RiverSwarm: Topology-Aware Distributed Planning for Obstacle Encirclement in Connected Robotic Swarms

Pradipta Ghosh[1], Jie Gao[2], Andrea Gasparri[3], and Bhaskar Krishnamachari[1]

[1]Ming Hsieh Department of Electrical Engineering, University of Southern California, {pradiptg, bkrishna}@usc.edu
[2]Computer Science, Stony Brook University, jgao@cs.stonybrook.edu
[3]Dipartimento di Ingegneria Università degli studi "Roma Tre", gasparri@dia.uniroma3.it

***Abstract*—Distributed motion control of robotic swarms has been receiving increased attention due to their potential for application in many domains including emergency response and remote sensing and exploration. A challenging aspect of motion control for swarms is enabling them to move past large obstacles without losing global connectivity. In this paper we present a novel motion primitive for swarms of robots which allows them to flow past large obstacles while remaining connected. This technique relies on a key result from differential geometry, the Gauss-Bonnet theorem, which allows tracking and counting the number of holes in a given triangulated graph in a distributed manner.**

I. Introduction

Swarm robotics can be defined as the study of how a group of relatively simple, low-cost robots can be constructed to collectively accomplish tasks that are beyond the capabilities of a single one while providing a higher robustness and flexibility. The idea is that the collective behavior should emerge from the local interaction between neighboring robots. This is inspired by the typical social behavior among insects like ants, termites, wasps, bees, birds etc. Briefly speaking, the main aspect of a robotic swarm are: (1) Robustness that help the system to continue functioning properly even some of its element have failed, (2) Flexibility to assign many different tasks from a broad range, and (3) Scalability to incorporate any number of elements. Robotic swarms can be used for several applications ranging from remote exploration, agricultural foraging, to search and rescue operations. The reader is referred to [1], [2] for a complete overview on swarm robotics.

In this work, we are interested in studying the collective behavior of a robotic swarm moving in a cluttered environment. In particular, we focus on the problem of moving past large obstacles without losing the global connectivity of the network topology. Indeed, the capability to preserve the network connectivity over time is a crucial point in the context of robotic swarms for which the capability to exchange information is vital to perform almost any collaborative task.

*This work has been partially supported by the Italian grant FIRB "Futuro in Ricerca", project NECTAR, code RBFR08QWUV, funded by the Italian Ministry of Research and Education (MIUR).

*This work has been partially supported by the US National Science Foundation via awards CNS-1217260, CNS-1017881.

*Jie Gao would like to acknowledge the support of NSF through grants DMS-1221339, CNS-1217823 and CNS-1116881.

The proposed motion primitive, which we denote as RiverSwarm, relies on prior work for distributed motion using potentials [3] and flexible connectivity maintenance using algebraic connectivity control [4]. The core novel idea in RiverSwarm is to leverage a fundamental theorem of differential geometry, the Gauss-Bonnet theorem [5], to detect when the swarm has surrounded an obstacle and initiate a cut at the rear which allows the swarm to continue moving. We show that this approach allows connected swarms to flow past large obstacles under many conditions where state of the art approaches may suffer from many difficulties like disconnection, getting stuck at the obstacle etc. In Section III, we have presented a detailed discussion of such problems with state of art techniques.

II. Related Work

A large number of swarm aggregation algorithms can be found in the literature [6], [7]. Among the others, the usage of potential fields represents a very common design technique for coordinating the motion of robotic swarms [8], [9]. The core idea is to control the local interaction between neighboring robots through the design of proper attractive and repulsive artificial potential fields. The resultant force determines the direction and speed of travel for each single unit of the swarm, ultimately yielding to a collective behavior. Connectivity maintenance algorithms in the literature can be classified into *local* or *global* techniques. Local techniques aim at preserving the original set of links defining the network topology over time. An example of a decentralized algorithm for local connectivity maintenance can be found in [10], and references therein. Notably, the fact of preserving each link of the initial network topology significantly reduces the capability of the robotic swarm. Global techniques attempt to overcome this limitation by allowing initial (redundant) links to be removed under the assumption the overall network topology remains connected. In this regards, the estimation and control of the algebraic connectivity originally introduced in [11] represents a very effective approach. Extensions to this framework range from the integration of additional (bounded) control terms [4] to the saturation of the connectivity control term itself [12].

