Path Planning for Swarms in Dynamic Environments by Combining Probabilistic Roadmaps and Potential Fields

Alex Wallar Erion Plaku

Abstract— **This paper presents a path-planning approach to enable a swarm of robots move to a goal region while avoiding collisions with static and dynamic obstacles. To provide scalability and account for the complexity of the interactions in the swarm, the proposed approach combines probabilistic roadmaps with potential fields. The underlying idea is to provide the swarm with a series of intermediate goals which are obtained by constructing and searching a roadmap of likely collision-free guides. As the swarm moves from one intermediate goal to the next, it relies on potential fields to quickly react and avoid collisions with static and dynamic obstacles. Potential fields are also used to ensure that the swarm moves in cohesion. When the swarm deviates or is unable to reach the planned intermediate goals due to interferences from the dynamic obstacles, the roadmap is searched again to provide alternative guides. Experiments conducted in simulation demonstrate the efficiency and scalability of the approach.**

I. Introduction

Emerging applications of swarm robotics in exploration, monitoring, inspection, and search-and-rescue missions require the swarm to be able to move to a goal destination while avoiding collisions with obstacles [1], [2]. As the swarm is often comprised of a large number of robots, path planning becomes challenging due to the high-dimensionality of the underlying configuration space.

As a result, reactive approaches are often employed that avoid planning directly in the high-dimensional configuration space but instead regulate the swarm behavior through common rules of interactions. Stigmergic approaches, inspired by how ants move back-and-forth from the nest to a food source, utilize synthetic pheromone traces left in the environment by forager robots to guide the swarm to the target [3]–[5]. While pheromone-based navigation minimizes communication, it lacks the flexibility to quickly adapt to dynamic changes in the environment that could block or disrupt existing pheromone traces. Other approaches rely on wireless network nodes deployed at various locations in the environment to provide global coverage and guide the swarm to the target [6], [7]. Genetic algorithms have also been used to facilitate navigation by evolving collective behaviors for the swarm, such as chain formation or obstacle avoidance [8], [9]. In leader-follower approaches, one or few robots move along precomputed paths to the target location while the rest of the swarm seeks to follow the leaders, often maintaining a desired formation [10]–[12]. The work in [13] proposes a fully-decentralized algorithm based on the concept of reciprocal velocity obstacles to enable collision-free motions among a group of robots. Other approaches based on virtual fixtures treat the swarm as a rigid body, seeking to maintain a fixed relation among swarm members [14]–[16]. While such approaches reduce swarm path planning to single-robot planning, the imposed structure rigidity and the increased dimensionality make it difficult to efficiently plan paths especially when considering a large number of robots.

Artificial potential functions (APFs) provide a common approach for swarm path planning by imposing virtual repulsive forces from obstacles and attractive forces to the goal [17]–[22]. APFs are fast to compute as the resulting forces for each robot in the swarm depend only on the nearby obstacles, the goal region, and limited interactions with neighboring robots in the swarm. While APFs provide local, reactive, behaviors that enable the swarm to avoid obstacles, the global path guidance toward the goal suffers from local minima. This limitation of APFs becomes more prevalent when considering path planning for large swarms moving in cluttered environments. Designing APFs that avoid or minimize the likelihood of the robots becoming stuck in local minima remains a challenging problem [23].

Alternative approaches seek to avoid local minima by relying on global path planning. Probabilistic roadmaps (PRM) [24] and other sampling-based path planners [25]–[29] provide global path planning by using sampling to capture the connectivity of the free configuration space. In particular, PRM approaches construct roadmaps resembling a network of roads obtained by sampling a large number of configurations and connecting neighboring configurations via collision-free paths. The roadmap is then used to perform different tasks, such as homing, goal searching, or shepherding [30], [31]. PRMs have also been combined with Bezier curves to guide a number of nonholonomic robots to the goal while maintaining formation [32].

While PRMs avoids the issues associated with local minima, scalability starts becoming problematic when dealing with a large number of robots. As the configuration space of a swarm consist of the Cartesian product of the individual configuration spaces of each robot, it becomes challenging to generate collision-free configurations and connect neighboring configurations via collision-free paths.

To provide scalability and account for the complexity of the interactions in the swarm, this paper builds upon our prior work [33] which combined PRMs with APFs. While prior work was limited to static obstacles, the proposed approach, termed `dCRoPS` (`C`ombined `Ro`admaps and `P`otentials for `S`warms for `D`ynamic Obstacles), can efficiently guide the

A. Wallar is with the School of Computer Science, University of St Andrews, Fife KY16 9AJ, Scotland, UK. E. Plaku is with the Dept. of Electrical Engineering and Computer Science, Catholic University of America, Washington DC 20064 USA.

swarm to the goal even in the presence of dynamic obstacles. The underlying idea is to provide the swarm with a series of intermediate goals which are obtained by constructing and searching a roadmap of likely collision-free guides. To maintain scalability, the roadmap is constructed over the two-dimensional workspace in which the swarm moves rather than the high-dimensional configuration space associated with the swarm. As the swarm moves from one intermediate goal to the next, it relies on potential fields to quickly react and avoid collisions with static and dynamic obstacles.

In contrast to prior work [33], dCRoPS provides robots in the swarm with alternative guides to the goal when a robot is unable to reach the planned intermediate targets due to interferences from the dynamic obstacles. Weights of roadmap edges in or near the obstructed area are dynamically adjusted to account for lack of progress and to guide the swarm away from such areas. While it may also be possible to use grids or Voronoi diagrams to guide the swarm, roadmaps have been shown to better capture the connectivity of the free space [34][chap. 7].

dCRoPS also improves the APFs over our prior work [33] in order to enable the swarm to react rapidly to the dynamic obstacles moving towards it. In particular, dCRoPS introduces a potential field which influences the heading of a robot b by leveraging the headings of other robots that have been in the past close to b's current position and have managed to make further progress toward the goal.

Experiments are conducted in simulation where large swarms have to move in complex environments, often through narrow passages, while avoiding collisions with static obstacles and numerous dynamic obstacles. Results demonstrate the efficiency and scalability of dCRoPS.

