Multi-Robot Path Following with Visual Connectivity

Magnus Lindhé, Tamas Keviczky and Karl Henrik Johansson

***Abstract*— We consider a group of N robots moving through an obstacle field, where only robots that have a clear line of sight can communicate. When passing the obstacles, the group must coordinate its motion to remain connected. We propose using the path–velocity decomposition: Given obstacle-free paths that fulfill a higher-level goal, we propose a method to coordinate the robot motions along the paths so visual connectivity is maintained. The problem is shown to be equivalent to finding a path through an N-dimensional configuration space, avoiding unconnected configurations. We solve this problem with a rapidly exploring random tree algorithm and demonstrate by simulations how the solution time varies with the obstacle density.**

I. Introduction

Using multiple mobile sensors has many advantages. A system of multiple sensors is reconfigurable, more robust to failures and can provide broader coverage than a single sensor. This comes at the price of higher system complexity and the need to maintain communication between sensors to allow coordination. The latter requires communication-aware motion planning: the trajectories of all sensors must be planned so the sensing goals are fulfilled, subject to constraints on maintaining communication.

This paper considers a problem that can arise in this context: We assume that a number of mobile sensors are moving through an obstacle field, following given paths that are chosen to fulfill a sensing goal. Can the motion of each sensor along its path be planned, so that the group maintains visual connectivity despite the obstacles? Fig. 1 illustrates an example snapshot of seven robots passing through an obstacle field. Each robot (black circle) follows a pre-defined path (blue line) between the obstacles (red polygons). Robots can only communicate along clear lines of sight (green lines), so the group must coordinate its motion to maintain connectivity. This paper presents a method to do this, which could be implemented as a communication-aware layer in the motion planner, ensuring that the group stays connected.

The outline of the paper is as follows: We first give a brief overview of related work and what our contributions are. In Section II, we define models of the robot motion, the terrain and the communication constraints, and use these to formally state the problem. Section III explains how the problem can be represented as a classic robot motion planning problem:

The work by M. Lindhé and K. H. Johansson was supported by the Knut and Alice Wallenberg Foundation and the Swedish Research Council

M. Lindhé and K. H. Johansson are with the ACCESS Linnaeus Centre, School of Electrical Engineering, KTH Royal Institute of Technology, Sweden {lindhe|kallej}@ee.kth.se

T. Keviczky is with the Delft Center for Systems and Control, TU Delft, The Netherlands t.keviczky@tudelft.nl

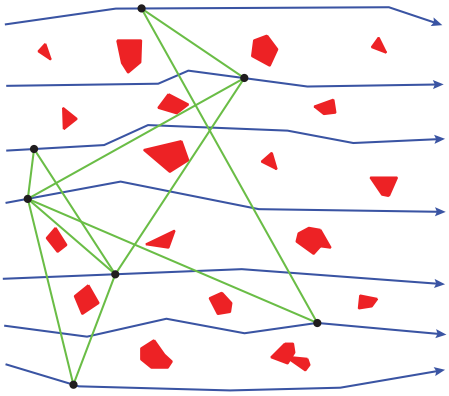

Fig. 1. Seven robots (black circles) moving along pre-planned paths (blue lines) in a field of obstacles (red polygons). Only robots with a clear line of sight (green line) can communicate, but by coordinating their motion, the group can traverse the paths while maintaining connectivity.

finding an obstacle-free path through a high-dimensional configuration space. We also motivate why we have chosen to use rapidly exploring random trees (RRT) to solve this problem. Section IV presents the RRT algorithm and then describes the collision detector, here called outage detector. We also comment on the computational complexity of the method. To illustrate the method, Section V shows some simulations and finally we conclude and indicate possible future directions of research in Section VI.

A. Related Work

In indoor and urban environments, visibility communication models are useful not only for optical communication but also for high-bandwidth microwave radios. Signals in the popular 2.4 GHz band exhibit strong attenuation when going through obstacles and if there is no direct line of sight between transmitter and receiver, multipath fading may also degrade the signal strength. This suggests that if two robots have a clear line of sight between each other, it is reasonable to assume that they can communicate. Problems such as autonomous router placement [1], swarming [2], rendezvous [3] and sensor deployment [4] have been studied under such visibility constraints. The constraints can also be relaxed to requiring connectivity only at certain time instances [5], [6].

