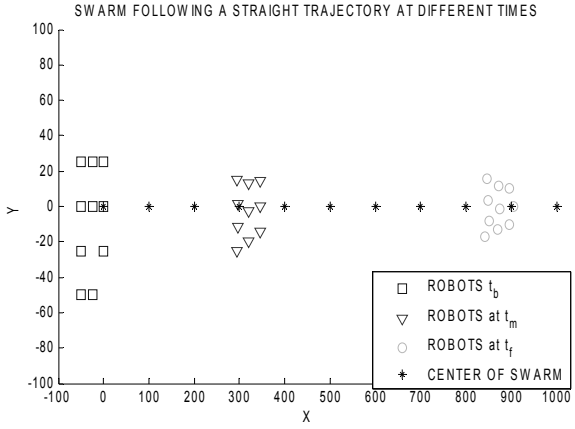


Fig 6. Swarm of 10 robots following a straight trajectory with formation shown at different times without utilizing limiting functions

In Figure 7, obstacles are placed throughout the robots' paths. The $\alpha_{avoid}$ is made smaller for this case so the avoidance vector field will extend further from the other robots and the obstacles. In addition $\omega_{avoid}$ is also changed so the reaction time is quicker in a more heavily populated area.

The robots avoid each other as well as the obstacles throughout navigation.

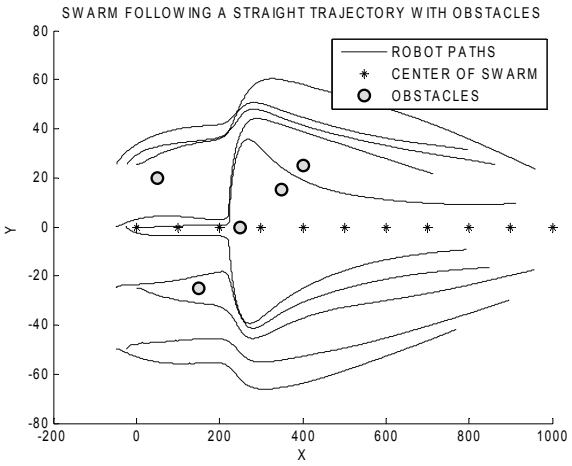


Fig 7. Swarm of 10 robots following a straight trajectory avoiding fixed obstacles $\alpha_{avoid}$=8 and $\omega_{avoid}$=0.005

## VI. Conclusions

The paper has shown that bivariate normal functions coupled by limiting functions may be successfully derived and implemented to control robot swarm formation, obstacle avoidance and swarm movement as a whole.

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 heterogeneous swarm systems. Swarm member communication will also be considered, as well as a leader/follower approach for robot movement. A sensitivity analysis of the control variables, $\alpha$ and $\beta$, of the swarm and limiting functions will be performed with automated tuning through neural networks 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.

## 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.