II. Problem Formulation

The swarm consists of a number of mobile circular robots $b_1, \ldots, b_m$. The swarm operates inside a two-dimensional environment W populated by polygonal static and dynamic obstacles, denoted as *StaticObstacles* and *DynamicObstacles*, respectively. While the static obstacles remain fixed, the dynamic obstacles move along different directions. Starting at an initial placement, the objective is for each robot to reach a goal region $G \subset W$ while avoiding collisions with the static and dynamic obstacles in W.

Each robot is capable of detecting nearby obstacles within Δ_{obst} radius. dCRoPS seeks to keep the robots moving as one group as much as possible, separating into two or more groups only when necessary to avoid the dynamic obstacles. The geometry and placement of each static obstacle is assumed to be available beforehand. No information is given to dCRoPS beforehand in regards to the dynamic obstacles. In the experiments, the dynamic obstacles move along random directions, which are not known to dCRoPS.

III. Method

Pseudocode for dCRoPS is provided in Alg. 1. The main steps are described below.

A. Global Path Planning

dCRoPS uses a roadmap to provide each robot in the swarm with global path planning and find alternative guides to the goal region when dynamic obstacles prevent a robot from moving along its current guide.

1) Roadmap Construction: Pseudocode for the roadmap construction is provided in Alg. 2. The roadmap is represented as a graph $RM = (V, E)$. A vertex $q \in V$ corresponds to a collision-free point in W. An edge $(q_i, q_j) \in E$ corresponds to a collision-free straight-line segment from q_i to q_j. Collision checking during roadmap construction is performed with respect to the static obstacles, which are assumed to be known.

To generate a roadmap vertex, a point is sampled uniformly at random inside W. The point is checked whether it is at least a certain distance, d_{clear}, away from the static obstacles. If so, the sampled point is added as a roadmap vertex. Otherwise, this sampling process is repeated until the sampled point has a clearance of d_{clear} from the static obstacles (Alg. 2:1–4). In this way, dCRoPS seeks to avoid areas close to the obstacles since such areas are more likely to lead to collisions with the dynamic obstacles, other robots, and static obstacles. The total number of roadmap vertices, n, is provided by the user. This is a common strategy in PRM approaches [24], [34], as it is difficult to determine beforehand an optimal number of roadmap vertices [35]. Experiments are presented in Section IV that show the impact of the roadmap size on the overall performance of dCRoPS.

Roadmap edges are generated by connecting each $q \in V$ with straight-line segments to its k-nearest roadmap vertices according to the Euclidean distance (Alg. 2:6–7). Collision checking is performed with respect to the static obstacles, discarding any edge that is found in collision (Alg. 2:8). A weight $w(q_i, q_j)$ is associated with each edge $(q_i, q_j) \in E$ as an estimate on the feasibility of having the swarm move from q_i to q_j (Alg. 2:9). More specifically, $w(q_i, q_j)$ is defined as

$$w(q_i, q_j) = \left(\min_{o \in \textit{StaticObstacles}} \textit{dist}((q_i, q_j), o) \right)^{-3},$$

where $\textit{dist}((q_i, q_j), o)$ denotes the Euclidean distance between the static obstacle o and the straight-line segment from q_i to q_j. As such, small values of $w(q_i, q_j)$ indicate that the straight-line segment from q_i to q_j is away from the static obstacles. As explained later, dCRoPS guides the swarm along low-weight roadmap edges in order to make it easier to avoid collisions with the static and dynamic obstacles, and to minimize congestion within the swarm. Note that while we have chosen through experimentation a particular definition of $w(q_i, q_j)$ for this paper, alternative weight functions can also be used.

2) Guiding the Swarm: In order to move the swarm toward the goal region G, each robot $b \in \textit{Robots}$ is provided with a global guide, denoted by $b.\textit{guide}$, which represents a path along low-weight roadmap edges from the current position of the robot, denoted by $b.\textit{pos}$, to the goal region G. More precisely, each robot b is initially assigned a random

Algorithm 1 Pseudocode for `dCRoPS`

```
1: $\langle RM = (V, E), w\rangle \leftarrow$ ROADMAP$(W, n, k, d_{\mathit{clear}})$
2: for $b \in$ Robots do
3:   $b.\mathit{guide} \leftarrow \emptyset$; $b.\mathit{finalGoal} \leftarrow$ RandomPoint$(G)$
4: while $\exists b \in$ Robots with $b.\mathit{pos} \notin G$ do
5:   for $o \in$ DynamicObstacles do
6:     $o.\mathit{pos} \leftarrow$ MoveDynamicObstacle$(o)$
7:   for $b \in$ Robots with $b.\mathit{pos} \notin G$ do
8:     if $b.\mathit{guide} = \emptyset$ then
9:       $b.\mathit{guide} \leftarrow$ GUIDE$(RM, w, b.\mathit{pos}, b.\mathit{finalGoal})$
10:      $b.\mathit{next} \leftarrow 1$
11:    while $||b.\mathit{pos}, b.\mathit{guide}(b.\mathit{next}||_2 \leq d_{\mathit{reach}}$
       and $b.\mathit{next} < |b.\mathit{guide}|$ do $b.\mathit{next} \leftarrow b.\mathit{next} + 1$
12:    $b.\mathit{heading} \leftarrow$ SUPERIMPOSE$(PF_{\mathit{obst}}(b), PF_{\mathit{sep}}(b),$
                                    $PF_{\mathit{next}}(b), PF_{\mathit{hist}}(b))$
13:    $b.\mathit{pos} \leftarrow b.\mathit{pos} + \mathit{step} \cdot \frac{b.\mathit{heading}}{||b.\mathit{heading}||_2}$
14:    if COLLISION$(b)$ return failure
                         ◇ compute alternative guide if necessary
15:    if STUCK$(b)$ = true then
16:      for $\ell = b.\mathit{next} \ldots \min(b.\mathit{next} + e, |b.\mathit{guide}| - 1)$ do
17:        $q_i \leftarrow b.\mathit{guide}(\ell)$; $q_j \leftarrow b.\mathit{guide}(\ell + 1)$
18:        $w(q_i, q_j) \leftarrow w(q_i, q_j) \cdot \mathit{penalty}$
19:      $b.\mathit{finalGoal} \leftarrow$ RandomPoint$(G)$
20:      $b.\mathit{guide} \leftarrow$ GUIDE$(RM, w, b.\mathit{pos}, b.\mathit{finalGoal})$
21: return success
```

