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

Alex Wallar Christopher Choi 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 pathways. 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 dynamic obstacles, the roadmap is searched again to provide alternative pathways. 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 in cohesion to a goal destination while avoding 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 (Combined Roadmaps and Potentials for Swarms for Dynamic Obstacles), can efficiently guide the

A. Wallar and C. Choi are 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 pathways. 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 introduces the idea of branching, which provides robots in the swarm with alternative pathways to the goal when a robot is unable to reach the planned intermediate goals due to interferences from dynamic obstacles. Weights of roadmap vertices and 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. dCRoPS also improves the APFs in order to enable the swarm to react rapidly to dynamic obstacles moving towards it. 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 $b_1, \ldots, b_n$ operating in a two-dimensional workspace W. Each robot is capable of detecting nearby obstacles. Starting at an initial placement, the objective is for each robot to reach a goal region $G \subset W$ while avoiding collisions with static and dynamic obstacles present in the workspace W. While branching is allowed, dCRoPS seeks to keep the robots moving as a swarm as much as possible, branching off only when necessary to avoid dynamic obstacles. The geometry and placement of each static obstacle is assumed to be available beforehand. No information is given to dCRoPS in regards to the dynamic obstacles.

III. Method

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

A. Global Path Planning

As mentioned, dCRoPS uses a roadmap to provide the swarm with global path planning and find alternative pathways to the goal when dynamic obstacles prevent the swarm from moving along the current pathway.

1) Roadmap Construction: The roadmap is represented as a graph $RM = (V, E)$. A vertex $q \in V$ corresponds to a collision-free point in the workspace. An edge $(q_i, q_j) \in E$ corresponds to a collision-free path 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. The roadmap is later repaired in response to interference by dynamic obstacles.

Algorithm 1 Pseudocode for dCRoPS

1: $RM = (V, E) \leftarrow$ ConstructRoadmap()
2: $w(q_i) \leftarrow$ assign weight to each vertex $q_i \in V$
3: $w(q_i, q_j) \leftarrow$ assign weight to each edge $(q_i, q_j) \in E$
4: $\zeta = [q_k]_{k=1}^{m} \leftarrow$ pathway$(RM, [w(q_i)], [w(q_i, q_j)])$
5: **for** $b \in$ *Robots* **do** $goal_b \leftarrow \zeta(1)$; $pathway_b \leftarrow \zeta$
6: **while** solved() $=$ `false` **do**
7: **for** $b \in$ *Robots* **with** reached$(b, goal_b) =$ `true` **do**
8: $goal_b \leftarrow$ NextGoal$(pathway_b, b)$
9: **for** $b \in$ *Robots* **with** stuck$(b) =$ `true` **do**
10: $pathway_b \leftarrow$ branching$(RM, [w(q_i)], [w(q_i, q_j)], b)$
11: $goal_b \leftarrow pathway_b(1)$
12: **for** $b \in$ *Robots* **do**
13: $pf_b \leftarrow$ superimpose$(PF_{obst}(b), PF_{sep}(b), PF_{goal}(b), PF_{heading}(b))$
14: **for** $b \in$ *Robots* **do**
15: $heading_b \leftarrow w_1 heading_b + w_2 pf_b$
16: $pos_b \leftarrow pos_b + heading_b$

In order to avoid cluttered or narrow-passage areas, which are more likely to lead to collisions with dynamic obstacles, dCRoPS generates roadmaps vertices that are at least a certain distance, d_{clear}, away from the static obstacles. Each roadmap vertex is generated by sampling a point at random inside the workspace W until it satisfies both of the above conditions.

A weight $w(q)$ is associated with each vertex $q \in V$ as an estimate on the feasibility of the swarm to pass near q. More specifically, $w(q)$ is defined as

$$w(q) = (\sum_{o \in StaticObstacles} dist(q, o))^3,$$

where $dist(q, o)$ denotes the distance from the point q to the obstacle o. Small values of $w(q)$ are indicative of nearby obstacles, which the swarm will seek to avoid. Note that while we have chosen through experimentation a particular definition of $w(q)$ for this paper, alternative weight functions can also be used.

