Auton Robot (2014) 37:401–415
DOI 10.1007/s10514-014-9412-1

Goal assignment and trajectory planning for large teams of interchangeable robots

Matthew Turpin · Kartik Mohta · Nathan Michael · Vijay Kumar

Received: 4 October 2013 / Accepted: 14 July 2014 / Published online: 8 August 2014

Abstract This paper presents Goal Assignment and Planning: a computationally tractable, complete algorithm for generating dynamically feasible trajectories for N interchangeable (identical) robots navigating through known cluttered environments to M goal states. This is achieved by assigning goal states to robots to minimize the maximum cost over all robot trajectories. The computational complexity of this algorithm is shown to be polynomial in the number of robots in contrast to the expected exponential complexity associated with planning in the joint state space. This algorithm can be used to plan trajectories for dozens of robots, each in a potentially high dimensional state space. A series of planar case studies are presented and finally, experimental trials are conducted with a team of six quadrotor robots navigating in a constrained three-dimensional environment.

Keywords Multiple robot · Trajectory planning · Task assignment · Micro aerial robot

1 Introduction

This paper addresses the problem of simultaneously finding optimal trajectories and assignments of goals to robots in a setting where robots are identical, and in which it is desirable to minimize the maximum cost over all robot trajectories. This problem is relevant to missions with many tasks that have to be performed as quickly as possible and the tasks can be performed in parallel. In such settings, the cost function may need to reflect the maximum effort over all robots or the the maximum distance traveled by a robot. This is particularly important in first-response and search-and-rescue applications in which a number of robots must visit, for example, all of the rooms in a building. A constraint on the fuel or energy that can be spent per each robot on the mission leads quite naturally to a setting where we want to minimize the maximum cost. In such conditions, we wish to ensure the existence or non-existence of solution given the current system state and thus require that the planning methodology be complete (LaValle 2006). We would like the algorithm to find a solution when one exists, even though the solution may not be the optimal time solution. Finally, it is necessary to find time parameterized trajectories that are safe. This entails respecting robot dynamics and avoiding collisions with the environment and with other robots.

Optimal trajectory planning for multiple robots with collision avoidance can be performed by a conventional path planner simply by planning in the joint state space. Of course, the computational complexity typically grows exponentially with the number of robots (Erdmann and Lozano-Perez 1987). One approach studied to counter this growth is to decouple the path planning from the speed at which each path is traversed (Erdmann and Lozano-Perez 1987; Kant and Zucker 1986). Similarly, it is possible to plan trajectories using random prioritization (Van Den Berg and Overmars 2005) or more intelligent sequencing (van Den Berg et al. 2009). As decoupled approaches are generally not complete, several approaches seek to balance coupled and decoupled formulations in pursuit of completeness with computational

M. Turpin (✉) · K. Mohta · V. Kumar
University of Pennsylvania, Philadelphia, PA, USA
e-mail: mturpin@seas.upenn.edu

K. Mohta
e-mail: kmohta@seas.upenn.edu

V. Kumar
e-mail: kumar@seas.upenn.edu

N. Michael
Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: nmichael@cmu.edu

tractability (LaValle and Hutchinson 1998). This algorithm presented in this paper also which makes use of decoupled path planning; however, the proposed approach is complete.

Computational techniques such as subdimensional expansion (Wagner and Choset 2011; Wagner et al. 2012) or conflict based search (Sharon et al. 2012) can be employed to mitigate increased computational complexity by selectively expanding the higher-dimensional state space as required. Other approaches switch robot positions until robots can easily navigate to goal positions without collision (Peasgood et al. 2008; Luna and Bekris 2011), however these do not incorporate interchangeability and can yield poor trajectories.

An alternative strategy is to formulate multi-robot coordination and collision avoidance as reactive control or local coordination problems. Both navigation functions (Loizou and Kyriakopoulos 2008) and the decentralized collision avoidance method proposed in Van Den Berg et al. (2011) enable a robot team to navigate to an assigned set of goals and scale well with the number of agents but lack safety and optimality guarantees for systems with higher-order dynamics.

We approach the problem as one of task allocation as developed in the operational research community in the context of finding an optimal assignment of workers to goals (Kuhn 1955) . This has recently been applied to multi-robot task allocation (Gerkey and Matarić (2004)) and trajectory planning (Kloder and Hutchinson 2006; Liu and Shell 2011; Turpin et al. 2012; Liu and Shell 2012). Alternatively, when some agents are interchangeable, the k-color planning problem can be considered (Solovey and Halperin 2014). As shown in Turpin et al. (2012); Yu and LaValle (2012), coupling goal assignment with trajectory planning paradoxically reduces the complexity.

This paper, following the authors' previous work (Turpin et al. 2013a,b), considers the problem of simultaneously planning trajectories and goal assignments for interchangeable robots with initial and final deployment states at rest in an environment with static obstacles. Section 2 introduces notation, assumptions, and the requirements of the system. The Goal Assignment and Planning (GAP) algorithm is presented in Sect. 3. Section 4 presents a number of case studies to plan minimum distance safe trajectories for teams of two-dimensional kinematic robots. The performance of the algorithm is studied in simulation for large teams in complex and cluttered environments and experimentally with a team of six quadrotor vehicles in Sect. 5.

In addition to numerous updates and clarifying remarks throughout, the major contributions of this paper are as follows. While the GAP algorithm has been previously presented in Turpin et al. (2013b), Sect. 3 builds on that work by providing in-depth proofs of completeness and the guarantees of the GAP algorithm. Section 4 has been added to demonstrate how the GAP algorithm can be applied to a team of robots in practice.

2 Preliminaries

Let the set of natural numbers between 1 and Z be represented by $\mathcal{I}_Z \equiv \{1, 2, \ldots, Z\}$. Designate the state of robot $i \in \mathcal{I}_N$ as $x_i \in \mathcal{X}$ where $\mathcal{X}$ is the state space of a single robot. Define the initial state of robot i as $s_i = x_i(0)$. Similarly, g_j specifies the $j \in \mathcal{I}_M$ desired goal state.

The set of all points in Euclidean space occupied by robot i is represented by the open set $B(x_i) \subset \mathbb{R}^d$, where d is the dimensionality of the robot extent. With slight abuse of notation, $B(\tau) \subset \mathbb{R}^d$ is the set of all points swept out by a robot tracking trajectory $\tau : t \to \mathcal{X}$.

It should be stressed at this point that the algorithm will only work when states s_i and g_j have self-loops. In other words, it is assumed that robots have the ability to remain at states s_i and g_j indefinitely. This requirement may preclude the use of fixed wing aerial vehicles that require forward velocity to remain in flight.

The assignment of robots is represented by the mapping $\phi : \mathcal{I}_N \to \mathcal{I}_M \cup 0$ where each goal can be assigned to a maximum of one robot such that:

$$\phi_i = \begin{cases} j & \text{if robot } i \text{ is assigned to goal } j \\ 0 & \text{otherwise} \end{cases}$$

3 Algorithm

The Goal Assignment and Planning (GAP) algorithm presented in Algorithm 1 synthesizes assignment and trajectory generation methods to generate collision-free trajectories for a large number of robots. It is assumed that each vehicle is locally executing a trajectory tracking controller and that trajectories are dynamically feasible.