Algorithm 2 ROADMAP$(W, n, k, d_{\mathit{clear}})$

```
1: for $i = 1 \ldots n$ do
2:   repeat $q \leftarrow$ RandomPoint$(W)$
3:   until $(\min_{o \in \mathit{StaticObstacles}} \mathit{dist}(q, o)) > d_{\mathit{clear}}$
4:   $V \leftarrow V \cup \{q\}$
5: for $q_i \in V$ do
6:   for $q_j \in$ NEIGHS$(V, q_i, k)$ do
7:     if COLLISION$(q_i, q_j, \mathit{StaticObstacles})$ = false then
8:       $E \leftarrow E \cup \{(q_i, q_j)\}$
9:       $w(q_i, q_j) \leftarrow (\min_{o \in \mathit{StaticObstacles}} \mathit{dist}((q_i, q_j), o))^{-3}$
10: return $(RM = (V, E), w)$
```

point inside G to act as its final goal, denoted by $b.\mathit{finalGoal}$ (Alg. 1:3). Then, $b.\mathit{guide}$ is computed as the shortest path in the roadmap $RM = (V, E)$ from q' to q'', where q' is the closest roadmap vertex to $b.\mathit{pos}$ and q'' is the closest roadmap vertex to $b.\mathit{finalGoal}$ (Alg. 1:9). A* search is used to make the computation of the guide efficient.

Each robot b then seeks to reach G by following its guide, i.e., going to $b.\mathit{guide}(1), \ldots, b.\mathit{guide}(|b.\mathit{guide}|)$ in succession. More precisely, $b.\mathit{next}$ keeps track of the index in $b.\mathit{guide}$ which serves as the next target for b. Initially, $b.\mathit{next}$ is set to 1 since b seeks to move to $b.\mathit{guide}(1)$ (Alg. 1:10). The target is considered to be reached when $b.\mathit{pos}$ is within a certain radius, d_{reach}, from $b.\mathit{guide}(b.\mathit{next})$. When b reaches $b.\mathit{guide}(1)$, then $b.\mathit{next}$ is set to 2, and so on (Alg. 1:11).

As discussed later in Section III-B, potential fields are used to move b from its current position to the next target $b.\mathit{guide}(b.\mathit{next})$. The combination of these potential fields determines the vector $b.\mathit{heading}$ along which the robot b should move in order to make progress toward the next target, $b.\mathit{guide}(b.\mathit{next})$, while avoiding collisions with the dynamic obstacles, static obstacles, and other robots. The robot b then takes a step along this direction (Alg. 1:12–13).

param	description	section
n	number of roadmap vertices	§III-A.1
k	number of nearest neighbors in roadmap	§III-A.1
d_{clear}	clearance from static obstacles	§III-A.1
d_{reach}	next target radius	§III-A.2
step	robot step size	§III-A.2
e	number of current guide edges to penalize	§III-A.3
$\mathit{penalty}$	scaling factor for current guide edges	§III-A.3
δ_{next}	scaling constant for PF_{next}	§III-B.1
δ_{obst}	scaling constant for PF_{obst}	§III-B.2
Δ_{obst}	radius of influence for PF_{obst}	§III-B.2
δ_{sep}	scaling constant for PF_{sep}	§III-B.3
Δ_{sep}	radius of influence for PF_{sep}	§III-B.3
δ_{hist}	scaling constant for PF_{hist}	§III-B.4

Fig. 1. Parameters used by `dCRoPS`. More details can be found in the corresponding sections.

3) Global Replanning via Alternative Guides: If a robot b gets stuck in a local minima or a dynamic obstacle prevents it from reaching its next target $b.\mathit{guide}(b.\mathit{next})$ along its current guide, `dCRoPS` will search the roadmap again to find an alternative guide for b (Alg. 1:15–19). This is akin to the replanning stage used in sampling-based motion planning [36]–[38]. Before the roadmap search, `dCRoPS` adjusts the weight of the next few edges on the current guide in order to guide b away from this area. More precisely, let $q(i) = b.\mathit{guide}(b.\mathit{next}{+}i)$. Then, `dCRoPS` scales the weight of edges $(q(0), q(1)), \ldots, (q(e-1), q(e))$ by a factor $\mathit{penalize} > 1$, where e is the number of edges to penalize. As a result of these weight increases, the shortest-path search will produce an alternative guide.

This adjustment of edge weights is used to relieve swarm congestion when it is confronted by an obstructing obstacle. The creation of a new guide provides the robot with an alternative pathway, which could help it reduce the average wait time, the average time stuck, and it can guide the swarm around obstructing dynamic obstacles that are blocking critical passageways.

B. Local Path Planning

`dCRoPS` uses APFs to make each robot in the swarm follow its guide and quickly react to obstacles it may encounter along the way. Several criteria are taken into consideration when designing the APFs for a robot b:

1) b is attracted to its next target $b.\mathit{guide}(b.\mathit{next})$;
2) b is repulsed away from obstacles;
3) b moves in cohesion with the other robots while maintaining some separation from neighboring robots in order to minimize the risk of robot-robot collisions;
4) b leverages the headings of other robots that have been in the past close to b's current position and have managed to make further progress.

1) Attraction to the Next Target along the Current Guide: To pull the robot to the next target, `dCRoPS` defines an attractive potential as follows:

$$PF_{\mathit{next}}(b) = \delta_{\mathit{next}} \cdot (b.\mathit{guide}(b.\mathit{next}) - b.\mathit{pos}) \cdot ||b.\mathit{guide}(b.\mathit{next}) - b.\mathit{pos}||_2,$$

where δ_{next} is a scaling constant. Note that when the robot is away from its next target, the potential exerts a larger attractive force to bring the robot closer.