Roadmap edges are generated by connecting each $q \in V$ to its k-nearest roadmap vertices according to the Euclidean distance. Collision checking is performed with respect to static obstacles, discarding any edge that is found in collision. 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. More specifically, $w(q_i, q_j)$ is defined as

$$w(q_i, q_j) = ||q_i, q_j||_2 / \min(w(q_i), w(q_j)),$$

As such, small values of $w(q_i, q_j)$ indicate that q_i and q_j are close to each other but away from obstacles.

2) Guiding the Swarm: A global pathway $\zeta = [q_i]_{i=1}^{m}$ (Algorithm 1:4) guides the swarm to the goal region. The pathway is computed using Dijkstra's algorithm to search the roadmap $RM = (V, E)$ for the shortest path from start to goal. Such pathway, as a result of edge weights, seeks to keep the swarm away from cluttered areas and narrow passages. Initially, each robot b is given ζ as the pathway that it needs

to follow (Algorithm 1:5). The objective for each robot is to move to $area(\zeta(1)), \ldots, area(\zeta(m))$ in succession, where $area(q)$ denotes a small area around q (defined as a disk with radius d_{clear}). As discussed later in the section, an attractive potential is used to move b from its current position to the next goal area along the pathway ζ (Algorithm 1:7–8).

3) Branching: If a robot b gets stuck in a local minima or a dynamic obstacle prevents it from reaching its next target $area(q_i)$ along its current pathway, dCRoPS will search the roadmap again to find an alternative pathway for b (Algorithm 1:9–10). Before the roadmap search, dCRoPS adjusts the weight of the next few edges $(q_i, q_{i+1}), (q_{i+1}, q_{i+2}), \ldots, (q_{i+\ell-1}, q_{i+\ell})$ on the current path in order to guide b away from this area. This local repair is used to relieve swarm congestion when it is confronted by an obstructing obstacle. The creation of a new pathway may not always make the swarm move faster to the end goal, but it reduces 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 the swarm follow the current global pathway 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 goal;
2) b is repulsed away from obstacles;
3) b seeks to move in cohesion with its neighbors without straying too far away or getting too close.

1) Attraction to the Next Goal: Let $goal_b$ denote the next goal for the robot b. To pull the robot to the goal, dCRoPS defines an attractive potential as follows:

$$PF_{goal}(b) = \frac{goal_b - pos_b}{1 + \exp(\delta_{goal}\,||goal_b, pos_b||_2)},$$

where δ_{goal} is a scaling constant. Although stronger potentials could be defined, using a weaker potential enables the robot to reach $goal_b$ while allowing for other behaviors (such as obstacle avoidance) to take priority. When $goal_b$ is reached, the next goal along $pathway_b$ is set as the target for b.

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

$$PF_{obst}(b, o) = \frac{pos_b - ClosestPoint(o, pos_b)}{\sqrt{|(dist(pos_b, o) - radius_b)|}}.$$

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.,

$$PF_{obst}(b) = \sum_{\substack{o \in Obstacles \\ dist(pos_b, o) \leq \Delta_{obst}}} PF_{obst}(b, o).$$

3) Moving in Cohesion with Neighbors: 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

$$PF_{sep}(b_i, b_j) = \frac{pos_{b_i} - pos_{b_j}}{1 + \exp(\delta_{sep}\,||pos_{b_i}, pos_{b_j}||_2)},$$

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.,

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

In order to make the robots move in cohesion, the heading of the neighbors also exert an influence on b. Preference is given to neighbors that are not stuck (as such neighbors can help b avoid obstacles), and are neither too far nor too close to b. More specifically, the potential function is defined as

$$PF_{heading}(b) = \sum_{b_i \in Neighs(b)} heading_{b_i},$$

where $Neighs(b)$ are selected as the k-closest nonstuck robots to b according to

$$\gamma(b, b_i) = \exp\left(\frac{-(||pos_b, pos_{b_i}||_2 - \mu)^2}{2\sigma^2}\right),$$

where $\gamma(b, b_i)$ is a Gaussian function with mean μ and standard deviation σ.

