Lab Papers (GRASP)

General Robotics, Automation, Sensing and
Perception Laboratory

4-1-2005

Controlling Swarms of Robots Using Interpolated Implicit Functions

Luiz Chaimowicz
University of Pennsylvania, chaimo@grasp.upenn.edu

Nathan D. Michael
University of Pennsylvania, nmichael@seas.upenn.edu

Vijay Kumar
University of Pennsylvania, kumar@seas.upenn.edu

Controlling Swarms of Robots Using Interpolated Implicit Functions

Abstract

We address the synthesis of controllers for large groups of robots and sensors, tackling the specific problem of controlling a swarm of robots to generate patterns specified by implicit functions of the form s(x, y) = 0. We derive decentralized controllers that allow the robots to converge to a given curve S and spread along this curve. We consider implicit functions that are weighted sums of radial basis functions created by interpolating from a set of constraint points, which give us a high degree of control over the desired 2D curves. We describe the generation of simple plans for swarms of robots using these functions and illustrate.

Disciplines
Engineering

Comments

Suggested Citation:

Chaimowicz, L., N. Michael and V. Kumar. (2005). Controlling Swarms of Robots Using Interpolated Implicit Functions. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation.* Barcelona, Spain. April 2005.

Proceedings of the 2005 IEEE
International Conference on Robotics and Automation
Barcelona, Spain, April 2005

Controlling Swarms of Robots
Using Interpolated Implicit Functions

Luiz Chaimowicz, Nathan Michael and Vijay Kumar

GRASP Laboratory – University of Pennsylvania

Philadelphia – PA – USA

{*chaimo, nmichael, kumar*}*@grasp.upenn.edu*

Abstract— **We address the synthesis of controllers for large groups of robots and sensors, tackling the specific problem of controlling a swarm of robots to generate patterns specified by implicit functions of the form** $s(x, y) = 0$**. We derive decentralized controllers that allow the robots to converge to a given curve** S **and spread along this curve. We consider implicit functions that are weighted sums of radial basis functions created by interpolating from a set of constraint points, which give us a high degree of control over the desired 2D curves. We describe the generation of simple plans for swarms of robots using these functions and illustrate our approach through simulations and real experiments.**

I. INTRODUCTION

Large groups of robots and sensors, generically called *swarms of robots*, have received much attention in recent years. Basically, these systems try to employ a large number of simpler agents to perform different types of tasks, oftentimes inspired by their biological counterparts.

We are particularly interested in planning and controlling the trajectories of swarms in dynamic, resource-constrained, adversarial environments. One of the main concerns of these types of tasks is scalability. Systems to control and coordinate swarms of robots must be scalable from tens to hundreds of agents and must be robust to the dynamic deletion or addition of new agents. Agents should operate asynchronously and rely only on local sensing and communication, since the maintenance of a global state of the system is impractical. Furthermore, robots must be *anonymous* due to the challenges of uniquely identifying individual members within the swarm.

In this paper we present a scalable approach that allows a swarm of robots to synthesize shapes and patterns, converging and spreading along complex 2D curves. We specify an implicit function and control the robots to track this function using a modified gradient descent technique. These implicit functions are generated interpolating from several user-selected constraint points, which provide a great flexibility in generating different shapes for the swarm. We illustrate this approach through simulations and real experiments using a group of ER1 robots.

We believe that these controllers can be used in applications such as perimeter surveillance, multi-robot and sensor deployment, environmental protection, among others. Also, the approach presented here is a first step towards having a scalable methodology for the motion planning of robot swarms.

This paper is organized as follows: Section II presents some related work in motion planning and control for large groups of robots. Section III presents our approach, describing the controllers and the implicit function generation process. Simulations and real experiments are presented in sections IV and V respectively. Finally, Section VI brings the conclusion and directions for future work.