2) Repulsion from Obstacles: To avoid collisions, dCRoPS relies on a repulsive potential to push each robot away from the obstacles. Since dCRoPS is designed for dynamic obstacles, the responsiveness in the obstacle potential needs to also increase in order for the robots to react quickly to moving obstacles. More specifically, the repulsive potential between a robot b and an obstacle o is defined as

$$PF_{obst}(b, o) = \delta_{obst} \frac{b.pos - ClosestPoint(o, b.pos)}{(dist(b.pos, o))^2},$$

where δ_{obst} is a scaling constant and $dist(b.pos, o)$ is the distance from the point $b.pos$ to the obstacle o. As it is common in APFs, obstacles that are far away do not exert any repulsive forces. As such, only obstacles that are within a certain distance Δ_{obst} from pos_b are used for the computation of the overall repulsive potential, i.e.,

$$PF_{obst}(b) = \sum_{\substack{o \in StaticObstacles \cup DynamicObstacles \\ dist(b.pos, o) \leq \Delta_{obst}}} PF_{obst}(b, o).$$

3) Repulsion from Neighboring Robots: In order to ensure that robots do not get too close to one another, which could lead to collisions and swarm congestion, dCRoPS imposes a repulsive potential. A weak potential is used instead of a strong one as the objective is to push b at a safe distance from its neighbors but not to separate it from the swarm. For this reason, the repulsive potential between two robots b_i and b_j is defined as

$$PF_{sep}(b_i, b_j) = \delta_{sep} \frac{b_i.pos - b_j.pos}{||b_i.pos, b_j.pos_{b_j}||_2},$$

where δ_{sep} is a scaling constant. Similar to the case of the repulsive potential for obstacles, only robots that are within a certain distance Δ_{sep} have an influence on b, i.e.,

$$PF_{sep}(b) = \sum_{\substack{b_i \in Robots - \{b\} \\ dist(b, b_i) \leq \Delta_{sep}}} PF_{sep}(b, b_i).$$

4) Leveraging History: The heading of the robot b is also influenced by the headings of other robots that have been in the past close to b's current position and have managed to make further progress. More specifically, a grid is implicitly imposed on W. Each grid cell c keeps track of the heading of the robots that have entered c in the past and have then left it – the list of such robots is denoted by $c.Robots$. Then,

$$PF_{hist}(b) = \frac{\delta_{hist}}{|cell(b).Robots|} \sum_{b' \in cell(b).Robots} b'.lastHeading,$$

where δ_{hist} is a scaling constant, $cell(b)$ denotes the grid cell that contains $b.pos$, and $b'.lastHeading$ denotes the direction of b' when it was last in $cell(b)$. In this way, b can use the heading of other robots that have left $cell(b)$ so that it too can leave this cell and make progress toward the next target. By following headings of other robots, this potential field also promotes cohesion among robots in the swarm.

5) Combining the Potential Fields: The overall effect on b is obtained by combining the four different potential fields:

$$PF(b) = \frac{\sum_{\phi \in fields} (||PF_\phi(b)||_2 PF_\phi(b))}{\sum_{\phi \in fields} ||PF_\phi(b)||_2},$$

where $fields = \{next, obst, sep, hist\}$. This overall force is then used to update the heading and position of b (Alg. 1:12–13). In this way, b is attracted to the next target, strongly repulsed from the obstacles, seeks to maintain a separation distance from its neighbors, and leverages headings of other robots to move to the next target. When a robot becomes stuck, the weights associated with the next several roadmap edges of the guide are increased, and a new alternative guide is computed to help it escape. This combination of global path planning via roadmaps and local path planning via potential fields enables the approach to effectively guide the swarm to the final goal while avoiding collisions with the static and dynamic obstacles.

IV. Experiments and Results

Experiments are conducted in simulation with an increasing number of robots moving in complex environments, seeking to avoid numerous static and dynamic obstacles. The scene models and parameter values used in the experiments are made publicly available [39].

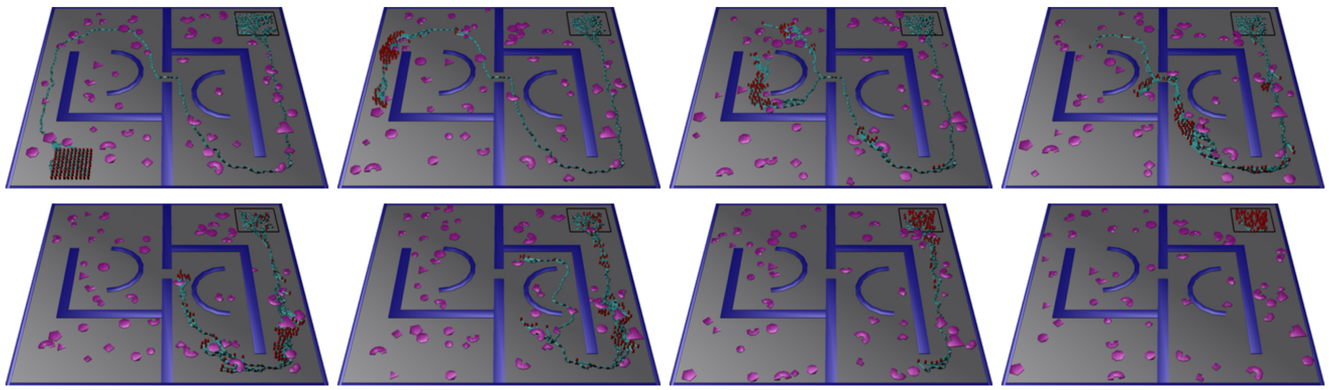
A. Experimental Setup

An illustration of the three different scenes used in the experiments is provided in Fig. 2. Each scene is comprised of static and dynamic obstacles. These scenes provide challenging test cases as the swarm has to go through numerous narrow passages that are frequently blocked by the random motions of the dynamic obstacles.