Algorithm 1 Goal Assignment and Planning (G, s, g)

```
1: for $i \in \mathcal{I}_N$ do
2:     for $j \in \mathcal{I}_M$ do
3:         Compute $\gamma_{ij}(t)$, optimal trajectory from $s_i$ to $g_j$
4: Compute $\phi^*$, the optimal assignment of goals to robots
5: for $i \in \mathcal{I}_N$ do
6:     for $j \in \mathcal{I}_N \backslash i$ do
7:         if $s_i \in P^\star(s_j, g_{\phi_j^\star})$ then
8:             Assign partial order $\mathcal{P}$: $j \succ i$
9:         if $g_i \in P^\star(s_j, g_{\phi_j^\star})$ then
10:            Assign partial order $\mathcal{P}$: $i \succ j$
11: Construct suitable total ordering $\psi$ from partial ordering $\mathcal{P}$
12: Optional refinement of $\gamma_{i\phi_i^\star}(t)$. See Sect. 3.6
13: for $i = 1 \to N$ do
14:     Compute $\hat{t}_{\psi_i}$, time offset of robot with priority $i$
15: return trajectories $\gamma_{i\phi_i^\star}(t - \hat{t}_{\psi_i})$
```

Section 3.1 details the generation of graph G, which is embedded in $\mathcal{X}$, the state space of a single robot. GAP then computes the cost of the optimal paths for each robot to every goal in the absence of other robots, a process discussed in Sect. 3.2. Section 3.3 defines the optimal assignment as that which solves the Lexicographic Bottleneck Assignment Problem. Next, Sect. 3.4 introduces the notion of prioritization of robots, which is always possible due to the minimum maximum cost property of the optimal assignment. In Sect. 3.5, trajectories are computed that satisfy robot dynamics and avoid collisions with the environment and other robots. Optional trajectory refinement is presented in Sect. 3.6. Finally, completeness of Algorithm 1 is shown in 3.7 and a complexity analysis is performed in Sect. 3.8.

3.1 Graph Generation

A directed graph $G = (V, E)$ is constructed by sampling the state space using either a deterministic or probabilistic planner. A planner which is resolution complete will find a solution if one exists if the resolution of the underlying graph is sufficiently fine. Similarly, probabilistic completeness signifies that the probability of a random sampling motion planner will fail to return a path if one exists approaches zero as the number of samples goes to infinity. Although this paper only presents proof of resolution completeness, GAP also permits probabilistically complete planning.

Each vertex $v_i \in V$ represents a valid, collision-free state in $\mathcal{X}$ and V must contain all starting and goal states:

$$s_i \in V \;\forall\, i \in \mathcal{I}_N, \quad g_j \in V \;\forall\, j \in \mathcal{I}_M \tag{1}$$

An edge $e_{ij} \in E$ corresponds to a collision-free trajectory segment $\tau_{ij}(t)$ where $\tau_{ij}(t_0) = v_i$, $\tau_{ij}(t_f) = v_j$ and has cost $C(\tau_{ij}) > 0$. The underlying dynamics of the system will determine how to best compute these segments. Possible cost functions C include distance traveled, energy used, and trajectory duration.

The graph G cannot include edges e_{kl} corresponding to trajectories τ_{kl} that would result in a collision with a robot at states s_i or g_j unless one of the vertices v_k or v_l is s_i or g_j:

$$\begin{aligned} v_k \neq s_i \text{ AND } v_l \neq s_i &\implies B(\tau_{kl}) \cap B(s_i) = \emptyset \\ v_k \neq g_i \text{ AND } v_l \neq g_i &\implies B(\tau_{kl}) \cap B(g_i) = \emptyset \end{aligned} \tag{2}$$

For example, a regular orthogonally-connected grid for spherical robots satisfies Eq. (2).

A path in the graph G is represented by an ordered list of its edges, $P(v_i, v_j) = e_{ik}, e_{kl} \ldots, e_{mn}, e_{nj}$. The cost of a path in the graph is represented using the norm:

$$||P(v_i, v_j)|| = \sum_{e_{kl} \in P(v_i, v_j)} C(\tau_{kl})$$

The optimal path is that which minimizes the sum of costs of edges between vertices:

$$P^\star(v_i, v_j) = \underset{P(v_i, v_j)}{\arg\min} \;\; ||P(v_i, v_j)||$$

The corresponding trajectory for a path from the graph can be constructed by the concatenation of trajectory segments. For example, $\gamma_{ij}(t)$ is the optimal trajectory from s_i to g_j and can be written as the sequence: $\gamma_{ij} = \{\tau_{ik}, \tau_{kl}, \ldots, \tau_{mn}, \tau_{nj}\}$.

Sufficient conditions for resolution completeness To ensure resolution completeness, it is critical that a path generated by the single robot planning algorithm reflect the possibilities of the full muti-robot system. This section presents one set of sufficient conditions that ensure the resolution completeness of GAP.

First, define Y_i as the set of all valid states that, if occupied by a robot, would be in collision with a robot at s_i. Similarly, define W_i as the set of all valid states that, if occupied by a robot, would be in collision with a robot at g_i. It will be shown that GAP is resolution complete if the following conditions are respected. First, every valid state in Y_i must be reachable from s_i without colliding with a robot at s_j, for all $i \neq j$. A related constraint is that s_i must be reachable from every valid state in Y_i without colliding a robot at s_j, for all $i \neq j$. Additionally, these constraints must also hold for the set of goal conditions.

Mathematically, these conditions can be represented by the following sets of equations:

$$\begin{aligned} &\forall v_j \in Y_i \;\exists \tau_{ij} \,\big|\, v_i = s_i,\, B(\tau_{ij}) \cap B(s_k) = \emptyset \;\forall\, k \in \mathcal{I}_N \backslash i \\ &\forall v_j \in Y_i \;\exists \tau_{ji} \,\big|\, v_i = s_i,\, B(\tau_{ji}) \cap B(s_k) = \emptyset \;\forall\, k \in \mathcal{I}_N \backslash i \end{aligned} \tag{3}$$

$$\begin{aligned} &\forall v_j \in W_i \;\exists \tau_{ij} \,\big|\, v_i = g_i,\, B(\tau_{ij}) \cap B(g_k) = \emptyset \;\forall\, k \in \mathcal{I}_M \backslash i \\ &\forall v_j \in W_i \;\exists \tau_{ji} \,\big|\, v_i = g_i,\, B(\tau_{ji}) \cap B(g_k) = \emptyset \;\forall\, k \in \mathcal{I}_M \backslash i \end{aligned} \tag{4}$$

An intuition for these conditions is that a robot at its start or goal state should not modify the reachable configuration space of other robots. For kinematic fully actuated robots, conditions (3) and (4) reduce to:

$$\begin{aligned} &Y_i \cup Y_j \;\emptyset\; \forall i \neq j \\ &W_i \cup W_j \;\emptyset\; \forall i \neq j \end{aligned}$$

See Fig. 1 for an example of initial conditions which violate the assumption in Eq. (3) and a slightly modified environment which respects this condition.