C. Combining the Potential Fields

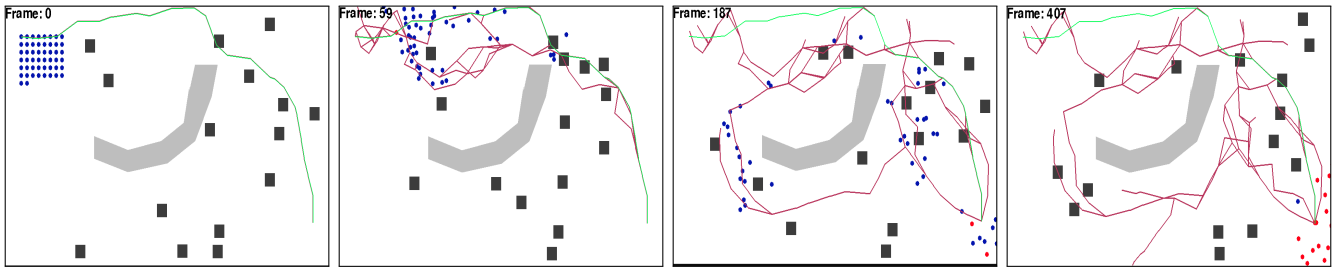
The overall effect on b is obtained by combining the four different potential fields:

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

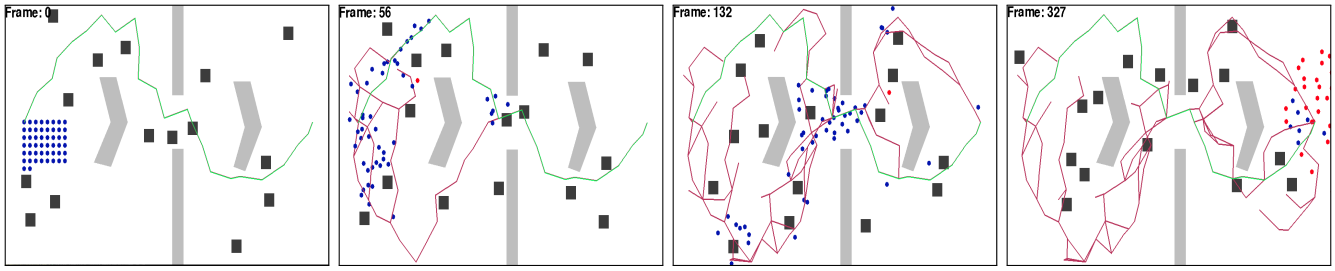
where $fields = \{goal, obst, sep, heading\}$. This overall force is then used to update the position and heading of b (Algorithm 1:15–16). In this way, b is attracted to the next goal, strongly repulsed from the obstacles, seeks to maintain a minimum separation distance from its neighbors without getting separated from the swarm, and moves in cohesion with its neighbors toward the next goal. When a robot becomes stuck, the roadmap is locally repaired a new alternative pathway 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 static and dynamic obstacles.