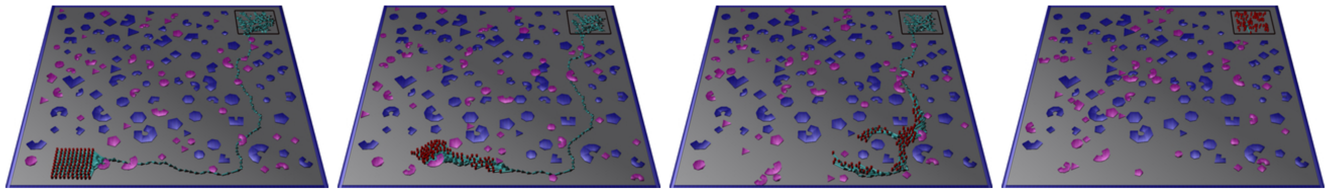
1) Random Motions for the Dynamic Obstacles: Recall that dCRoPS has no *a priori* information on the behavior of the dynamic obstacles. In the experiments, the dynamic obstacles are made to move at random. More specifically, each dynamic obstacle moves in small steps toward a random point. When it reaches the random point or encounters a static or a dynamic obstacle, it starts moving toward another random point. The dynamic obstacles move about 2–4 times slower than the robots.

The initial placement of the dynamic obstacles is also done at random. More specifically, the i-th dynamic obstacle o_i is placed by generating a random position inside W repeatedly until the placement of o_i is not in collision with the static obstacles, the previously-placed dynamic obstacles $o_1, \ldots, o_{i-1}$, and the robots in their initial placement.

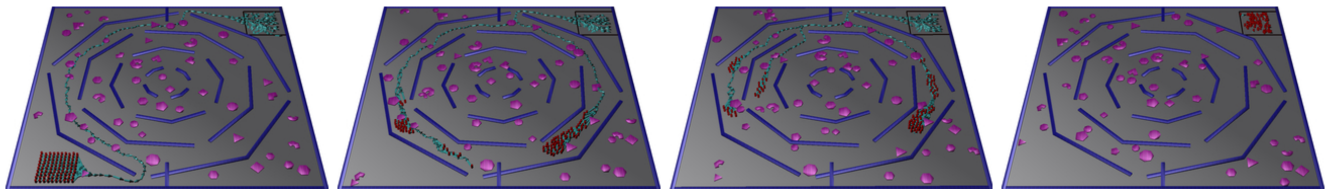
2) Measuring Performance: A problem instance is defined by a scene, the number of robots, and the number of dynamic obstacles. In the experiments, the number of robots is varied from 10 to 100 in increments of 10. The number of dynamic obstacles is varied from 0 to 50 in increments of 10. Results for each problem instance report mean time and standard deviation based on twenty different runs. Runtime includes the time to construct the roadmap, which was negligble (around 0.2s). Experiments are conducted on an



(a) scene 1



(b) scene 2



(c) scene 3

Fig. 3. Illustration of dCRoPS on three different scenes. For each scene, several frames of a run of dCRoPS are shown. The first frame shows the swarm in the initial placement. The last frame shows the swarm when each robot has reached the goal region. The other frames show how the swarm progresses towards the goal region. Guides, which are computed by searching the roadmap, are shown in green arrows. Note that dCRoPS uses alternative guides when necessary to avoid collision with dynamic obstacles.

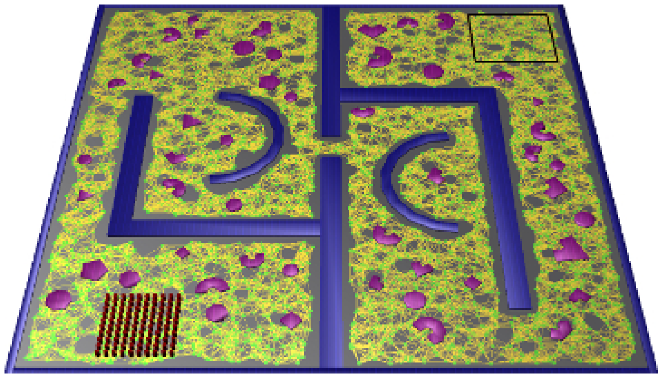
Intel Core i7 machine (CPU: 2.40GHz, RAM: 8GB) using Ubuntu 14.04. Code is written in C++ and compiled with g++4.8.2 using the optimization flag O2.

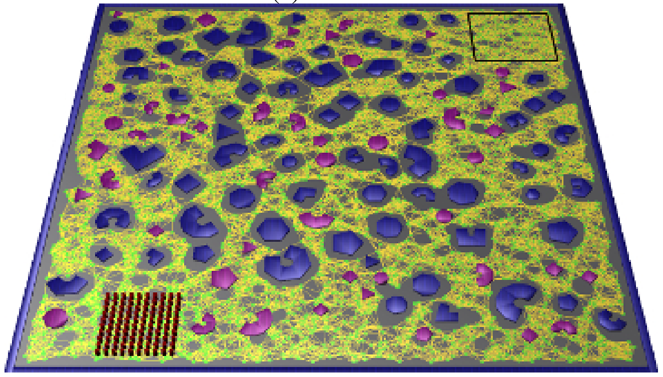
B. Results

Fig. 3 provides snapshots of dCRoPS when run on the three different scenes. These snapshots show that the swarm reaches the goal region, making use of alternative guides at times in order to avoid the obstacles. As shown next, dCRoPS is quite efficient and the running time scales linearly with the number of robots.

1) Scalability as a Function of the Number of Robots and Dynamic Obstacles: Fig. 4 provides a summary of the results when varying both the number of the robots and the number of the dynamic obstacles. These results show that dCRoPS is capable of efficiently enabling the swarm to reach the final goal while avoiding collisions with static and dynamic obstacles. The third scene presents the most challenging test case as the motions of the dynamic obstacles often block the small openings through which the swarm needs to pass in order to reach the goal region. As the results indicate, in all scenes, dCRoPS scales linearly with the number of robots, even as the number of the dynamic obstacles is increased. By combining roadmaps with potential fields, dCRoPS provides the swarm with global path planning to guide the robots toward the goal region while reacting to obstacles encountered along the way, making local adjustment and seeking alternative guides to avoid getting trapped.