Visibility constraints are interesting not only for communication in the classical sense, but also for sensing the surroundings or non-cooperative evaders with a camera or other optical device. This can be used to formulate problems in searching [7], [8], [9], stationary sensing [10], [11] or

tracking [12], [13]. The latter is an example of visual servoing [14] for mobile robotics.

B. Contributions

Our first contribution is to use the path–velocity decomposition for communication-aware motion planning. It was proposed by Kant and Zucker [15] to solve a problem of multi-robot path planning: First paths are planned for all robots, then the velocities along the paths are modified to avoid collisions when paths intersect. In contrast, we assume that the paths have been planned not to intersect, which removes the collision-avoidance constraints on the velocities. An example where this is applicable is when identical robots are used as sensors. If the paths intersect, this can be avoided by exchanging the allocation of path segments between robots. Without the velocity constraints, it is appealing to use this freedom to instead maintain communications. Our second contribution is to propose and analyze an RRT solution to the problem, with an efficient problem-specific collision checker. The result is an algorithm that can solve problems with up to seven robots in practice, using a standard laptop.

II. Preliminaries

In this section, we present models for the motion of the robots, the obstacles and the visibility-constrained communication. Then we formally define the problem of maintaining visual connectivity.

A. Robot and World Model

We consider a group of N robots, where each robot i is given an obstacle-free path $P_i \subset \mathbb{R}^2$, consisting of Π_i straight line segments. Each path P_i is defined by its vertices $\{p_i^1, \dots p_i^{\Pi_i+1}\}$, as illustrated in Fig. 2. When robot i has moved the fraction $x_i \in [0,1]$ of its path length from p_i^1 to $p_i^{\Pi_i+1}$, it has position $r_i(x_i) \in P_i$, where $x = (x_1, \dots, x_N)$ denotes the state of the whole group. To ensure that the paths are collision-free, two different paths may not intersect.

The world contains M obstacles, modeled as (possibly non-convex) polygons. Obstacle $W_k \subset \mathbb{R}^2$ is defined by its Ω_k vertices $\{w_k^1, \dots w_k^{\Omega_k}\}$, also illustrated in Fig. 2. Obstacles may intersect each other, but obstacles and paths may not intersect.

B. Communication Model

Motivated by the ray-like propagation of high-frequency radio signals, we study a visibility-based communication model. We define a connectivity graph $G_c(x) = (V_c, E_c(x))$ with vertices $V_c = \{1, \dots, N\}$. The state-dependent set of edges $E_c(x) \in V_c \times V_c$ consists of undirected links $e = \{i, j\}$ between all robots i, j whose connecting line of sight is not obstructed by any obstacles:

$$\{i,j\} \in E_c(x) \Leftrightarrow \text{convhull}(r_i(x_i), r_j(x_j)) \cap \bigcup_{k \in [1,M]} W_k = \emptyset.$$

At any time instant, the graph is *connected* if there is a path in the graph between any two vertices. Connectivity could also be defined over time intervals, like in delay-tolerant networks [16], but in this paper we only consider instantaneous connectivity. We assume that the initial graph $G_c(\mathbf{0}) = (V_c, E_c(\mathbf{0}))$ is connected, since otherwise the problem is trivially unsolvable.

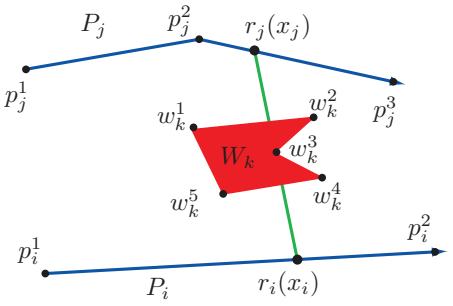

Fig. 2. Robots i and j are at positions $r_i(x_i)$ and $r_j(x_j)$, respectively. Each state x_i denotes the fraction of its path that robot i has moved. Due to the obstacle W_k, the robots have an obstructed line of sight.