II. RELATED WORK

The general area of motion planning for large groups of robots has been very active in the last few years. One of the first works to deal with the motion control of a large number of agents was proposed for generating realistic computer animations of flocks of birds (called *boids*) [1]. A detailed analysis of the stability of such flocking behaviors and their robustness to changes in nearest neighbors is presented in [2]. In the robotics community, the more classical approaches for planning the motion of groups of robots have generally been divided into centralized and decoupled [3]. Centralized planning consists of planning for the entire group, considering a composite configuration space. It normally leads to complete solutions but becomes impractical as the number of robots increases due to the high dimensionality of the joint configuration space. On the other hand, decoupled approaches plan for each robot individually and later try to deal with the interactions among the trajectories. This reduces the dimensionality of the problem, but can result in a loss of completeness. Another way of avoiding the dimensionality problem is to treat groups of robots as a single entity with a smaller number of degrees of freedom and then perform the motion planning for this entity. The work presented in [4], for example, models a group with a deformable shape and uses a *Probabilistic Roadmap* to plan for this shape. Another approach is presented in [5] where robots can be dynamically grouped together in a hierarchical manner using a sphere tree structure.

In our previous work [6], we derived abstractions for groups of robots and decentralized controllers that allow the motion planning problem for the abstract group to be solved in a lower dimensional space. We showed how groups of robots can be modeled as deformable ellipses, and presented decentralized controllers that allowed the control of the shape and position of the ellipses. In [7], we extended this approach building a hierarchy of ground and air vehicles and allowing groups to split and merge.

Instead of performing motion planning in the traditional sense, some works focus on specific types of motions for swarms of robots. For example, algorithms for dispersing a swarm [8], [9], moving in formation [10], covering areas while maintaining constraints [11] and to perform shepherding behaviors [12]. Similarly, our main objective in this paper is not to plan a complete trajectory for the group in the traditional motion planning sense, but to navigate a large group of robots through some locations where they have to organize themselves into certain shapes. As mentioned, we present a simple yet robust approach where large groups of robots perform a gradient descent and spread along the zero isocontour of a predetermined function.

Gradient descent techniques for smaller groups of robots have been studied, for example in [13] and [14], but without considering the requirements of synthesizing specific shapes. On the other hand, an interesting approach is presented in [15], where a group of robots track the boundaries of an environmental function using active contour models (*snakes*). In this case, the robots are able to converge to certain shapes (according to the tracked gradient), but the control equations are more complex than a simple gradient descent and may require a high level of inter-robot communication.

III. ROBOT CONTROL USING IMPLICIT FUNCTIONS

A. Controllers

We want our robots to converge to and spread along a 2D curve S given by an implicit function $s(x, y) = 0$. This implicit function can be viewed as the zero isocontour of a 3D surface $f = s(x, y)$ whose value is less than zero for all points (x, y) that are inside the S boundary and is greater than zero for all points outside the S boundary as shown in Figure 1.

Fig. 1. 3D plot of the function: $f(x, y) = d^2 log(d)$ where $d = \sqrt{(x - 10)^2 + (y - 10)^2}$. The zero isocontour is depicted in white.

By controlling individual robots to perform a gradient descent on function f, we are able to make the group converge to S. We should invert the direction of the gradient when $f < 0$, so the robots will converge to the zero

isocontour and not to the function minima. Alternatively, the robots can descend the gradient of the square of the function (∇f^2), that has the zero isocontour as its minima. This removes the discontinuity that arises when inverting the direction of the gradient. Robots are also subject to simulated collision forces F_c due to interactions with other robots. These forces are computed using a simplified rigid body dynamic model based on [16].

To spread themselves along the curve, the robots apply repulsive forces F_r to their neighbors when they are close to S. This force is inversely proportional to the distance between the robots and is only active when this distance is less than a certain threshold, that is basically the robot sensing range.

More specifically, we consider a fully actuated robot i with dynamic model given by:

$$\dot{\mathbf{q}}_i = \mathbf{v}_i, \tag{1}$$
$$\dot{\mathbf{v}}_i = \mathbf{u}_i, \tag{2}$$

where $\mathbf{q}_i = [x_i, y_i]^T$ is the configuration of robot i, $\mathbf{u}_i$ is its control input and $\mathbf{v}_i$ is the velocity vector. The control law is given by:

$$\mathbf{u}_i = -k\nabla f^2(\mathbf{q}_i) - C\dot{\mathbf{q}}_i + \sum_{j \in N_i} F_r(\mathbf{q}_i, \mathbf{q}_j) + \sum_{j \in C_i} F_c(\mathbf{q}_i, \mathbf{q}_j). \tag{3}$$

Constant k is positive ($k > 0$) and N_i and C_i are the sets of neighboring robots that are applying respectively repulsive (spreading) and collision forces to robot i.