On the other hand, maintaining a triangulation of nodes under motion is a common task for mobile networks and robot swarms. Delaunay triangulation, for its many nice properties e.g Convex hull, Empty circle property, Minimum Spanning

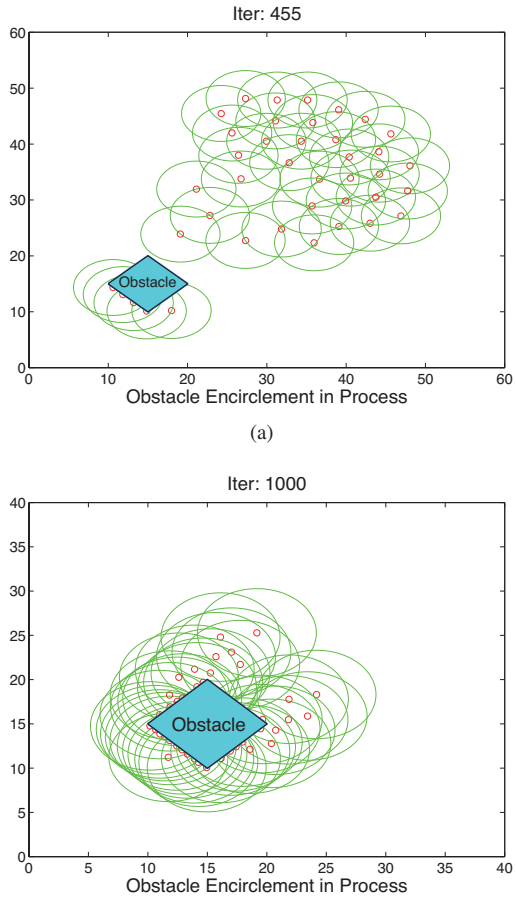


Fig. 1: Typical behavior of swarm aggregation algorithms a) without connectivity maintenance b) with connectivity maintenance

Tree [13], Minimal roughness property[14] etc., has been the one mostly used. In a network of robots using wireless communication, edges longer than the communication range of the robots are unavailable. Thus a restricted version of the Delaunay triangulation, using only the edges shorter than 1, was introduced in [15]. It was also shown in the same paper that the restricted Delaunay graph can be locally computed – each node computes the local Delaunay graph on its immediate neighbors, by sharing this local Delaunay graph with only one hop neighbors and removing the edges that are inconsistent, the result is a globally consistent planar graph including all the short Delaunay edges. It turns out that the restricted Delauany graph can be computed and maintained on mobile nodes without location information making it easier to implement in practice. It is shown in [16] that one can locally decide all the edges of a restricted Delaunay triangulation if each node knows the angles of its edges. This can be relaxed even further. If each node knows only the *angular ordering* of its edges, we can decide and maintain a valid (albeit non-Delaunay) triangulation. For our application this will suffice. Triangulations of a wireless mobile network and its curvature have been used in previous work to generate virtual coordinates for greedy routing [5]. We note that the use of the Gauss-Bonnet Theorem for tracking network topology in robot swarms is new.

III. Challenges for State of the Art

The state of art techniques in the field of Swarm robotics for moving around obstacles suffer from two major limitations:

1) **Disconnection:** In face of an obstacle most of the existing methods that do not include connectivity control fail to allow the robots to move while maintaining connectivity. Moving around an obstacle could cause disruption in global connectivity even though every robot might have local connectivity to some other robots, causing partitioning of the network. Figure 4a illustrates the problem where it can be seen that some robots got stuck on the obstacle wall while the rest of them moved away, resulting in network disconnection.
2) **Inefficiency in movement:** A common limitation of existing approaches able to preserve connectivity is that the overall motion capability of the swarm is drastically reduced if any of the robots get stuck due to an obstacle. They might be stuck forever or they will try to move through one side of the obstacle even though there may be two ways around it when the obstacle is moderately large with respect to the size of swarm. From figure 4b it is clear that the entire swarm is stuck due to connectivity constraints. The only option for this case is to try and force the entire swarm through one side of the obstacle, which can be a slow process.

Note that it may be possible to solve this in part by moving the swarm collectively in a different / random direction temporarily then reverting back to the original control. However this approach is undesirable as it offers no guarantees and can be potentially very slow in some cases.