C. Problem Formulation

Using the models above, we can now define the problem of path following with continuous connectivity:

Problem 1 Given paths $P_1, \dots, P_N$ and obstacles $W_1, \dots, W_M$, find an end-time $T > 0$ and time-continuous state trajectory $x : [0,T] \to [0,1]^N$ such that $x(T) = \mathbf{1}$ and $G_c(x(t))$ is connected for all $t \in [0,T]$.

Note that this problem is not guaranteed to have a solution. It is easy to construct cases where obstacles make it impossible to maintain connectivity. We will return to this in the next section and comment on how it affects the choice of solution method. But first we will show how the problem maps to a classic motion planning problem, represented in a configuration space.

III. Configuration Space Representation

The system has N degrees of freedom, each corresponding to the position of one robot along its path. We thus define the configuration space of the system as $\mathcal{C} = [0,1]^N$. Configurations $x \in \mathcal{C}$ such that $G_c(x)$ is not connected, are defined to be in *outage*. Obstacles in the world cause regions of $\mathcal{C}$ to be in outage, and these regions are called $\mathcal{C}$-obstacles. The set of all configurations not in outage is called freespace, or $\mathcal{C}_{\text{free}} \subseteq \mathcal{C}$. The problem under consideration is thus equivalent to finding a continuous path $\tilde{x} : [0,1] \to \mathcal{C}_{\text{free}}$ such that $\tilde{x}(0) = \mathbf{0}$ and $\tilde{x}(1) = \mathbf{1}$. This path can then be traversed at any velocity to find the trajectory $x(t)$. This is a standard motion planning problem, for which there exist several solutions. The choice of method depends on, among other things, the geometry of the $\mathcal{C}$-obstacles and the requirements on completeness of the solution.

In its general form, with non-parallel paths and multiple path segments, our problem generates $\mathcal{C}$-obstacles that are non-convex and cannot be described as polygons. Instead, they can be described as semi-algebraic sets.

For problems with this general geometry, the best known complete methods have time complexity that scales exponentially with the number of dimensions [17]. We remind the reader that a complete method is guaranteed to find a solution if it exists or correctly report that the problem is unsolvable. The traditional way to handle poor scalability is to relax the completeness requirement and use sampling-based methods. In this paper, we use the RRT algorithm [18]. It is a probabilistically complete method, which means that if there is a solution, the probability of finding it converges to one as the number of samples increases. But if there is no solution, the solver will run forever. In practice, one can set a timeout for the solver. We will comment more on this in Section VI. In the following section, we describe how the RRT method was used to solve the problem.

IV. Rapidly Exploring Random Tree Solver

By connecting randomly chosen points in $\mathcal{C}_{\text{free}}$, we can construct a tree that eventually contains a path from the start to the goal configurations. The only problem-specific component is the collision detector, which we refer to as the outage detector. It determines if a new sample can be connected by a straight line to the nearest part of the tree without colliding with a $\mathcal{C}$-obstacle. We will present the RRT algorithm and then describe how the outage detector can be implemented. We end this section by discussing the computational complexity of the outage detector.

A. Rapidly Exploring Random Tree

The RRT solver builds a tree graph $G_R(V_R, E_R, Q_R)$, with vertices $V_R = \{1, \ldots, \Gamma\}$ and edges $E_R \in V_R \times V_R$. The set $Q_R = \{q_1, \ldots, q_\Gamma\}$ consists of configurations $q_i \in \mathcal{C}_{\text{free}}$ corresponding to each vertex $v_i \in V_R$. If there is an edge $\{i, j\} \in E_R$, the straight line between q_i and q_j is contained in $\mathcal{C}_{\text{free}}$. This is described in Algorithm 1, taken from [17].

The algorithm iteratively constructs a tree that fills $\mathcal{C}_{\text{free}}$. In each iteration, a random point $y \in \mathcal{C}$ is chosen. The function $\text{NEAREST}(G_R, y)$ returns the index i of the configuration $q_i \in Q_R$ that is closest to y. If y is closer to a point between two configurations q_i, q_j such that $\{i, j\} \in E_R$, than to a configuration q_i, the edge $\{i, j\}$ is split, a new vertex is inserted there and the corresponding configuration is returned. Then the outage detector $\text{FIRST\_OUTAGE}(q_i, y)$ returns a point $\tilde{y}$ on the straight line from q_i to y, as close to y as possible such that the line from q_i to $\tilde{y}$ is contained in $\mathcal{C}_{\text{free}}$. If there is progress, so $\tilde{y} \neq q_i$, we add this new configuration and the corresponding edge to G_R.