Note that this controller can cope with the addition or deletion of new members and is scalable to a large number of robots. Each robot requires its local state feedback and local sensing (to detect close neighbors) but no global state information is required. Also, each robot must know the shape of the function in order to compute its gradient $\nabla f(\mathbf{q}_i)$. We accomplish this by using specific types of functions (explained in Section III-C) and broadcasting the function parameters upon initialization.

B. Performance

We want to show that the controller described in the previous section will drive the robots to curve S and make them spread along the curve.

Let's first consider Equation (3) without the repulsive forces term F_r and the collision forces term F_c. The system of robots under this control law is in equilibrium when both $\mathbf{v}_i = 0$ and $\mathbf{u}_i = 0$:

$$\mathbf{v}_i = 0 \Rightarrow \dot{\mathbf{q}}_i = 0$$
$$\mathbf{u}_i = 0 \Rightarrow \nabla f^2(\mathbf{q}_i) = 0 \Rightarrow f(\mathbf{q}_i)\nabla f(\mathbf{q}_i) = 0.$$

We want to minimize the error associated with the convergence to function f, that is given by:

$$E(\mathbf{q}) = \frac{1}{2} \sum_i f^2(\mathbf{q}_i). \tag{4}$$

Thus, for the function $E(\mathbf{q})$ to be at a minimum we need:

$$\sum_i f(\mathbf{q}_i)\nabla f(\mathbf{q}_i)\dot{\mathbf{q}}_i = 0. \tag{5}$$

It is easy to see that when the robots are in equilibrium, the error associated with the function f is also at a local minimum. Hence, the condition for equilibrium satisfies the necessary condition for minimizing the error associated with the convergence to the function f.

Now, to establish convergence, consider the Lyapunov function $V(\mathbf{q}, \dot{\mathbf{q}})$ given by:

$$V(\mathbf{q}) = k\phi(\mathbf{q}) + \frac{1}{2}\sum_i \dot{\mathbf{q}}_i^2, \qquad (6)$$

where ϕ is the potential energy associated with the group of robots defined by:

$$\phi(\mathbf{q}) = \sum_i f^2(\mathbf{q}_i). \qquad (7)$$

It is not difficult to show that V is a monotonically decreasing function and the system is asymptotically stable:

$$\begin{aligned}
\dot{V}(\mathbf{q}) &= k\sum_i 2f(\mathbf{q}_i)\nabla f(\mathbf{q}_i)\dot{\mathbf{q}}_i + \sum_i \dot{\mathbf{q}}_i \ddot{\mathbf{q}}_i \\
&= k\sum_i 2f(\mathbf{q}_i)\nabla f(\mathbf{q}_i)\dot{\mathbf{q}}_i - \sum_i \dot{\mathbf{q}}_i(k\nabla f^2(\mathbf{q}_i) + C\dot{\mathbf{q}}_i) \\
&= -\sum_i C\dot{\mathbf{q}}_i^2.
\end{aligned}$$
$$(8)$$

The addition of collision forces F_c only adds a dissipative term, so the above result does not change. The final step is to show that the addition of the repulsive forces F_r will cause the robots to spread along the curve but will not interfere significantly with the gradient forces, so the robots still converge to S. This is a little trickier since the addition of repulsive forces may create new equilibrium points outside the curve boundary. It is possible to show convergence to an invariant set using the approach in [2], but the set itself is hard to characterize geometrically.

Figure 2 helps to illustrate some of the ideas. As mentioned, the repulsive forces are only active when the robots are close enough to the curve boundary ($f(\mathbf{q}_i) < \varepsilon$) and the distance between the robots is smaller than a threshold ($|\mathbf{q}_i - \mathbf{q}_j| < R$). For example, robots 1, 2, and 3 in Figure 2 do not suffer any repulsive force, and move directly towards S. Robots that are inside ε and close to other robots will be subjected to both repulsive and gradient forces (robot 6 in Figure 2). In this case, setting the constants accordingly, it is possible to configure these forces in a way that the resultant force will drive the robot to the curve while slightly moving away from the other robot. Finally, when the robots are on the curve (robots 4 and 5 for example) only the repulsion forces will be active, causing the robots to spread around it.