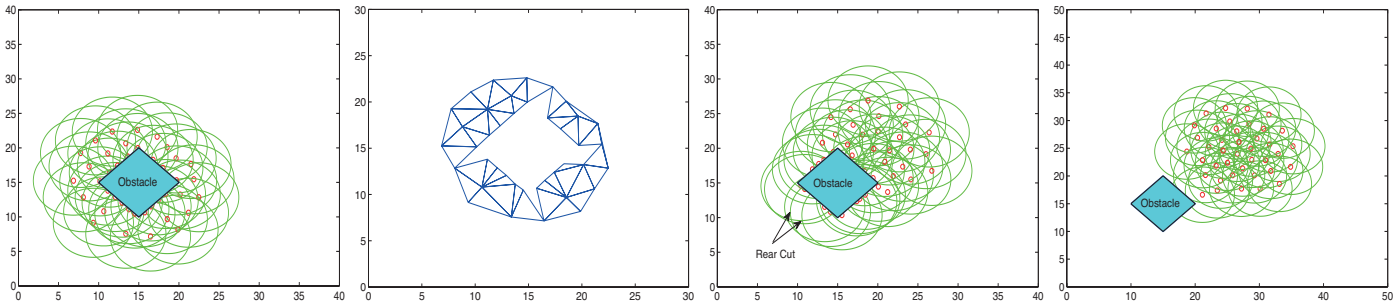
In this paper we propose a solution for a scenario where the swarm as a whole is sufficiently large to "flow" around the obstacle.

IV. RiverSwarm

The proposed mechanism has three key components: 1) Initiating encirclement of the obstacle , 2) Determining that the obstacle has been encircled, i.e. that there is a hole in the connected network and 3) Disconnecting from the rear (boundary of the obstacle that blocks movement) of the obstacle to allow the swarm to proceed further.

A. Initiating Encirclement

Whenever a swarm of robots encounters an obstacle in its path, at-least one of the robots' movement gets restricted. In our proposed method, instead of just restricting the motion, we make the robot completely stationary at this point. Then the connectivity control, attraction/ repulsion between each element comes into play. Under influences of all these forces, some robots move towards obstacle and also get stuck near the obstacle boundary.This way we will be able to identify the boundary of the obstacle that blocks movement. The agents moving parallel to the obstacle will also get stuck due to the connectivity constraint. The connectivity control is



(a) Hole Encirclement (b) Triangulation & Hole Detection (c) Initiate Detachment (d) Detachment Complete

Fig. 2: Steps of our Riverswarm Algorithm

based on Algebraic Connectivity which is the second smallest eigenvalue of the Graph Laplacian [4]. But to force the swarm to encircle the obstacle we need to add another potential function. For solely that purpose, for a 2D environment we have identified two fronts of the swarm in the two mutually perpendicular directions which are at $\frac{\pi}{4}$ w.r.t to the destination direction, say (x_1, y_1) and (x_2, y_2) , and added a potential towards the co-ordinate $((x_1+x_2)/2, (y_1+y_2)/2)$ proportional to the distance between the robot and calculated point. We assume that all the robots share a common reference frame. This potential will add a attraction force towards encircling the obstacle. We assume that the localization of each robot is perfect. So at each moment the exact locations of each robots in the swarm are known. Also, by using connectivity controller in our method we ensure that there are no loss of communications between the robots.

In summary, there are four major potential fields that contribute to the movements of the robots: a fixed velocity/potential towards goal, attraction-repulsion potential between neighboring robots[4] , connectivity controlling potential [4] and the novel encirclement potential discussed earlier. Figure 2a presents a simulated instance of the encirclement of the obstacle.

B. Detecting Encirclement

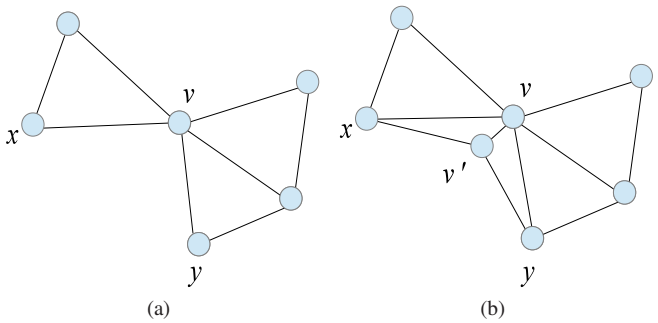
The Gauss–Bonnet theorem[5] is an elegant way to connect geometry, specifically, curvature, to its Euler characteristics. According to the theorem, given $\mathcal{H} = (\#\ of\ handles)$ and $\mathcal{Z} = (\#\ of\ holes)$, the total curvature of a surface M is a topological invariant: $\sum_{v_i \epsilon V} K_i = 2\pi\chi(M)$ where $\chi = 2 - 2 * \mathcal{H} - \mathcal{Z}$. On the other hand, the Delaunay triangulation is a method for triangulating the area covered by a set of points. The most important property is the "empty-circle" property that means that there should be no other points inside a Delaunay triangle except the vertices. We are interested in the restricted version of Delaunay triangulation with two restriction. Firstly, the length of the edges should be less than some threshold (Communication range of the robots) and secondly, there should not be any edge that intersects the obstacle region. In [16], Bruck, Gao and Jiang presented the method for the Delaunay triangulation in a practical network containing multiple nodes.