Note that Algorithm 1 will run forever, giving an RRT that is arbitrarily close to any point in $\mathcal{C}_{\text{free}}$. To make it terminate in finite time, every 100th iteration, we replace the random y with $y = \mathbf{1}$ and abort if $\tilde{y} = y$. Then the RRT contains a path to the goal. Next, we describe how the outage detector can be implemented.

Algorithm 1 Rapidly Exploring Random Tree [17]

$\Gamma := 1$
$V_R := \{1\}$
$E_R := \emptyset$
$Q_R := \{\mathbf{0}\}$
loop
  Randomly select $y \in \mathcal{C}$
  $i := \text{NEAREST}(G_R, y)$
  $\tilde{y} := \text{FIRST\_OUTAGE}(q_i, y)$
  if $\tilde{y} \neq q_i$ **then**
    $\Gamma := \Gamma + 1$
    $V_R := V_R \cup \Gamma$
    $E_R := E_R \cup \{i, \Gamma\}$
    $Q_R := Q_R \cup \tilde{y}$
  end if
end loop

B. Outage Detector

The outage detector answers the question of how far the system can go from configuration q_i to another one, y, before intersecting a $\mathcal{C}$-obstacle, *i.e.*going into outage. Since under Algorithm 1, $q_i \in Q_R$, the communication graph $G_c(q_i)$ is always connected. Hence, it is sufficient to check how far the system can go towards y before any link $\{i, j\} \in E_c$ is blocked by an obstacle. We update $\tilde{y}$ to this location and search $G_c(\tilde{y})$ for an indirect path between vertices i and j. If it exists, the link $\{i, j\}$ was redundant, so we can continue. The iteration ends if $G_c(\tilde{y})$ is disconnected or we reach y, as summarized in Algorithm 2.

Algorithm 2 $\tilde{y} = \text{FIRST\_OUTAGE}(q_i, y)$

$\tilde{y} := q_i$
while $\tilde{y} \neq y$ and $G_c(\tilde{y})$ is connected **do**
  $\gamma^* := \min_{e \in E_c(\tilde{y}), k \in K(e), n \in (1, \Omega_k)} \gamma(e, w_k^n, \tilde{y}, y)$
  $\tilde{y} := \tilde{y} + (y - \tilde{y}) \min(1, \gamma^*)$
end while

To check $G_c(\tilde{y})$ for connectivity in Algorithm 2, we do a breadth-first search with robot i as the root. Algorithm 2 uses the set

$$K(e) \triangleq \{k : \exists\, x \in \mathcal{C} : \text{convhull}(r_i(x), r_j(x)) \cap W_k \neq \emptyset\},$$

which are the indices of all obstacles that may block the link $e = \{i, j\}$. It also uses the function $\gamma(e, w_k^n, x, y)$, which, for a link $e = \{i, j\}$, is defined as

$$\gamma(e, w_k^n, x, y) \triangleq \min\{g : (1-\lambda)\left[(1-g)r_i(x) + g r_i(y)\right] + \lambda\left[(1-g)r_j(x) + g r_j(y)\right] = w_k^n,\ g \geq 0,\ 0 < \lambda < 1\}.$$

Finding candidate solutions g requires finding the intersections of two rectangular hyperbola or, in degenerate cases, of two straight lines. As illustrated in Fig. 3, $\gamma(\{i, j\}, w_k^n, x, y)$ is the fraction of the straight line from x to y that the system can move before the line of sight between robots i and j intersects the obstacle vertex w_k^n. For simplicity, we assume that both robots i and j move on one single segment of their paths. In the case of multi-segment paths, the computation is done separately over each interval of γ corresponding to different combinations of path segments. If the problem is

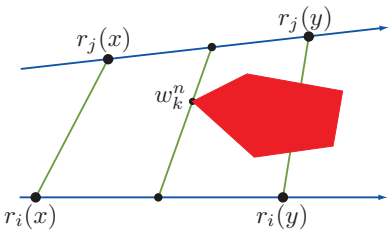

Fig. 3. The core of the outage detector: At what fraction $\gamma(\{i,j\}, u_k^n, x, y)$ of the motion from configuration x to y does the line of sight of robots i and j intersect the obstacle vertex u_k^n?

infeasible, we let $\gamma = \infty$. Note that we do allow solutions where $\gamma > 1$. This allows the result to be reused to speed up computations, as described below.