3.2 Computing individual trajectories

After construction of graph G, the graph is searched for the minimum cost path for all NM combinations of robots and goal states. Dijkstra's algorithm (Dijkstra 1959) is a natural choice for this graph search as it returns optimal paths in the graph from the start vertex of a robot to all other vertices.

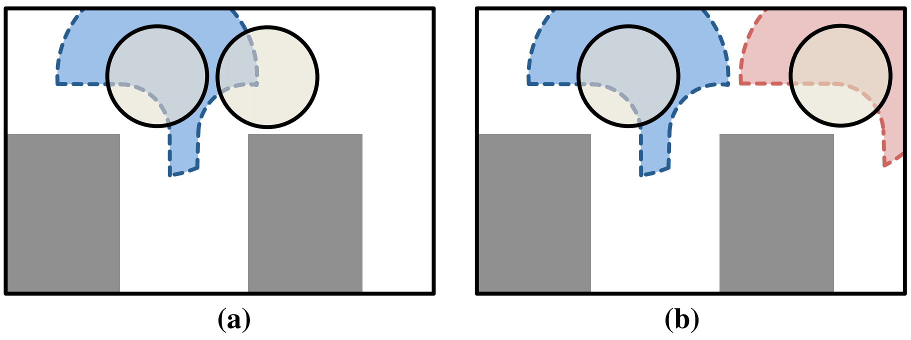


Fig. 1 In each of **a**, **b**, example initial locations for two-dimensional circular robots are shown. *Tan circles* represent the initial states of the two robots and the *gray rectangular* regions indicate obstacles. The *dotted* regions represent Y. **a** shows an example of poor initial conditions which do not satisfy Eq. (3). This is clear since to reach the rightmost side of Y for the robot on the *left* results in a collision with the robot on the *left*. **b** slightly modifies the initial conditions such that there exists a valid trajectory from the starting location to every point in the region Y and therefore this initial condition is valid

After the optimal path through the graph is found, full robot trajectories can be constructed from the trajectory segments τ_{kl} corresponding to $e_{kl} \in E$ to form $\gamma_{ij}(t)$ for robot i such that $\gamma_{ij}(t_{i,0}) = s_i$ and $\gamma_{ij}(t_{i,f}) = g_j$. Each robot will remain at its starting vertex when $t < t_{i,0}$ and its goal vertex if $t > t_{i,f}$.

If the graph search finds that goal g_j is unreachable from start location s_i, the cost $C(\gamma_{ij})$ is infinite.

3.3 Assignment

The optimal assignment $\phi^\star$ is defined as that which solves the lexicographic bottleneck assignment problem (LexBAP) (Burkard and Rendl 1991). Similar to the standard bottleneck assignment problem, the LexBAP solution finds the feasible solution that minimizes the largest cost. Additionally, the LexBAP minimizes the second highest cost, as well as the third highest, and so on. Using this cost function ensures any pairwise exchange of assignment between robots i and j will not result in a decrease of maximum trajectory cost.

$$\begin{aligned}&\max\left(||P^\star(s_i, g_{\phi_i^\star})||, ||P^\star(s_j, g_{\phi_j^\star})||\right)\\&\quad\le \max\left(||P^\star(s_i, g_{\phi_j^\star})||, ||P^\star(s_j, g_{\phi_i^\star})||\right)\end{aligned} \tag{5}$$

The best solutions to the LexBAP have computationally complexity of $\mathcal{O}(N^4)$ (Sokkalingam and Aneja 1998).

In practice, it is possible to relax the optimal LexBAP and use the solution of the linear task assignment problem. This can be solved using the well-known Hungarian algorithm (Kuhn 1955) with bounded computational complexity of $\mathcal{O}(N^3)$. The modified cost function minimizes the p-norm of the costs incurred by the team:

$$\phi^* \equiv \underset{\phi}{\operatorname{argmin}} \left(\sum_{i \in \mathcal{I}_N} ||P(s_i, g_{\phi_i}||^p\right)^{\frac{1}{p}}, \tag{6}$$

where p is sufficiently large such that the minimum-maximum cost principle in (5) holds. The solution to the modified linear assignment problem can be checked in $\mathcal{O}(N^2)$ time to determine if the assignment in (6) satisfies (5). Ideally, p is as large as possible, but unfortunately the size of p is bounded due to numerical issues. If (5) is not satisfied, the $\mathcal{O}(N^4)$ LexBAP solution can be utilized. In all of the simulation and experimentation trials used, a value of $p = 50$ was sufficient to satisfy (5) and the LexBAP solver was never required.

Recall that $\phi_i = 0$ signifies that robot i in not assigned to visit any goal location. The cost of a path to g_0 is defined to be a value larger than any feasible cost. This choice will force as many robots as possible to navigate to a goal state. Excess goals will not be assigned to any robot through this assignment procedure and Algorithm 1 must be used iteratively to ensure all goals are eventually visited.

3.4 Prioritization

The prioritization stage induces an ordering among all of the robots:

$$s_i \in P^\star(s_j, g_{\phi_j^\star}) \implies i \prec j \quad \forall i \neq j \tag{7}$$

$$g_i \in P^\star(s_j, g_{\phi_j^\star}) \implies i \succ j \quad \forall i \neq j \tag{8}$$

Theorem 1 *The conditions in Eqs.* (7) *and* (8) *induce partial ordering* $\mathcal{P}$.

Proof Appendix 1 contains proofs of the antisymmetry and transitivity of Eqs. (7), (8) in Lemma 2 and Lemma 3, respectively. Combining this result with the inherent reflexivity, the conditions in Eqs. (7) and (8) induce a valid partial ordering.□

The next step is to construct a total ordering which respects all relations of the partial ordering $\mathcal{P}$. In the case of an ambiguity in the partial ordering, the cost of the trajectories can be used as a tie breaker when constructing this total ordering.

This total ordering can then be represented by the mapping $\psi : \mathcal{I}_N \rightarrow \mathcal{I}_N$ where ψ_i identifies the robot with priority i.

3.5 Parameterization and collision avoidance

At this point, feasible trajectories have been computed for every robot-goal pair (γ_{ij}) with robots either assigned to a goal destination $\phi_i^\star = j \in \mathcal{I}_M$ or stationary $\phi_i^\star = 0$, and a prioritization order (ψ_i). The trajectories $\gamma_{i\phi_i^\star}(t)$ will navigate the robot from the initial state s_i to the final assigned state $g_{\phi_i^\star}$. However at this point, there may be collisions between robots. This section details a reparametrization of the planned trajectories, making use of the resting boundary conditions assumption.

Similar to greedy direct sequential routing in Schüpbach and Zenklusen (2011), robots are systematically assigned time offsets, $\hat{t} \geq 0$, to avoid collision with robots of higher priority. The robot with highest priority (ψ_1) begins with an offset of $\hat{t}_{\psi_1} = 0$. Then, the robot with the second highest priority (ψ_2) computes an offset such that its trajectory never collides with the robot with highest priority. It is usually best to minimize the offset times:

$$\hat{t}_{\psi_2} = \arg\min_{\hat{t}} \; B(\gamma_{\psi_2,\phi_{\psi_2}}(t-\hat{t})) \cap B(\gamma_{\psi_1,\phi_{\psi_1}}(t-\hat{t}_{\psi_1})) = \emptyset \tag{9}$$

Similarly, the robot with the third highest priority then finds the minimum time offset which results in a collision-free trajectory between it and all other robots with higher priority. This continues with each robot requiring knowledge of the full trajectory of all robots with higher priority than itself until all offset times are computed, $\hat{t}_i \; \forall\, i \in \mathcal{I}_N$. Lemma 1 demonstrates that such a time offset always exists to avoid collision with all robots.