IV. Experiments and Results

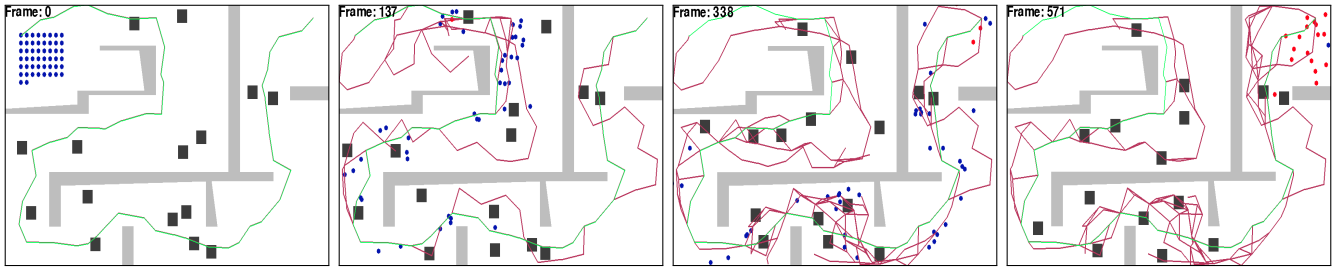
Experiments are conducted in simulation with an increasing number of robots moving in complex environments, seeking to avoid numerous static and dynamic obstacles. Results demonstrate the efficiency and scalability of dCRoPS.



(a) scene 1



(b) scene 2



(c) scene 3

Fig. 1. Illustration of dCRoPS on three different scenes. For each scene, four frames of a run of dCRoPS are shown. The first frame shows the swarm in the initial placement. The other three frames show how the swarm progresses towards the final goal. Static obstacles are shown in light gray. Dynamic obstacles are shown in dark grey. The initial pathway is shown in green (see first frame). Alternative pathways that are computed when some robots get stuck are shown in other colors.

A. Experimental Setup

An illustration of the three different scenes used in the experiments is provided in Fig. 1. 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 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, dynamic obstacles are made to move at random. More specifically, they can only traverse in four directions, up, down, left, or right. Each dynamic obstacle has a fixed change-direction threshold that is triggered when the dynamic obstacle has traveled along the current direction for a set distance. At that point, the dynamic obstacle decides uniformly at random to go left, right, up, or down. Robots have no influence on the direction or speed of dynamic obstacles. When a dynamic obstacle encounters a static obstacle, it starts traveling in the opposite direction.

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 as $5, 10, 15$. Results for each problem instance report median time based on ten different runs. Experiments are conducted on an Intel Core i7 machine (CPU: 2 GHz, RAM: 16GB) using OS X 10.9.1. Code is written in Python 2.7.3.

B. Results

Fig. 2 provides a histogram on the percentage of robots that reached the final goal during a particular time interval. The figure illustrates that due to interference from dynamic obstacles robots may reach the final goal at different times. As shown next, dCRoPS is quite efficient and scales linearly with the number of robots.

Fig. 3 provides a summary of the results when varying both the number of robots and the number of 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. As the results indicate, dCRoPS scales linearly with the number of