C. Computational Complexity

In worst case, finding γ^* in each iteration of Algorithm 2 requires checking M obstacles per link and N^2 links. If there is more than one iteration, $\gamma(e, w_k^n, x, y)$ is never recomputed for a link. Instead, one only needs to subtract γ^* to get an updated value. Evaluating the iteration condition on connectivity requires testing at most N^2 links against M obstacles in each iteration. In worst case, there could be $O(MN^2)$ iterations. Since computing $\gamma(e, w_k^n, x, y)$ and testing a link against an obstacle are constant-time operations, the worst-case time complexity of each query to the outage detector is $O(M^2N^4)$.

As mentioned earlier, the RRT algorithm is only probabilistically complete, so its worst-case execution time is unbounded. As described by LaValle [17], the running time grows with the dimensionality N of $\mathcal{C}$, but also heavily depends on the presence of narrow bottlenecks in $\mathcal{C}_{\text{free}}$. In our problem, this corresponds to dense obstacles, where only a small set of robot configurations allows the group to pass without losing connectivity. In the following section, we will illustrate how the obstacle density affects the solution times.

V. Simulations

In this section, we show some simulations to illustrate the solution method. We also demonstrate how the solution times increase when the obstacles are denser, creating bottlenecks in $\mathcal{C}$. All simulations were made in Matlab, and we have not optimized the implementation for speed. It mainly serves as a proof of concept and to elucidate the relative differences in typical solution times.

As a small illustrative example, Fig. 4 shows a scenario with two obstacles and three paths. The robots are depicted in the start configuration. The corresponding configuration space is shown in Fig. 5. It shows the $\mathcal{C}$-obstacles, drawn as point clouds. When the system is at a configuration x inside a $\mathcal{C}$-obstacle, it means that the group is in outage, *i.e.*, that not enough robots have a clear line of sight for the communication graph $G_c(x)$ to be connected. The RRT G_R is grown between the obstacles, starting at $(0,0,0)$ and expanding through free space towards the goal $(1,1,1)$. Its vertices are drawn as circles and the edges are drawn

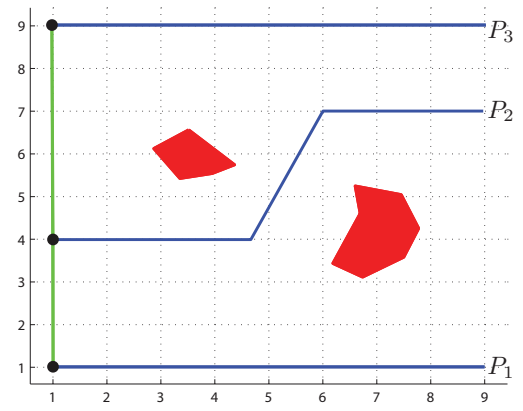

Fig. 4. A simple example scenario with three robots and two obstacles. The robots are depicted in the initial configuration.

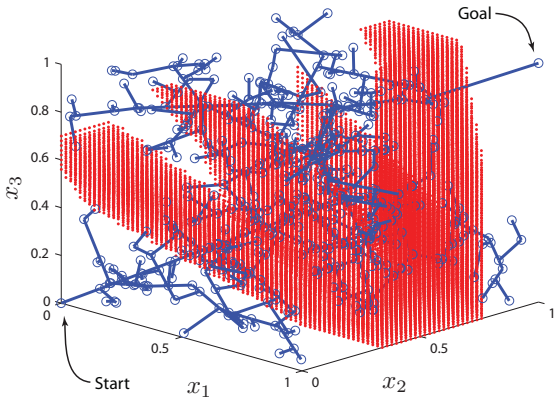

Fig. 5. Configuration space corresponding to the example in Fig. 4. The RRT vertices are shown as blue circles, joined by edges in the form of blue lines. The $\mathcal{C}$-obstacles are shown as red point clouds and the start and goal configurations are indicated.

as straight lines connecting the circles. Fig. 6 shows eight snapshots of the resulting solution, with dashed red lines depicting lines of sight that are blocked. A movie of the resulting robot motion can be found at `http://www.ee.kth.se/~lindhe/RRT_MOV`. At the same location, there is also a movie of a solution to the example scenario in Fig. 1. That solution took 144 s to compute on a laptop with an Intel Core 2 Duo processor at 2.2 GHz and 2 GB of RAM.