Lemma 1 *Time offset $\hat{t}_i$ exists for all robots such that no collisions occur for robots to navigate to their assigned goal location.*

Proof The robot with highest priority will be able to have a time offset of $\hat{t}_{\psi_1} = 0$. This robot will be able to navigate to its goal location using the trajectory corresponding to its assignment without colliding with any robots at their starting locations. If a collision were going to occur between robot ψ_1 and robot j remaining at its starting state, using Eqs. (2) and (7) dictates that robot j would have higher priority and contradicting the fact that robot ψ_1 has highest priority.

Robot ψ_2 can find time offset $\hat{t}_{\psi_2} \geq 0$. Using a similar logic to that of robot ψ_1, robot ψ_2 will not collide with any robots having lower priority if they are to remain at their initial state. If $\hat{t}_{\psi_2}$ is large enough, robot ψ_1 will have arrived at its goal location. In this case, either robot ψ_2 cannot collide with robot ψ_1 or the condition in Eq. (2) results in a contradiction with Eq. (8).

Existence of time offsets for all other robots can be found by extension. □

In the worst case, Lemma 1 would have only one robot moving at a time, in order of priority. In practice, trajectories will complete much more quickly than this bound.

3.6 Optional trajectory refinement

While the algorithm presented thus far in Sect. 3 is very general for dynamic systems, systems with complex dynamics may require trajectory planning in the complete state space which may be very difficult, time consuming, or memory intensive to perform. Therefore, this section presents an optional method to refine trajectories, allowing a lower resolution graph to be constructed or even the use of a lower dimensional robot state. Examples of trajectory refinement will be presented in Sects. 4 and 5.

An assigned trajectory $\gamma_{i\phi_i^\star}$ can be optionally refined to $\tilde{\gamma}_{i\phi_i^\star}$ immediately preceding calculation of time parameterization in Sect. 3.5.

There are four requirements for a valid trajectory refinement. First, the refined trajectories must be dynamically feasible and not cause a collision with any obstacle. Next, refined trajectories must satisfy the boundary conditions s_i and g_i. Third, refined trajectories must respect the constraints specified by the computed total ordering ψ:

$$\psi_i > \psi_j \implies B(\tilde{\gamma}_{i\phi_i^\star}) \cap B(g_j) = \emptyset$$
$$\psi_i < \psi_j \implies B(\tilde{\gamma}_{i\phi_i^\star}) \cap B(s_j) = \emptyset$$

Finally, the refinement must decrease the cost of the trajectory for an individual robot:

$$C(\tilde{\gamma}_{i\phi_i^\star}) < C(\gamma_{i\phi_i^\star}) = \sum_{e_{ij} \in P^\star(s_i, g_{\phi_i^\star})} C(\tau_{ij})$$

If these conditions are met, the results from Theorem 2 continue to apply and collision avoidance is guaranteed. The basic steps of trajectory refinement are shown in Fig. 2.

3.7 Completeness

Theorem 2 GAP*, as presented in Algorithm 1, is resolution complete and returns collision-free trajectories tasking as many robots to goal states as possible.*

Proof Resolution completeness requires finding a solution if one exists and otherwise correctly returning that there is no solution.

By design, all trajectories are dynamically feasible and do not result in robot-obstacle collision. The assumptions in Eqs. (3) and (4) specify a robot at either a start or goal state

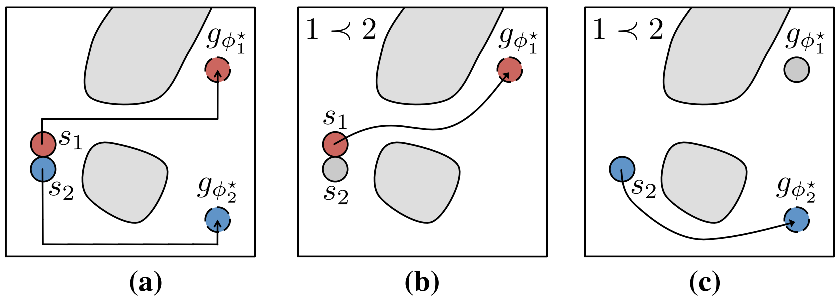


Fig. 2 Optional refinement of trajectories. **a** shows optimal initial trajectories and **b** shows trajectory refinement of a higher priority robot. The robot must avoid collisions with the starting states of all robots with lower priority. **c** shows trajectory refinement of a lower priority robot. This robot must avoid goal states of higher priority robots

cannot change the topology of the configuration space of any robot. Combining this with the fact that Dijkstra's algorithm is resolution complete implies that if there is a solution for robot i to reach goal g_j in the coupled planning problem, it will be found using the single robot planner in a sufficiently sampled graph. If there is not a trajectory for any robot to reach goal g_j in the coupled planning problem, clearly the single agent planner will not find it.

The optimal task assignment presented in Sect. 3.3 ensures that as many goals as possible will be visited. The partial ordering imposed in (7) and (8) guarantees that a valid total ordering is always found for a given assignment. Finally, Lemma 1 states that a set of valid time offset exists for a valid total ordering.

Therefore, the GAP algorithm is resolution complete. □

3.8 Complexity analysis

This section will analyze the computational complexity of the presented algorithm in terms of number of the robots, N, the number of goals, M, the number of vertices in the graph, $|V|$, and the number of edges in the graph, $|E|$. Sect. 3.2 outlines the systematic planning of individual robot trajectories and makes the assumption that G has already been constructed.

To search for the optimal solution using Dijkstra's algorithm using a min priority queue for each start location to all vertices has complexity bound of $\mathcal{O}(|E| + |V|\log|V|)$ (Fredman and Tarjan 1987). Therefore, finding paths from all starting locations to all goal locations has bounded complexity of $\mathcal{O}(N(|E|+|V|\log|V|))$. As discussed in Sect. 3.3, LexBAP can be solved in $\mathcal{O}(\max(N, M)^4)$. Although, typically, the linear assignment problem solution will be used with a complexity bound of $\mathcal{O}(\max(N, M)^3)$. The prioritization scheme designed in Sect. 3.4 considers pairwise interactions between robots and therefore grows with a complexity bound of $\mathcal{O}(N^2)$. Computing time offsets in Sect. 3.5 grows with complexity bound $\mathcal{O}(N^2)$. The optional refinement step considers robots one at a time and has complexity bound $\mathcal{O}(N)$. Therefore, the worst case complexity of this algorithm is $\mathcal{O}(\max(N, M)^4 + N(|E| + |V|\log|V|))$. This computational complexity is quartic in the number of robots or goals, which will tend to dominate the solution time at high N or M. However, in practice, the bound tends to be dominated by the $\mathcal{O}(\max(N, M)^3)$ Hungarian algorithm, which is the best known complexity bound for the optimal solution to the assignment problem. To visualize these bounds, Fig. 15 shows the computation times for a large number of simulated trials and the trends as the number of robots changes.