Remark that the Gauss-Bonnet Theorem [5] applies for any triangulation. With ease of implementation in mind, the triangulation we compute is only a "topological" triangulation, i.e., with no knowledge of the edge length. We artificially give each edge weight of 1, i.e, all triangles are equilateral. That means all the angles of the triangles are $\pi/3$. We call an edge a boundary edge if it is not in two triangles. We call a vertex to be an interior vertex if all its adjacent edges are interior. Otherwise, we call it a boundary vertex.

For each interior vertex u, define its Gaussian curvature as $2\pi - \sum_i \theta_i$, where θ_i is corner angle of the triangle adjacent to u. For each boundary vertex u, define its Gaussian curvature as $\pi - \sum_i \theta_i$, where θ_i is the corner angle of the triangle adjacent to u.

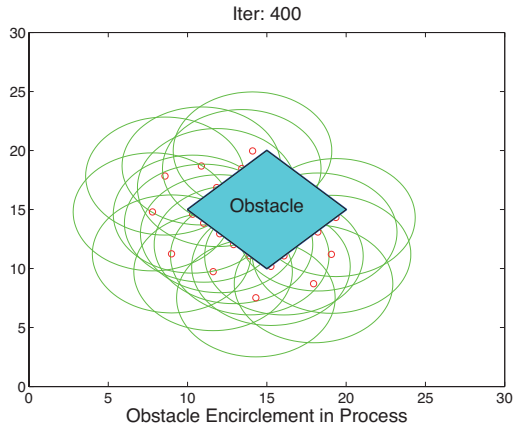
Take the sum of all curvatures for all vertexes. The sum is $2\pi(2-h)$, where h is the number of holes. When there is exactly one hole, this becomes 2π. One important point to note is that this formation will treat the outside surrounding of the network also as a large hole. Figure 2b shows the triangulation for detecting the hole.

The Gauss-Bonnet theorem requires the triangulation to be a manifold. Thus one vertex can only stay on one hole boundary. There might be situations where one vertex may be in more than one hole boundary. To fix this, we create a dummy vertex for each extra hole $--$ make the pinch node/vertex a duplicate copy and connect an edge between the two.In the calculation of the curvature, we are adding an extra $-2\pi/3$ at vertex v, an extra $-\pi/3$ at vertex x and vertex y each, an extra $\pi - 2\pi/3$ at vertex v'. This gives a total of extra $-\pi$ for the curvature. Figure 3 displays the concept.

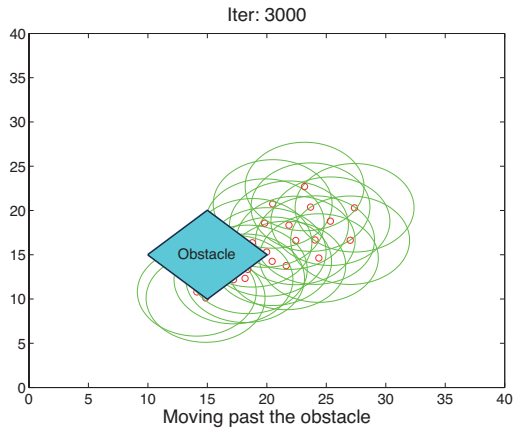


(a) (b)

Fig. 3: Solving the 'pinch node' problem



(a)



(b)

Fig. 4: Riverswarm Algorithm in case of large obstacles

C. Continuing the flow

After the encirclement of a obstacle and hole detection is complete, the robot that first encountered the obstacle initiates a cut and force all its neighbors to move. Also we place some dummy robots in the places of the robots that are stuck and apply a potential in the stopped robots that pushes them away from the respective dummy robots. The direction of the movements are taken towards either left or right to the direction of the repulsive force from the dummies. So now, all the robots that were stopped due to obstacles gets some velocity to move. We let the rest of the robots of the swarm move freely with the connectivity restriction. The robots in front of the obstacle will pull the robots behind the obstacle and continue movement of the swarm towards original goal. Figure 2c displays the starting of the detachment and figure 2d presents a situation after the detachment is complete and the swarm has moved past the obstacle.