Special situations may occur when the number of robots n is too small or too large compared to the curve length. When n is small, we will probably have "holes" in the curve. The robots that are close will spread themselves only until the point where their separation is greater than R, which does not guarantee a complete coverage of the curve. On the other hand, if n is too large, there will not be enough space to accommodate all robots along the curve. This will lead to a situation where robots will compete to move toward the curve. We are currently studying what

Fig. 2. Diagram of the forces acting on the robots close to the curve boundary.

is the ideal relationship among n, R, ε, and ρ (the curve radius) in order guarantee convergence and coverage of the curve. For example, considering Π as the curve perimeter, if $nR \leq \Pi$ and $\rho \gg \varepsilon \gg R$, it is possible to observe experimentally that the invariant set associated with the control law (3) is $f(\mathbf{q}_i) = 0$.

C. Interpolating Implicit Functions

In order to make the robots synthesize specific shapes, we consider f as a weighted sum of radial basis functions (RBFs) created interpolating from a set of constraint points. As explained in [17], a radial basis function $r(\mathbf{p})$ is a function that can be described in terms of a center point $\mathbf{c}$ and a function $h(d)$, where d is the distance between any point $\mathbf{p}$ and the center $\mathbf{c}$. The term radial comes from the fact that $r(\mathbf{p})$ evaluates exactly the same for all the points that are at a fixed radius from $\mathbf{c}$. Figure 1 is an example of a radial basis function where $h(d) = d^2 log(d)$ and $\mathbf{c} = (10, 10)$.

To generate a specific function, we specify some constraint points $\mathbf{p}_j$ along the the desired zero isocontour such that $f(\mathbf{p}_j) = 0$ and at least one constraint inside or outside the boundary (to avoid degenerate solutions). Each of these constraints will be the center of one RBF. Then, solving a simple linear system, we determine the weights (w_j) of all the RBFs that comprise the function f. This approach is commonly used in the computer graphics community for generating computer animations [18].

Thus, function f is given by:

$$f(\mathbf{q}_i) = \sum_j w_j h(|\mathbf{q}_i - \mathbf{p}_j|) \qquad (9)$$

where the term $|\mathbf{q}_i - \mathbf{p}_j|$ is the Euclidean Distance (d) between a robot $\mathbf{q}_i$ and a constraint $\mathbf{p}_j$ and the function h is given by $h(d) = d^2 log(d)$. As an example, figures 3 and 4 show a weighted sum of radial basis functions that has a zero contour forming the letter 'P'. This function was generated interpolating from the constraint points depicted.

One of the main advantages of this approach is that it gives a great flexibility for generating different 2D shapes. Complex curves can be synthesized interpolating from an adequate number of constraint points. Furthermore, the resulting radial basis functions are generally smooth, which allow the use of the gradient descent controllers described in the previous sections.

Fig. 3. Implicit function forming a letter 'P'. The constraint points used to generate the function are depicted: 37 along the boundary ($f(\mathbf{p}_j) = 0$) and 4 internal.

Fig. 4. 3D plot of the interpolated function with the zero isocontour forming the letter 'P' (depicted in white)

Fig. 6. Function composed of 93 radial basis functions that has the string "LUIZ" as its zero isocontour.

Fig. 7. Snapshot of a simulation where 80 robots converge and spread along the isocontour depicted in the previous figure.

IV. SIMULATIONS

To demonstrate our approach, we performed several simulations with different implicit functions and a large number of robots. We used MuRoS, a multirobot simulator developed in the GRASP Lab that allows us to implement various tasks, test different controllers, and observe the robots in real time.

We first generated implicit functions forming the letters G, R, A, S, P and switched among these functions, dynamically changing the robots' plans. Figure 5 shows the function graph and a snapshot of the simulation for each letter, in which the robots are represented by the small circles. Each of these functions was generated using an average of 40 constraint points. As expected the robots are attracted by and spread along the shapes defined by the zero isocontour of the functions. The time taken by the group to converge to the each shape is around 30 seconds.

More complex shapes can also be created using this approach. For example, the function depicted in Figure 6 was created using 93 constraints and has the string "LUIZ" as its zero isocontour. Figure 7 shows a simulation snapshot of 80 robots spreading along this isocontour.