4 Case study: minimum distance paths for two-dimensional robots

In this section, the problem of navigating a team of circular interchangeable two-dimensional robots is considered. This section, although considering a similar problem to that in Turpin et al. (2013a), is intended to give the reader a set of simple examples on which to extend for more complicated robots. Section 4.1 explores a simple example to show explicitly how each step of Algorithm 1 is computed. In Sect. 4.2, movement restrictions are lifted to show how the initial condition requirements affects the allowable graphs. For this system, a deterministic grid and a Probabilistic Road Map are compared. In Sect. 4.3, the unique minimum distance solution is considered as well as how to generate feasible trajectories for a robot with second order dynamics.

4.1 Explicit example for a team of robots with constrained motion

This section considers a team of interchangeable circular kinematic robots with diameter 1 that are restricted to move along a unit length four-connected grid. The state is represented as a two-dimensional position vector, $\mathcal{X} = \mathbb{R}^2$ and the space occupied by each robot is a disk of radius $1/2$. Additionally, all initial and goal locations for this system fall on a regular grid of edge length 1. The edges between neighboring cells have corresponding straight lines trajectories with constant velocity:

$$\tau_{ij}(t) = \left(v_i - v_j\right)\left(\frac{t - t_0}{t_f - t_0}\right)$$

For motivation for this system, see the Kiva Systems mobile fulfillment solution (Enright and Wurman 2011).

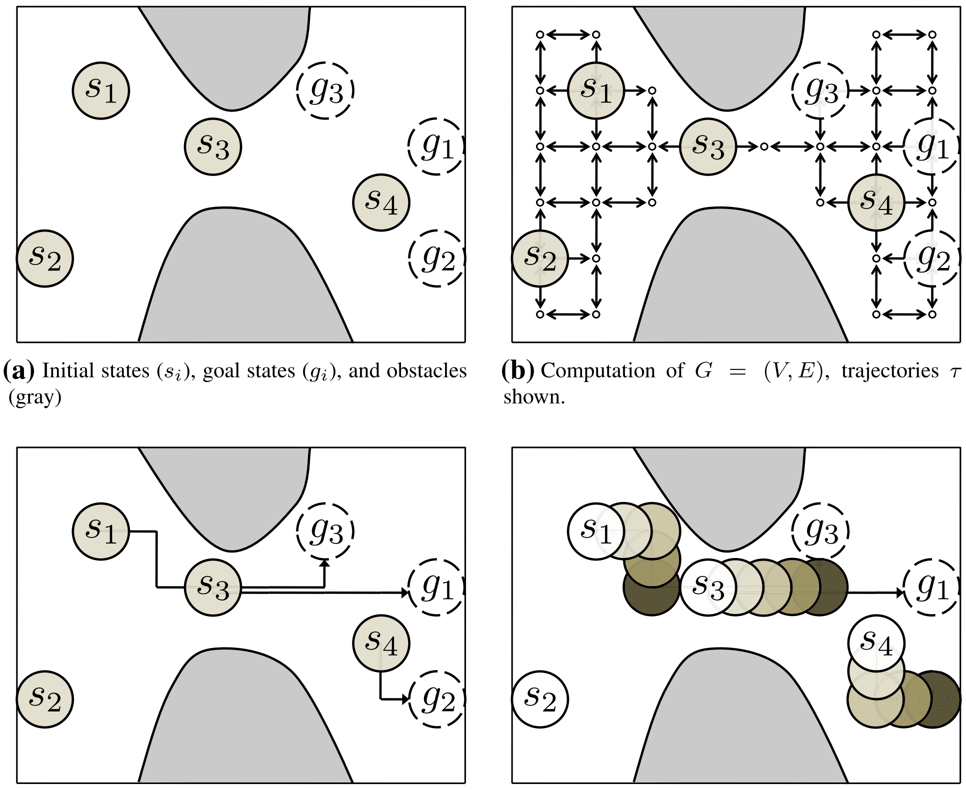


Fig. 3 Key algorithmic steps for a robotic team of $N = 4$ circular robots navigating to $M = 3$ goal states

(a) Initial states (s_i), goal states (g_i), and obstacles (gray)

(b) Computation of $G = (V, E)$, trajectories τ shown.

(c) Optimal assignment trajectories ($\phi^\star$)

(d) Snapshots of trajectory tracking at $t = 0.5, 1.0, 1.5, 2.0$

It is critical to ensure that this system respects the requirements specified in Sect. 2. Since all vertices fall along the grid, Eq. (1) is necessarily satisfied. The graph only contains orthogonal connections, resulting in each trajectory only causing collision with a vertex if that one of the endpoints of that trajectory are the vertex in the collision state. As this is satisfied for all vertices, it is satisfied for initial and goal vertices and satisfies Eq. (2). As all possible vertices and edges are included, Eqs.(3) and (4) are not required for resolution completeness.

The edge costs are defined as the distance traveled:

$$C(\tau_{ij}) = ||v_i - v_j||_2 = D = 1$$

Now, consider the $N = 4$ robot system with $M = 3$ goals depicted in Fig. 3a. The graph and corresponding trajectories are constructed making use of the regular grid as shown in Fig. 3b. Then the optimal path through the graph is planned for each robot to every goal, resulting in 12 trajectories. The trajectory costs returned for each of the paths through the graph in Fig. 3b are:

$$\left[\, ||P^\star(s_i, g_j)||\right] = \begin{bmatrix} 7 & 9 & 6 \\ 9 & 11 & 8 \\ 4 & 6 & 3 \\ 2 & 2 & 3 \end{bmatrix}$$

The optimal assignment assigns robot 1 to goal 3, robot 3 to goal 1, and robot 4 to goal 2, or $\phi_1^\star = 3, \phi_2^\star = 0, \phi_3^\star = 1, \phi_4^\star = 2$. The trajectories correlating to these assignments are plotted in Fig. 3c.

Notice how in Fig. 3c, the starting location of robot 3 is in the path of robot 1, or $s_3 \in P(s_1, g_{\phi_1^\star})$. Therefore, according to (7), robot 3 has priority over robot 1, or a partial order is induced of $1 \succ 3$. A valid total ordering might be $\{3, 2, 1, 4\}$. In this case, $\psi_1 = 3$, $\psi_2 = 2$, $\psi_3 = 1$, and $\psi_4 = 4$. Now, robot ψ_1, or robot 3 assigns its time offset of $\hat{t}_3 = 0$. Then robot $\psi_2 = 2$ assignes its time offset, which can also be 0. Next, robot $\psi_3 = 1$ assigns its offset time, which can be $\hat{t}_1 = 0$ as well since the trajectory will not lead to a collision even with zero offset. Finally robot $\psi_4 = 4$ plans $\hat{t}_4 = 0$.

4.2 Constrained movement relaxation

At this point, we relax the assumption that all boundary locations fall on a unit-spaced grid and allow robots to move in any direction. A much greater variety of systems are now able to be considered compared to those in Sect. 4.1.

Next, consider the following conditions on the construction of the graph such that it satisfies Eqs. (2), (3), and (4) for this fully actuated system.