Finally, we demonstrate how the solution times depend on the obstacle density. We define a scenario with five paths and two triangular obstacles between each path. The triangles were vertically centered between the paths, but the horizontal position and the orientation were randomized for each trial. We first made 100 trials with small obstacles, where the base of the triangles was 20% of the distance between paths. Fig. 7 shows a histogram of the solution times, along with one realization of a scenario. All scenarios were solved and

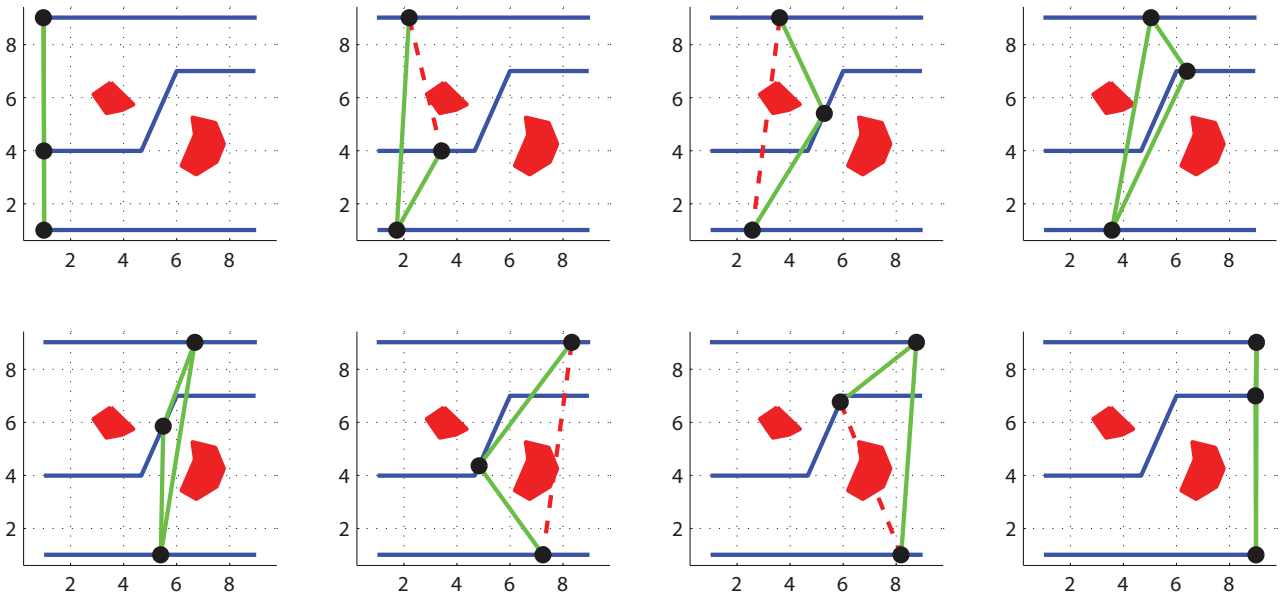

Fig. 6. Snapshots of a solution trajectory for the example in Fig. 4. The robots stay connected during the whole trajectory, even though some lines of sight (red dashed lines) are obstructed by obstacles.

the mean solution time was 6.3 s.

As a comparison, we made 100 similar trials with larger obstacles, where the base of each triangle was 50% of the distance between paths. Fig. 8 shows the resulting solution time histogram and an example realization. (Please note the different time scale from Fig. 7.) The maximum size of the RRT was bounded to $\Gamma = 50\ 000$ vertices, and with this termination rule, 89% of the scenarios were solved. The solution time was roughly proportional to Γ, and the search was terminated after about 700 s. The mean solution time for the solved scenarios was 78 s. We finally note that in similar experiments, with all 100 trials using the same scenario, the randomness of the RRT caused a similar spread in solution times, both for scenarios with small and large obstacles.