It is important to mention that, depending on the shape of the function, sometimes the robots can be trapped into "plain" regions where the gradient is close to zero. But this problem can be easily solved by adding new constraint points to the function that do not alter the zero isocontour but change the function's overall gradient. Another possibility, that does not involve changing the function, is to make the robots detect when they are trapped ($\nabla f(x, y) \approx 0, f(x, y) \neq 0$) and perform some random movements to escape these regions. Also, if the function has different zero isocontours, the robots may not be able to completely synthetize the desired shape depending on their initial position. Consider, for example, the letter 'P' shown in figures 3 and 4. If all robots start from outside the shape, they will have to overcome the minima imposed by the outer curve to reach the inner ellipsoid of the shape. In the simulations presented here, the initial distribution of the robots is roughly uniform, thus we do not observe this problem. But in general situations, this may be similar to a "local minima" scenario that can be solved using some approaches such as random exploration, etc. The main difference here is that the minima is not an undesired local minima but a valid curve that we also want to converge to. In fact, the objective in this case is divide the group between the two curves. We are currently studying some strategies to systematically perform this type of behavior.

V. EXPERIMENTS

We performed some initial experiments using a group of ER1 robots from Evolution Robotics (Figure 8). The ER1s are non holonomic, thus small changes had to be made to the controllers of equation 3 in order to drive them.

Fig. 5. Simulations of 55 robots tracking different functions to form the letters G, R, A, S, P. The figures on the top row show the functions generated for each letter, while the figures on the bottom row show snapshots of the simulation (robots are the small circles).

Basically, considering $\mathbf{q} = [x, y, \theta]^T$ we have:

$$\dot{\mathbf{q}} = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \quad (10)$$

$$\begin{bmatrix} \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} = \begin{bmatrix} -K_v \nabla f(\mathbf{q}) - C_v v_1 \\ K_w(\psi - \theta) - C_w v_2 \end{bmatrix} \quad (11)$$

where ψ is the direction of the inverse of the gradient.

In the experiments presented here, the robots have no information about their neighbors (no sensing and no communication) so they are not able to repel each other and spread along the zero isocontour as shown in the simulations. Also, they use only odometry to estimate their position and compute the function gradient. Odometry can lead to some errors in long runs but worked fine in these experiments. The robots were programmed and tasked using ROCI [19], a robotic programming framework that is being developed in the GRASP Lab.

Fig. 8. Group of 10 ER1 robots at the GRASP Lab.

In these experiments, six robots follow two different functions depicted in Figure 9. The first function has an ellipse-like shape as its zero isocontour and was generated using 7 constraint points. The second function was created using 14 constraints and has two separated smaller ellipses

as its zero isocontour. Figure 10 shows the robot trajectories, with the initial positions marked with a small circle and the zero isocontour of the two functions displayed as dotted lines. Initially, the robots track the first function converging to the ellipse-like shape in the center of the graph. Then, the function is dynamically changed (by broadcasting new function parameters) and the robots split in two small groups, descending the gradient of the second function and converging to the smaller ellipses. Figure 10 also shows two pictures of the robots at the zero isocontours of the two functions.

Fig. 9. Two functions used to drive the robots in the experiments. The function depicted on the top has a single 2D curve $S(x, y) = 0$, while the function on the botton has two curves, which causes the robot group to split.

These experiments demonstrate that the robots are capable of smoothly tracking different implicit functions that were generated interpolating from sets of constraint points.

Fig. 10. Trajectory performed by the robots: they first converge to the 2D curve in the center and then the group splits in two and converge to the top and bottom curves. The small circles indicate the initial position of each robot.

Also, we could observe that the group autonomously split in two subgroups in order to follow the new function with two minima. No specific command was given to individual members: the group simply descended the gradient of the new function which caused it to split. We are currently investigating ways of extending this approach to plan more complex trajectories, where large groups of robots would be able to split, merge and change shape while navigating in dynamic, cluttered environments.

VI. Conclusion

This paper presented a scalable approach for controlling large groups of robots to synthesize specific shapes given by implicit functions. These functions are generated interpolating from a set of constraint points along the desired curve, which gives a great flexibility for generating complex shapes. Using a gradient descent technique augmented with robot to robot repulsion, we were able to control large groups of robots to track these functions and spread along their zero isocontour using only state feedback and local sensing information. Both simulations and real experiments were used to validate this approach.