First, the position of all starting locations are required to be at least $4R$ from any other starting condition:

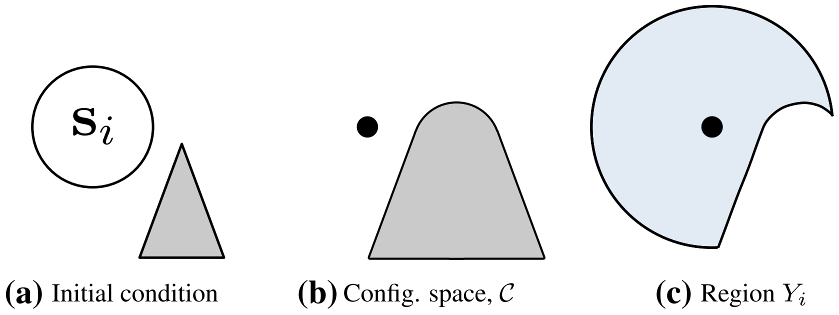

$\mathbf{s}_i$
(a) Initial condition
(b) Config. space, $\mathcal{C}$
(c) Region Y_i

Fig. 4 Example of valid initial conditions. The initial location of robot i is specified as s_i in (**a**) and the *gray triangle* represents an obstacle. The configuration space $\mathcal{C}$ is the *white* region in (**b**). The region $Y_i = \mathcal{C} \cap (s_i \oplus \mathcal{B}_{2R})$ is the *blue* region in (**c**). Every point in this region is reachable from s_i and is therefore valid (Color figure online)

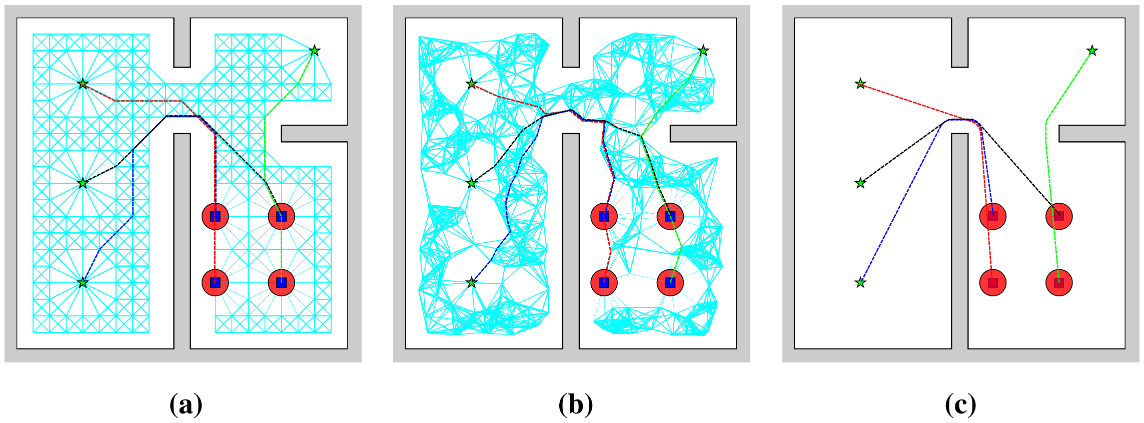


Fig. 5 **a** Shows a valid graph using an 8-connected grid with a grid-cell length of $D/2$ and the paths associated with the optimal assignment for $N = 4$ robots navigating to $M = 4$ goals. **b** Shows a PRM graph for the same environment. **c** Displays the unique minimum distance solution. Note that while it not guaranteed to be the case, all assignments are the same and the same partial orders are present using each of these graph generation methods

$$||s_i - s_j||_2 > 4R \quad \forall i \neq j \in \mathcal{I}_N$$

Similarly require all goal locations to be at least $4R$ from any other starting goal.

$$||g_i - g_j||_2 > 4R \quad \forall i \neq j \in \mathcal{I}_M$$

Denote the configuration space as $\mathcal{C}$ and consider the regions Y_i and W_i. These are the areas in free configuration space that are within $2R$ of the starting location i or goal location j as:

$$Y_i = \mathcal{C} \cap (\mathbf{s}_i \oplus \mathcal{B}_{2R})$$
$$W_j = \mathcal{C} \cap (\mathbf{g}_j \oplus \mathcal{B}_{2R}),$$

where $\oplus$ is the Minkowski sum. The initial locations are valid if the set Y_i is required to be a connected region around s_i. If this is the case, there exists some continuous path from the initial positions to anywhere that is in collision with the initial location, including its boundary. Similarly, the set $W_j \; \forall j \in \mathcal{I}_M$ must be a connected region around point g_j.

Enforcing the $4R$ condition ensures no other starting or goal states cause collision with any point in Y_i. By also ensuring $Y_i \; \forall i \in \mathcal{I}_N$ is a connected region, this means that the fully actuated robot can navigate from its initial state to any point in region Y_i. Therefore, this satisfies Eq. (3). Similarly, it can be shown that this condition satisfies Eq. (4) (Fig. 4).

As before, define the cost function for trajectory segments as distance between the end points and trajectories have constant velocity.

Figure 5 shows the graphs generated for an 8-connected grid as well as for a PRM with the same number of vertices and a similar number of edges. A prominent feature of these two graphs is that no trajectories pass through the region within $2R$ of a start or goal location unless the trajectory has an endpoint which is the start or goal location.

Figure 7 shows that for the environment which requires tight interaction between robots in Fig. 6 with 64 robots navigating on a PRM graph to 64 goals, clearance constraints are always satisfied ensuring trajectories are collision-free.

4.3 Optimal solution with second order dynamics

While experimental details are outside of the scope of this paper, (O'Hara et al. 2014) demonstrates that a team of interchangeable boats (See Fig. 8) with the capability to dock with other boats is shown to have great potential as quick response bridges or landing platforms. These boats utilize

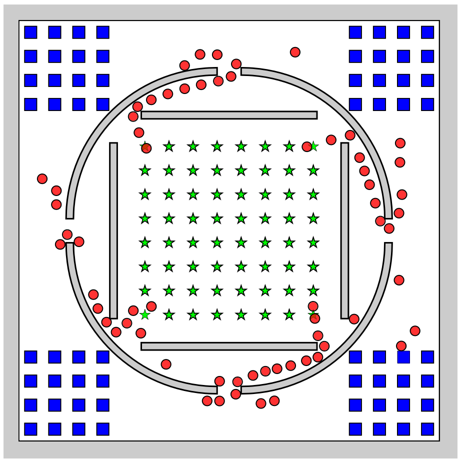


Fig. 6 A team of 64 robots mid-simulation using a PRM in a tight environment, forcing a high degree of interaction

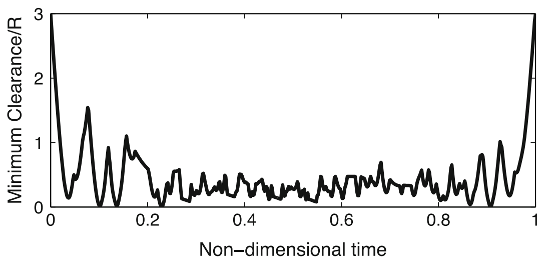


Fig. 7 A plot of the minimum clearance between any two robots over the length of all trajectories for 64 robots in Fig. 6. Clearance is defined as the space between robots. Collision avoidance is evidenced by the fact that inter-robot and environment clearance is never negative

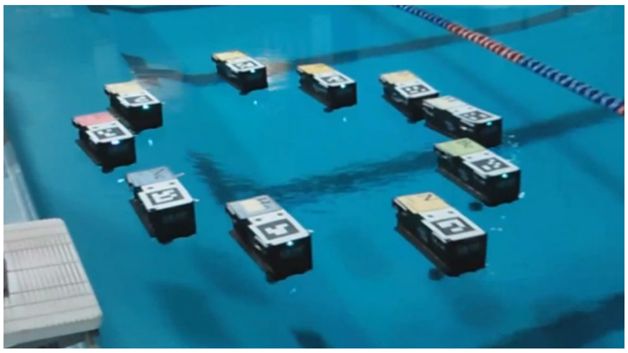


Fig. 8 Video of aquatic robots available at http://youtube.com/watch?v=2OY3nBtGqVU, image courtesy of O'Hara et al. (2014)

Gap and motivate the consideration of dynamics of the fully actuated second order systems.