2) Impact of Roadmap: Results in Fig. 5 and 6 show the impact of the roadmap on the overall performance of dCRoPS. Fig. 5 provides a summary of the results when varying the number of roadmap vertices (n in Alg. 2). In the case of scenes 1 and 2, even roadmaps with a small number of vertices are able to capture the connectivity of the free space and provide viable pathways to guide the swarm. In these cases, dCRoPS performs better with smaller roadmaps as it saves on the computational time required to query the roadmap to compute the robot guides without sacrificing the quality of the guides. In the case of scene 3, which provides a more challenging environment, the smaller roadmaps are not able to capture the connectivity of the free space. As a result, the quality of the guides deteriorates, which makes it more difficult for the robots to reach the goal region. Also, when the roadmap becomes


(a) scene 1


(b) scene 2

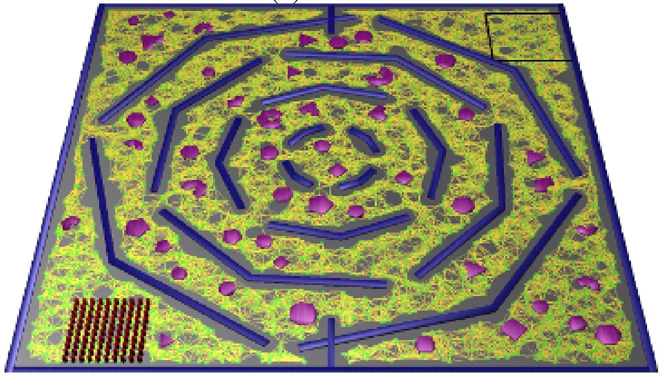
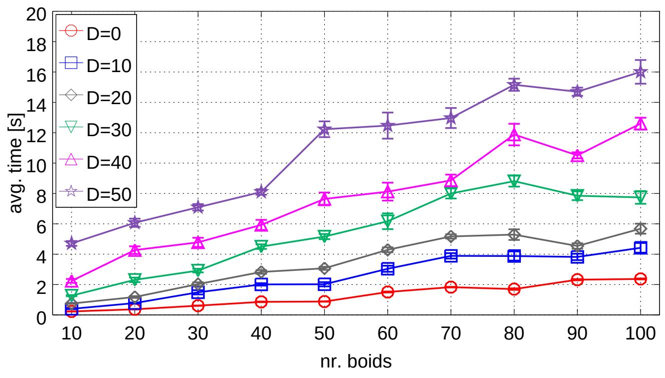
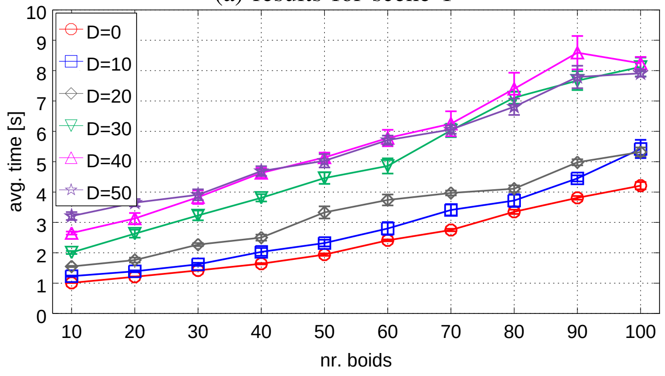

(c) scene 3

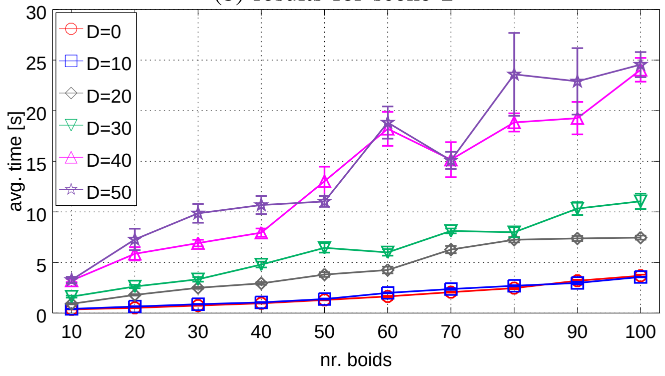
Fig. 2. Scenes used for the experiments. Each figure shows the static obstacles (blue), initial placement of the dynamic obstacles (magenta), initial placement of the robots (red), goal region (black box), and the roadmap (roadmap vertices are shown as green circle and roadmap edges are shown as yellow segments). Recall that collision checking in the roadmap is done only with respect to the static obstacles which are known beforehand.



(a) results for scene 1



(b) results for scene 2



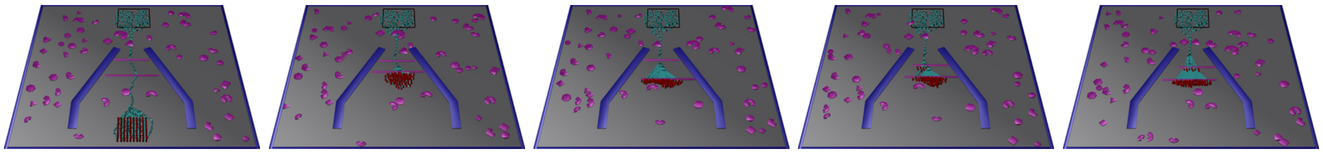
(c) results for scene 3

Fig. 4. Results show the mean runtime and standard deviation as a function of the number of robots and the number (D) of dynamic obstacles Runtime is measured as the time from the beginning till the last robot reaches the goal. It also includes the time to construct the roadmap (around 0.2s, roadmap used $n = 5000$ vertices and $k = 15$ neighbors). Mean is computed over twenty different runs for each problem instance.