Our future work is directed toward several fronts. We are currently working on obtaining formal guarantees of performance for the controllers considering the repulsive forces. On a more experimental perspective, we are equipping our robots with infrared sensors that will provide local sensing capabilities allowing them to repulse each other during the gradient descent. Finally, we want to extend this approach to automatically plan more sophisticated trajectories for the swarm. These trajectories would take into account spatial constraints and perform dynamic changes in the group's pose and shape, allowing swarms of robots to plan and navigate through cluttered environments in a scalable way.

References

[1] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics (SIGGRAPH 87)*. ACM Press, 1987, pp. 25–34.

[2] H. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," July 2003, submitted to Automatica.

[3] J.-C. Latombe, *Robot Motion Planning*. Kuwer Academic Publishers, 1991.

[4] A. Kamphuis and M. Overmars, "Motion planning for coherent groups of entities," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, Louisiana, Apr 2004, pp. 3815–3821.

[5] T. Y. Li and H. C. Chou, "Motion planning for a crowd of robots," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, 2003, pp. 4215–4221.

[6] C. Belta, G. A. S. Pereira, and V. Kumar, "Abstraction and control for swarms of robots," in *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*, 2003.

[7] L. Chaimowicz and V. Kumar, "Aerial shepherds: Coordination among uavs and swarms of robots," in *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems (DARS 2004)*, 2004, pp. 231–240.

[8] T.-R. Hsiang, E. M. Arkin, M. A. Bender, S. P. Fekete, and J. S. B. Mitchell, "Algorithms for rapidly dispersing robot swarms in unknown environments," in *Algorithmic Foundations of Robotics V*, ser. Springer Tracts in Advanced Robotics, J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, Eds. Springer-Verlag, 2003, vol. 7, pp. 77–94.

[9] A. Howard, M. Mataric, and G. Sukhatme, "Mobile sensor network deployment using potential fields: A distributd, scalable solution to the area coverage problem," in *Proceedings of the 6th International Symposium on Distributed Autonomous Robotic Systems (DARS 2002)*, 2002.

[10] T. Balch and M. Hybinette, "Social potentials for scalable multi-robot formations," in *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, 2000, pp. 73–80.

[11] S. Poduri and G. S. Sukhatme, "Constrained coverage for mobile sensor networks," in *IEEE International Conference on Robotics and Automation*, New Orleans, LA, May 2004, pp. 165–172.

[12] J.-M. Lien, O. Bayazit, R. Sowell, S. Rodriguez, and N. Amato, "Shepherding behaviors," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, 2004, pp. 4159–4164.

[13] R. Bachmayer and N. E. Leonard, "Vehicle networks for gradient descent in a sampled environment," in *Proceedings of the 41st IEEE Conference on Decision and Control (CDC-02)*, 2002, p. 112 117.

[14] V. Gazi and K. Passino, "Stability analysis of social foraging swarms," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 34, no. 1, pp. 539–557, 2004.

[15] D. Marthaler and B. A. L., "Collective motion algorithms for determining environmental boundaries," in *SIAM Conference on Applications of Dynamical Systems*, May 2003.

[16] P. Song, P. Kraus, V. Kumar, and P. Dupont, "Analysis of rigid body dynamic models for simulation of systems with frictional contacts," *ASME Journal of Applied Mechanics*, vol. 68, no. 1, pp. 118–128, April 2001.

[17] G. Turk, H. Q. Dinh, J. F. O'Brien, and G. Yngve, "Implicit surfaces that interpolate," in *Proceedings of the International Conference on Shape Modeling & Applications*. IEEE Computer Society, 2001, pp. 62–73.

[18] G. Turk and J. F. O'Brien, "Shape transformation using variational implicit functions," in *Proceedings of the 26th annual conference on Computer graphics (SIGGRAPH 99)*. ACM Press/Addison-Wesley Publishing Co., 1999, pp. 335–342.

[19] L. Chaimowicz, A. Cowley, V. Sabella, and C. J. Taylor, "ROCI: A distributed framework for multi-robot perception and control," in *Proceedings of the 2003 IEEE/RJS International Conference on Intelligent Robots and Systems*, 2003, pp. 266–271.