It is well known that the unique minimum distance trajectory for an individual two-dimensional robot can be found using a visibility graph. The assumption that boundary conditions are no closer than $4R$ from each other satisfy the requirements in Eq. (2). However, in order to ensure Eqs. (3) and (4) are satisfied, the following conditions must be met. Not only must the regions Y_i and G_j be connected regions, but any trajectory passing within $2R$ of an initial or final condition must induce the ordering as if that trajectory went through the vertex. This is the same condition present for the work in Turpin et al. (2013a), but can be shown to satisfy Eqs. (3) and (4) after slight modifications and trajectory refinement. See Fig. 5c for an example of a unique minimum distance solution.

For the unique minimum distance paths, $P^\star(s_j, g_{\phi_j^\star})$ will be $\mathbb{C}^1$ smooth as it will be a concatenation of straight lines and arcs. As such, it is straightforward to find the unique minimum distance paths and then parameterize them to allow robots with second order constrained dynamics to minimize the time to track that path. For instance, see Fig. 9 for time plots of position, velocity, and acceleration for a the team of robots in Fig. 5c.

5 Experimental results

The algorithm presented in this paper was implemented on a team of homogeneous quadrotors to navigate a complex maze-like environment. Figure 10 shows the 75 gram test vehicle used in these experiments. To guide the experiment design, consider a scenario where a team of quadrotors enters a structure, explores a series of chambers connected by narrow passageways, and departs the structure. The structure used for experimentation, shown in Fig. 11, has been designed to be reconfigurable with small passageways. For this experiment, the passageways are configured such that there exist very long internal paths within the structure.

Due to their dynamics, quadrotors require planning in a 12-dimensional state space to fully model the robots' capabilities. Fortunately, the quadrotor has been shown to be differentially flat with flat outputs of position and yaw: $x_i \in \mathcal{X} = \mathbb{R}^3 \times SO(2)$. For the purpose of these experiments, orientation of the quadrotor is not relevant due to symmetry, so the state of the robots is the position vector $\mathcal{X} = \mathbb{R}^3$. Minimum snap trajectories have been shown to be very well suited to quadrotors (Mellinger and Kumar 2011; Michael and Murray 1998) and these trajectories can be generated through a number of waypoints quickly and reliably.

In a closed environment as in these experiments, aerodynamic effects such as ground and inter-vehicle effects are substantial for the quadrotors, particularly between multiple robots in a confined space. Therefore, $B(x)$ is represented by an ellipsoid with elongated vertical dimension to minimize downwash effects on other robots (as proposed in Michael et al. (2010)). However, as the robots are navigating in a closed space with floors and ceilings in some of the chambers, additional considerations are required. The robots' extent is only valid within the chamber in which it is currently occupying.

The graph, G, is systematically generated by selecting a large number of vertices $v_i \in \mathbb{R}^3$ in and out of the structure

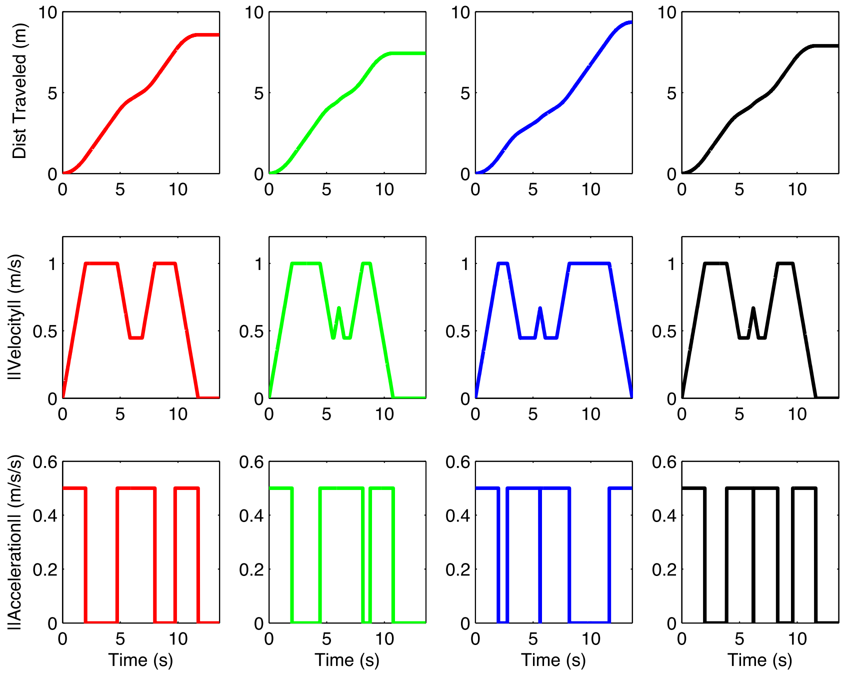


Fig. 9 State of second order robots with maximum velocity of 1 m/s and maximum acceleration of 0.5 m/s/s for the four robots in Fig. 5(c) with *colors* preserved. Note that the velocity slows around corners and that either velocity or acceleration is always at its maximum value (Color figure online)

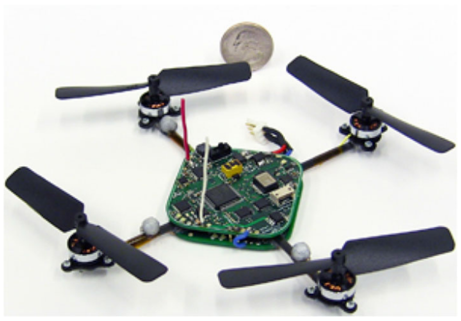


Fig. 10 KMel Robotics NanoQuad quadrotor used in experimentation

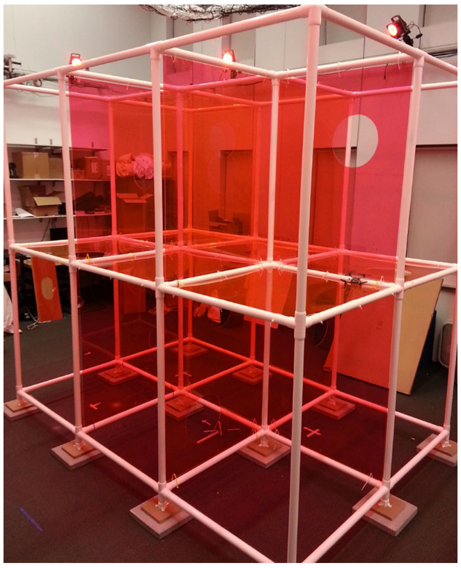


Fig. 11 The structure used for experimentation with 12 chambers

which have zero velocity, acceleration, and jerk and respect the conditions (1), (2). While it would be theoretically possible to sample states in the full 12 dimensional state space, this is impractical in this setting. The trajectory τ_{ij} is generated by minimizing snap to navigate from one vertex to the next where the cost function used is the integral of snap over the trajectory:

$$C(\tau_{ij}) = \int_{t_o}^{t_f} ||\ddddot{x}_i(t)||^2 \, dt$$

Edge e_{ij} is added if $||v_i - v_j||^2$ is below a threshold distance and trajectory τ_{ij} is obstacle collision-free.