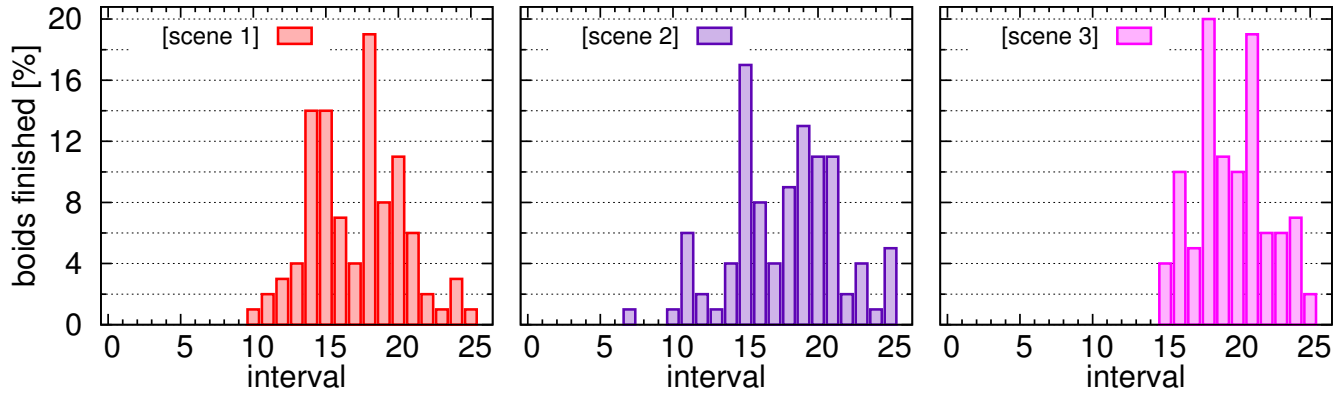


Fig. 2. Histogram on the time-to-reach final goal when running dCRoPS with 100 boids and 15 dynamic obstacles on each of the scenes. In each case, the overall time is divided into 25 equally-sized intervals. The i-th bar indicates the percentage of robots that reached the final goal during the i-th time interval.