Since practical world obstacle sizes are not constrained by the size of the swarm, there will be cases where the obstacle size is really large compared to the swarm. In those cases encirclement is impossible. In such situations, our algorithm tries to move past it by pulling all the robots through one side of the obstacle. In our algorithm, we don't need to switch to any other policy for this to work. It has been automatically taken care of by the acting potentials. Figure 4 clearly shows that even in case of large obstacle, our algorithm is able to make the swarm move past it.

V. Conclusion and Future Work

We have presented a novel idea that enables a swarm of robots to encircle an obstacle and then move past it. There is still much left to do with our proposed method, including fleshing out the full details of a distributed control law with provable theoretical guarantees, and detailed evaluation via simulations and real testbed deployment. We are also working on dealing with simultaneous detection of multiple obstacles to enable the swarm to move rapidly through all possible paths in between them using the same topology-based approach proposed here.

References

[1] Erol Aahin and Alan Winfield. Special issue on swarm robotics. *Swarm Intelligence*, 2(2-4):69–72, 2008.

[2] V. Gazi and K.M. Passino. *Swarm Stability and Optimization*. Springer, 2011.

[3] Veysel Gazi. Swarm aggregations using artificial potentials and sliding-mode control. *Robotics, IEEE Transactions on*, 21(6):1208–1214, 2005.

[4] L. Sabattini, C. Secchi, N. Chopra, and A. Gasparri. Distributed control of multirobot systems with global connectivity maintenance. *Robotics, IEEE Transactions on*, 29(5):1326–1332, Oct 2013.

[5] Rik Sarkar, Xiaotian Yin, Jie Gao, Feng Luo, and Xianfeng David Gu. Greedy routing with guaranteed delivery using ricci flows. In *Proc. of the 8th International Symposium on Information Processing in Sensor Networks (IPSN'09)*, pages 97–108, April 2009.

[6] Veysel Gazi and Kevin M Passino. A class of attractions/repulsion functions for stable swarm aggregations. *International Journal of Control*, 77(18):1567–1579, 2004.

[7] A. Leccese, A. Gasparri, A. Priolo, G. Oriolo, and G. Ulivi. A swarm aggregation algorithm based on local interaction with actuator saturations and integrated obstacle avoidance. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2013.

[8] Naomi Ehrich Leonard and Edward Fiorelli. Virtual leaders, artificial potentials and coordinated control of groups. In *IEEE Conference on Decision and Control*, pages 2968–2973 vol.3, 2001.

[9] Ryan K Williams and Gaurav S Sukhatme. Constrained Interaction and Coordination in Proximity-Limited Multiagent Systems. *IEEE Transactions on Robotics*, 29(4):930–944.

[10] D.V. Dimarogonas and K.J. Kyriakopoulos. Connectedness Preserving Distributed Swarm Aggregation for Multiple Kinematic Robots. *IEEE Transactions on Robotics*, 24(5):1213–1223, 2008.

[11] Peng Yang, Randy A Freeman, Geoffrey J Gordon, Kevin M Lynch, Siddhartha S Srinivasa, and Rahul Sukthankar. Decentralized estimation and control of graph connectivity for mobile sensor networks. *Automatica*, 46(2):390–396, 2010.

[12] A. Gasparri, A. Leccese, L. Sabattini, and G. Ulivi. Collective control objective and connectivity preservation for multi-robot systems with bounded input. In *IEEE American Control Conference (ACC), 2014*, June 2014. To appear.

[13] Jörg Liebeherr and Michael Nahas. Application-layer multicast with delaunay triangulations. In *Global Telecommunications Conference, 2001. GLOBECOM'01. IEEE*, volume 3, pages 1651–1655. IEEE, 2001.

[14] Samuel Rippa. Minimal roughness property of the delaunay triangulation. *Computer Aided Geometric Design*, 7(6):489–497, 1990.

[15] Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, and An Zhu. Geometric spanners for routing in mobile networks. *IEEE Journal on Selected Areas in Communications Special issue on Wireless Ad Hoc Networks*, 23(1):174–185, 2005.

[16] Jehoshua Bruck, Jie Gao, and Anxiao Andrew Jiang. Localization and routing in sensor networks by local angle information. *ACM Transactions on Sensor Networks (TOSN)*, 5(1):7, 2009.