These tests illustrate two things: First, as expected, the solution times increase when the obstacles become denser, since there are narrower bottlenecks that the RRT needs to pass through. Second, the solution time histogram has a long tail of solvable scenarios where the solution takes very long to find, making the choice of termination time difficult. As mentioned in Section III, this is a known problem with this class of sampling-based solvers [19].

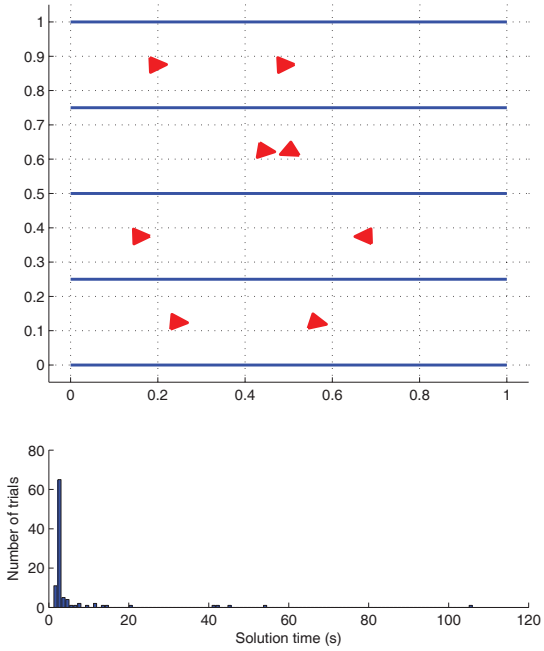

Fig. 7. Solution times for 100 randomized scenarios with small obstacles. One example realization is shown above.

VI. Conclusions

We have shown how the problem of path following under a visibility connectivity constraint can be represented as a classic motion planning problem: With N robots moving along pre-defined paths, we get an N-dimensional configuration space. All configurations that cause the group to be in outage are considered as $\mathcal{C}$-obstacles. The problem can then be stated as finding a continuous path in $\mathcal{C}_{\text{free}}$, from the initial to the final configuration.

Using the configuration space formulation, we have a number of candidate methods from the motion planning literature. Choosing the sampling-based RRT method, we get the advantage that the computation scales favorably with the number of robots. The main drawback is that the method is only probabilistically complete, so, as we have illustrated in simulations, the choice of termination criterion for the search

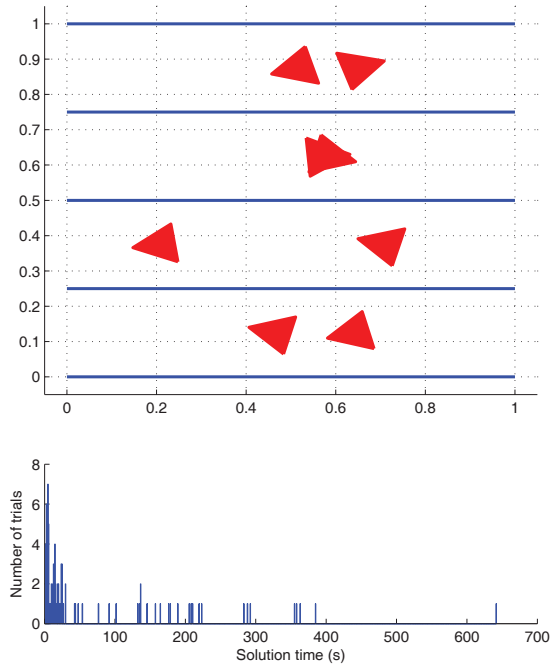

Fig. 8. Solution times for 100 randomized scenarios with larger obstacles. One example realization is shown above. Larger obstacles cause narrow bottlenecks in the configuration space, slowing down the RRT solver.

will be heuristic. This method cannot in finite time conclude that a problem is unsolvable, which may well be the case. For this reason, an interesting direction of future research would be to find an exact solution method and compare how its computational complexity scales with the number of robots.