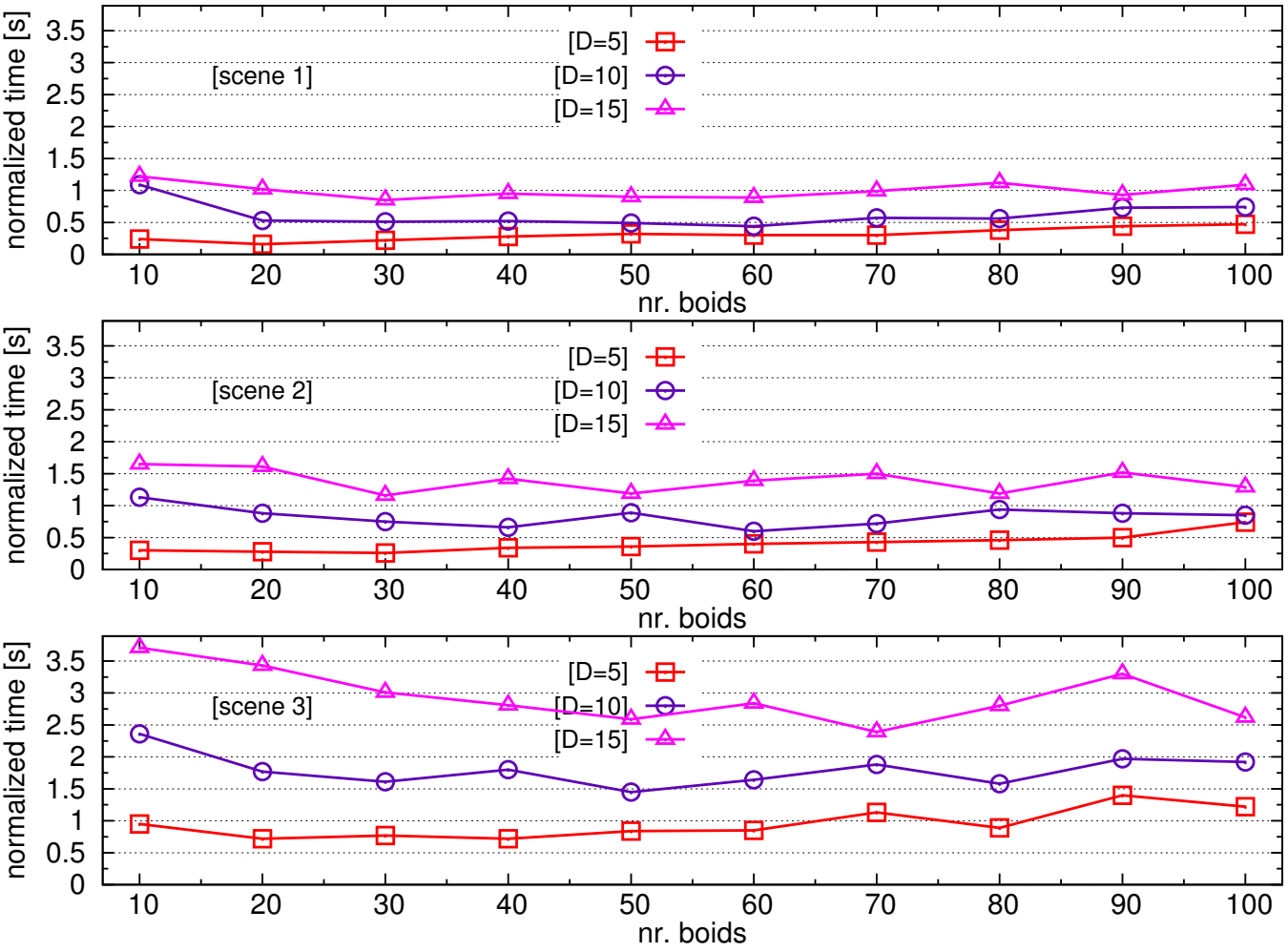


Fig. 3. Results show the normalized median runtime as a function of the number of robots and the number of dynamic obstacles ($D = \{5, 10, 15\}$). Runtime is measured as the time from the beginning till the last robot reaches the goal. Normalization corresponds to dividing the median runtime by the number of boids.

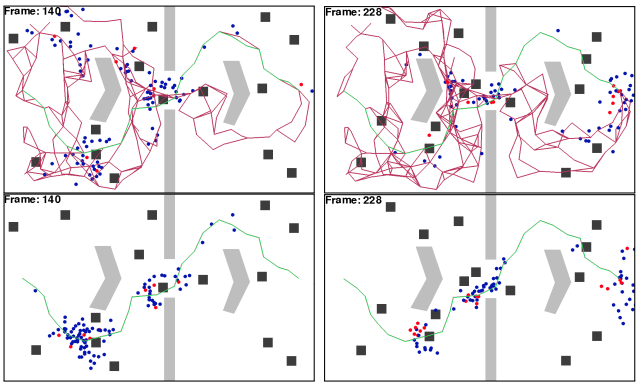


Fig. 4. Qualitative comparison of dCRoPS with (top) branching enabled vs (bottom) branching disabled.

[scene 1 with 10 dynamic obstacles]

nr. robots	10	50	100
branching on	10.87	26.14	73.83
branching off	12.12	76.33	151.28

[scene 2 with 100 robots]

nr. dynamic obstacles	5	10	15
branching on	74.37	84.98	128.67
branching off	141.59	138.09	231.57

TABLE I

IMPACT OF BRANCHING ON DCROPS. RESULTS SHOW MEDIAN TIME WHEN ENABLING OR DISABLING BRANCHING IN DCROPS

robots, even as the number of dynamic obstacles is increased.

By combining roadmaps with potential fields, dCRoPS provides the swarm with global path planning to guide toward the goal while reacting to obstacles encountered along the way, making local adjustment and seeking alternative pathways to avoid getting trapped. As such, branching plays a key role in the efficiency of dCRoPS. Fig. 4 provides a qualitative evaluation of the impact of branching. As shown, without branching, robots have a harder time finding a way around dynamic obstacles which prevent the swarm from following the current pathway.

A quantitative comparison of the impact of branching is provided in Table I. The results show that the time taken to navigate through the environment when the branching is turned off is larger than the time taken when the path branches. This is because of two things. Firstly branching 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 sets of swarms can go through different paths at the same time. The second reason branching 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. For example, if a dynamic obstacle obstructs a narrow passageway and the shortest path goes through that passageway, the robots will not be able to reach the end until the obstacle moves out of the way if the path does not branch. However, if the path branches, it is able to find a new way around the obstacle, therefore avoiding the obstructed passageway. This reduces the amount of time the swarm spends waiting and instead finds a new way to the goal. This, of course, reduces the time it takes to get to the goal.

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. A key component of dCRoPS is the branching aspect which provides alternative pathways 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 dynamic obstacles and incorporate those predictions to adjust the roadmap so that it can provide alternative pathways that avoid predicted trajectories. Improving the roadmap quality may 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*, Oxford, UK, 2013, in press.