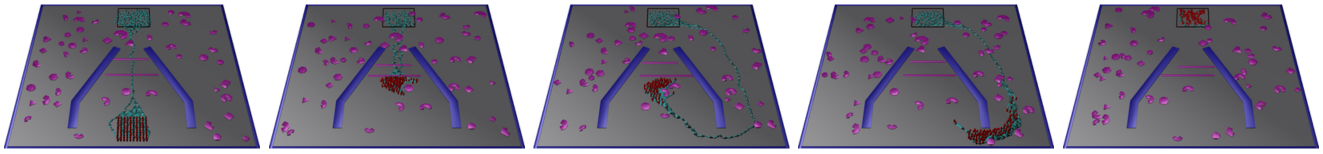
too large, the performance suffers from the computational cost associated with guide computations. Although finding an optimal number of roadmap vertices remains challenging [34], [35], [40], results show that dCRoPS works well for a wide range of values, e.g., $1000 - 5000$ for scenes 1 and 2, and $2500 - 7500$ for scene 3.

Fig. 6 provides a summary of the results when varying the number of nearest neighbors (k in Alg. 2) used for the roadmap construction. These results show that dCRoPS works well for a wide range of values. When k becomes too large, there is an increase in the running time of dCRoPS due to the larger computational cost to query the roadmap. Also note that in the case of scene 3, when k is small, it becomes difficult to capture the connectivity of the free space, which results in increased runtime for dCRoPS since the robot guides are not as effective.

3) Impact of Global Replanning: Fig. 7 provides a qualitative evaluation of the impact of global replanning via alternative guides on the overall performance of dCRoPS. As shown, without global replanning, it becomes difficult for the swarm to find a way around the dynamic obstacles which prevent the swarm from reaching the goal by following the initial guide. With global replanning, the swarm can make use of alternative pathways to reach the goal region.



(a) dCRoPS with global replanning disabled



(b) dCRoPS with global replanning enabled

Fig. 7. Illustration of the importance of global replanning via alternative guides in dCRoPS. The thin, rectangular, dynamic obstacles, which move left and right and a bit up and down, make it difficult for the swarm to reach the goal by following the initial guide. (a) When global replanning is disabled, the swarm gets stuck trying to pass these obstacles by following the initial guide. (b) When global replanning is enabled, once a robot become stuck, an alternative guide is computed, making it possible for the swarm to effectively reach the goal region.

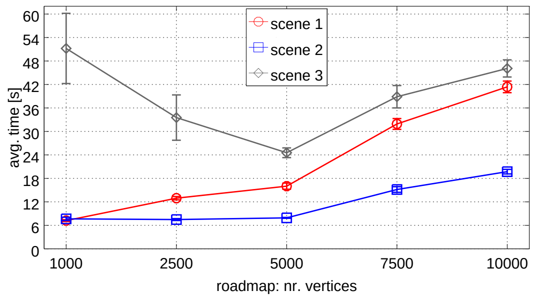


Fig. 5. Results show the mean runtime and standard deviation as a function of the number of roadmap vertices (n in Alg. 2). The number of nearest neighbors is kept at $k = 15$. Results are shown for the case of 100 robots and 50 dynamic obstacles. Runtime is measured as the time from the beginning till the last robot reaches the goal. Mean is computed over twenty different runs for each problem instance.

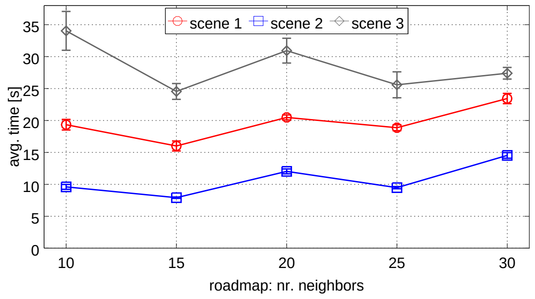


Fig. 6. Results show the mean runtime and standard deviation as a function of the number of nearest neighbors (k in Alg. 2) used for the roadmap construction. The number of roadmap vertices is kept at $n = 5000$. Results are shown for the case of 100 robots and 50 dynamic obstacles. Runtime is measured as the time from the beginning till the last robot reaches the goal. Mean is computed over twenty different runs for each problem instance.

	global replanning on		global replanning off	
nr. robots	mean	std	mean	std
20	4.14s	0.32	x	x
40	5.66s	0.32	x	x
60	8.00s	0.52	x	x
80	10.70s	0.82	x	x
100	11.77s	1.02	x	x

TABLE I

COMPARISON OF dCRoPS WHEN GLOBAL REPLANNING IS ENABLED VS DISABLED. RESULTS ARE FOR THE SCENE SHOWN IN FIG. 7. ENTRIES MARKED WITH X INDICATE THAT dCRoPS FAILED TO FIND A SOLUTION IN THE ALLOWED TIME OF 120S PER RUN.

Table I shows the running time of dCRoPS when global replanning is enabled vs disabled. As expected, dCRoPS fails to find a solution when global replanning is disabled since the dynamic obstacles prevent the swarm from reaching the goal. When global replanning is enabled, dCRoPS efficiently guides the swarm to the goal region by using alternative pathways. This is because of two things. First, global replanning relieves the congestion around narrow passageways and tries to split the swarm so it can "move in parallel." This means that instead of all of the robots going through one passageway sequentially, two groups of robots can go through different paths at the same time. The second reason global replanning produces better results is because the robots are not able to pre-generate a shortest path that will always keep the swarm out of unrecoverable configurations as they do not posses the ability to predict the position of dynamic obstacles. When a dynamic obstacle obstructs a narrow passageway and the shortest path goes through that passageway, as in Fig. 7, without global replanning the robots will not be able to reach the end until the obstacle moves out of the way. With global replanning enabled, the swarm is able to find a new way around the dynamic obstacle, therefore avoiding the obstructed passageway. This reduces the amount of time the swarm spends waiting and instead finds a new and more effective way to the goal region.

V. DISCUSSION

This paper proposed dCRoPS, a path-planning approach to enable a swarm of robots move to a goal region while avoiding collisions with static and dynamic obstacles. Scalability is obtained by an efficient combination of probabilistic roadmaps to provide global path planning with potential fields to provide local, reactive, behaviors necessary to avoid collisions with obstacles. dCRoPS leverages past history and global replanning via alternative guides to help stuck robots successfully find their way to the goal. Experimental results with an increasing number of robots and numerous dynamic obstacles show the efficiency and scalability of the approach. For future work, one direction is to predict the motions of the dynamic obstacles and incorporate those predictions to adjust the roadmap so that it can provide alternative guides that avoid predicted trajectories. Improving the roadmap quality could also improve the overall performance of dCRoPS. Another direction is to make use of high-level formation behaviors that can be leveraged to more easily move through certain narrow passages.