Simulations illustrate the expected result that it takes longer to solve problems with dense obstacles, since the RRT needs to pass through bottlenecks in $\mathcal{C}$. Replacing the local solver FIRST_OUTAGE, which searches in a straight line, with a potential-based solver that can reach further, could mitigate this effect. This is a tradeoff between complexity of the local solver and its ability to speed up the passage of bottlenecks [20, Sec. 3.6].

We end by noting that it would be straightforward to modify the method to only accept certain restricted types of connectivity, such as topologies with a specified degree of redundancy or a maximum number of hops between any two robots. The former could be useful to make the connectivity more robust and the latter corresponds to improving the speed of information sharing within the group. Similarly, the case of intersecting paths could be solved by extending the outage detector to also check for inter-robot collisions. When searching for a path between two points, it would stop just before the system either goes into outage or there is an inter-robot collision.

REFERENCES

[1] O. Tekdas, W. Yang, and V. Isler, "Robotic routers: Algorithms and implementation," *International Journal of Robotics Research*, vol. 29, no. 1, pp. 110–126, January 2010.

[2] J. M. Esposito and T. W. Dunbar, "Maintaining wireless connectivity constraints for swarms in the presence of obstacles," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, Orlando, USA, May 2006.

[3] A. Ganguli, J. Cortés, and F. Bullo, "Multirobot rendezvous with visibility sensors in nonconvex environments," *IEEE Transactions on Robotics*, vol. 25, no. 2, pp. 340–352, April 2009.

[4] A. Ganguli, J. Cortes, and F. Bullo, "Visibility-based multi-agent deployment in orthogonal environments," in *Proceedings of the American Control Conference*, New York City, USA, July 2007, pp. 3426–3431.

[5] D. Anisi, P. Ögren, and X. Hu, "Cooperative minimum time surveillance with multiple ground vehicles," *IEEE Transactions on Automatic Control*, vol. 55, no. 12, pp. 2679–2691, December 2010.

[6] G. Hollinger and S. Singh, "Multi-robot coordination with periodic connectivity," in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, May 2010.

[7] B. Tovar and S. M. LaValle, "Visibility-based pursuit-evasion with bounded speed," in *Proceedings of the Workshop on Algorithmic Foundations of Robotics*, New York City, USA, July 2006.

[8] S.-M. Park, J.-H. Lee, and K.-Y. Chwa, "Visibility-based pursuit-evasion in a polygonal region by a searcher," in *Lecture Notes in Computer Science*, ser. 456–468, vol. 2076, 2001.

[9] B. Tovar, L. Guilamo, and S. M. LaValle, "Gap navigation trees: Minimal representation for visibility-based tasks," in *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, vol. 17, 2004.

[10] C. T. Shermer, "Recent results in art galleries," in *Proceedings of the IEEE*, vol. 80, no. 9, September 1992.

[11] U. Nilsson, P. Ögren, and J. Thunberg, "Optimal positioning of surveillance UGVs," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, September 2008, pp. 2539–2544.

[12] R. Murrieta-Cid, T. Muppirala, A. Sarmiento, S. Bhattacharya, and S. Hutchinson, "Surveillance strategies for a pursuer with finite sensor range," *Internationall Journal of Robotics Research*, vol. 26, no. 3, pp. 233–253, 2007.

[13] P. Fabiani, P. Gonzalez-Banos, J. Latombe, and D. Lin, "Tracking a partially predictable target with uncertainties and visibility constraints. p." *Journal of Robotics and Autonomous Systems*, vol. 38, no. 1, pp. 31–48, 2002.

[14] F. Chaumette and S. Hutchinson, "Visual servo control, part I: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, December 2006.

[15] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *International Journal of Robotics Research*, vol. 5, no. 3, pp. 72–89, 1986.

[16] E. P. C. Jones, L. Li, J. Schmidtke, and P. A. S. Ward, "Practical routing in delay-tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, pp. 943–959, August 2007.

[17] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. [Online]. Available: http://planning.cs.uiuc.edu/

[18] ——, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, Tech. Rep. 98-11, October 1998.

[19] S. Carpin, "Randomized motion planning – a tutorial," *International Journal of Robotics and Automation*, vol. 21, no. 3, pp. 184–196, 2006.

[20] R. Geraerts, "Sampling-based motion planning: Analysis and path quality," Ph.D. dissertation, Utrecht University, 2006.