References

[1] E. Şahin, "Swarm robotics: from sources of inspiration to domains of application," in *International Conference on Swarm Robotics*, 2004, pp. 10–20.

[2] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, pp. 1–41, 2012.

[3] D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee, "Pheromone robotics," *Autonomous Robots*, vol. 11, pp. 319–324, 2001.

[4] S. Nouyan, A. Campo, and M. Dorigo, "Path formation in a robot swarm," *Swarm Intelligence*, vol. 2, no. 1, pp. 1–23, 2008.

[5] S. Nouyan, R. Groß, M. Bonani, F. Mondada, and M. Dorigo, "Teamwork in self-organized robot colonies," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 695–711, 2009.

[6] F. Ducatelle, G. A. Di Caro, C. Pinciroli, F. Mondada, and L. M. Gambardella, "Communication assisted navigation in robotic swarms: self-organization and cooperation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 4981–4988.

[7] K. J. OHara and T. R. Balch, "Pervasive sensor-less networks for cooperative multi-robot tasks," in *Distributed Autonomous Robotic Systems 6*, 2007, pp. 305–314.

[8] V. Sperati, V. Trianni, and S. Nolfi, "Evolving coordinated group behaviours through maximisation of mean mutual information," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 73–95, 2008.

[9] C.-C. Lin, K.-C. Chen, and W.-J. Chuang, "Motion planning using a memetic evolution algorithm for swarm robots," *International Journal of Advanced Robotic Systems*, vol. 9, pp. 1–9, 2012.

[10] S. Kloder and S. Hutchinson, "Path planning for permutation-invariant multirobot formations," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 650–665, 2006.

[11] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[12] G. L. Mariottini, F. Morbidi, D. Prattichizzo, G. J. Pappas, and K. Daniilidis, "Leader-follower formations: Uncalibrated vision-based localization and control," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 2403–2408.

[13] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[14] C. Belta and V. Kumar, "Optimal motion generation for groups of robots: a geometric approach," *Journal of Mechanical Design*, vol. 126, p. 63, 2004.

[15] T. Eren, P. N. Belhumeur, and A. S. Morse, "Closing ranks in vehicle formations based on rigidity," in *IEEE Conference on Decision and Control*, vol. 3, 2002, pp. 2959–2964.

[16] W. Ren and R. Beard, "Decentralized scheme for spacecraft formation flying via the virtual structure approach," *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 1, pp. 73–82, 2004.

[17] O. Khatib, "Real time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–99, 1986.

[18] J. H. Reif and H. Wang, "Social potential fields: A distributed behavioral control for autonomous robots," *Robotics and Autonomous Systems*, vol. 27, no. 3, pp. 171–194, 1999.

[19] W. M. Spears and D. F. Spears, *Physicomimetics: Physics-based swarm intelligence*. Springer, 2012.

[20] H. G. Tanner and A. Kumar, "Formation stabilization of multiple agents using decentralized navigation functions," in *Robotics: Science and Systems*, 2005, pp. 49–56.

[21] V. Gazi, "Swarm aggregations using artificial potentials and sliding-mode control," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1208–1214, 2005.

[22] L. E. Barnes, M.-A. Fields, and K. P. Valavanis, "Swarm formation control utilizing elliptical surfaces and limiting functions," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 39, no. 6, pp. 1434–1445, 2009.

[23] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Tr. on Rob. and Autom.*, vol. 8, pp. 501–518, 1992.

[24] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[25] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[26] S. M. LaValle, "Motion planning: The essentials," *IEEE Robotics & Automation Magazine*, vol. 18, no. 1, pp. 79–89, 2011.

[27] J. Denny and N. M. Amato, "Toggle PRM: A coordinated mapping of C-free and C-obstacle in arbitrary dimension," ser. Springer Tracts in Advanced Robotics, vol. 86, 2013, pp. 297–312.

[28] H.-Y. Yeh, S. Thomas, D. Eppstein, and N. M. Amato, "UOBPRM: A uniformly distributed obstacle-based PRM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 2655–2662.

[29] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 469–482, 2010.

[30] O. B. Bayazıt, J.-M. Lien, and N. M. Amato, "Swarming behavior using probabilistic roadmap techniques," in *Swarm Robotics*. Springer, 2005, pp. 112–125.

[31] J. F. Harrison, C. Vo, and J.-M. Lien, "Scalable and robust shepherding via deformable shapes," in *Motion in Games*. Springer, 2010, pp. 218–229.

[32] A. Krontiris, S. Louis, and K. E. Bekris, "Multi-level formation roadmaps for collision-free dynamic shape changes with non-holonomic teams," in *IEEE International Conference on Robotics and Automation*, 2012.

[33] A. Wallar and E. Plaku, "Path planning for swarms by combining probabilistic roadmaps and potential fields," in *Towards Autonomous Robotic Systems*, ser. Lecture Notes in Computer Science, Oxford, UK, 2014, vol. 8069, pp. 1–12.

[34] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[35] D. Xie, M. Morales, R. Pearce, S. Thomas, J.-M. Lien, and N. M. Amato, "Incremental map generation (img)," in *International Workshop on Algorithmic Foundations of Robotics*, ser. Springer Tracts in Advanced Robotics, New York, NY, 2008, vol. 47, pp. 53–68.

[36] J. Van Den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *IEEE International Conference on Robotics and Automation*, 2006, pp. 2366–2371.

[37] D. Ferguson and A. Stentz, "Anytime rrts," in *IEEE International Conference on Robotics and Automation*, 2006, pp. 5369–5375.

[38] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt*," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 1478–1483.

[39] A. Wallar and E. Plaku, "Supplementary material for path planning for swarms in dynamic environments by combining probabilistic roadmaps and potential fields." [Online]. Available: https://github.com/wallarelvo/dCRoPS-SupplementaryMaterial

[40] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.