Then, optimal individual trajectories are computed by using Dijkstra's algorithm. Optimal assignment $\phi^{\star}$ and the total ordering ψ are then computed based on the optimal paths, boundary conditions, and path costs. Finally, time offsets are computed in order of priority ψ to ensure collision avoidance of trajectories. Additional time is added to each time offset robots to minimize the effects of lingering aerodynamic turbulence and downwash.

As the quadrotor has homogeneous boundary conditions at every vertex, the original trajectories tend to take a substantial amount of time. Therefore, trajectory refinement is used to complement the complex dynamics of the quadrotor. The zero velocity, acceleration, and jerk boundary conditions are relaxed where possible to allow the quadrotor to make faster progress and expend less energy. In many instances, this results in longer distance traveled for each robot, but less energy expended and shorter mission completion times.

One experimental trial begins with 6 quadrotors hovering outside of the structure. Next, 6 goal states are specified, one

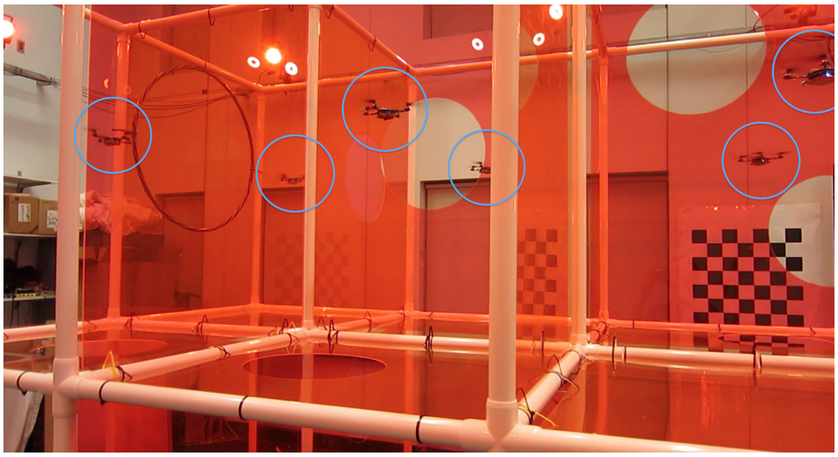


Fig. 12 Six robots flying in the structure used for experimentation

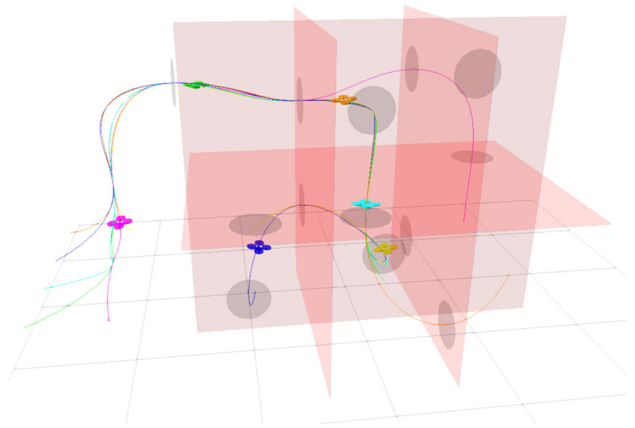


Fig. 13 Visualization of actual experiment with refined minimum snap trajectories being tracked for 6 robots in confined structure. Video available online[1]

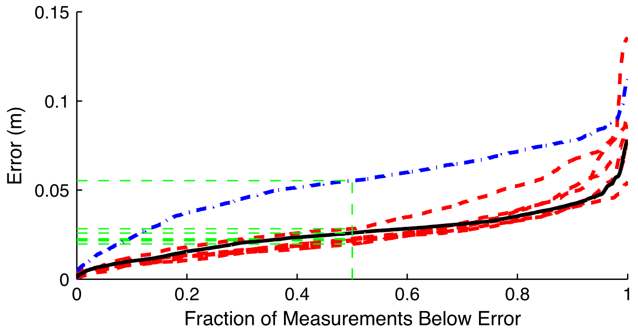


Fig. 14 Spherical error probable (SEP) for trials with 1 robot following an unrefined trajectory (*blue*, *dash dot*), 1 robot following a refined trajectory (*black solid*), and 6 robots tracking refined trajectories as a team (*red dashed*). Notice that the refined trajectories give substantially lower error. Additionally, the initial trajectory takes 6 times longer than a refined trajectory to complete as it is frequently stopping and starting (Color figure online)

goal state per chamber on the lower level of the structure. The trajectory for every robot is then computed using GAP (Algorithm 1). Figure 13 shows the trajectories computed, as well as the actual position of the robots in an experiment at one instant. Next, 6 additional goal states are given with one in each chamber of the upper level and those trajectories are computed and then tracked. Finally, 6 goal states are specified on the outside of the structure. See Fig. 12 for an image of the 6 robots arriving at specified waypoints.

Video results of single and multiple robot experiments are available online[1]. This video also includes a simulation of 25 robots navigating a cluttered 3D environment, which was computed in 0.5 seconds.

Figure 14 shows the spherical error probable (SEP) for a single robot tracking the original trajectory, the SEP for a single robot tracking a refined trajectory, and the SEP for 6 robots tracking refined trajectories. These plots clearly demonstrate the quadrotor's improved accuracy tracking these refined trajectories.

Robot states are tracked using a Vicon motion capture system. The material choice for the structure was carefully chosen to ensure localization performance anywhere inside or outside of the structure. All experiments are computed on an external computer which is running control code to maintain the quadrotors in flight, as well as computing the trajectories as a separate process. Code is mostly written in MATLAB. Construction of the graph with over 1,000 vertices generally takes less than 100 ms. Path planning for $N = 6$ robots reliably takes less than 1 ms. Assignment takes less than 10 ms. Prioritization generally takes 5 ms for 6 robots. Trajectory refinement using minimum snap trajectory generation requires between 5 and 10 ms per robot, depending on the length of trajectory. Computing time offsets requires a large number of collision checks of ellipsoids, and therefore takes between 10 and 100 ms for 6 robots depending on the length of trajectories. Thus, the total computation time is typically just under 0.2 s to generate dynamically feasible

[1] http://youtu.be/DRJPgOyN2so.

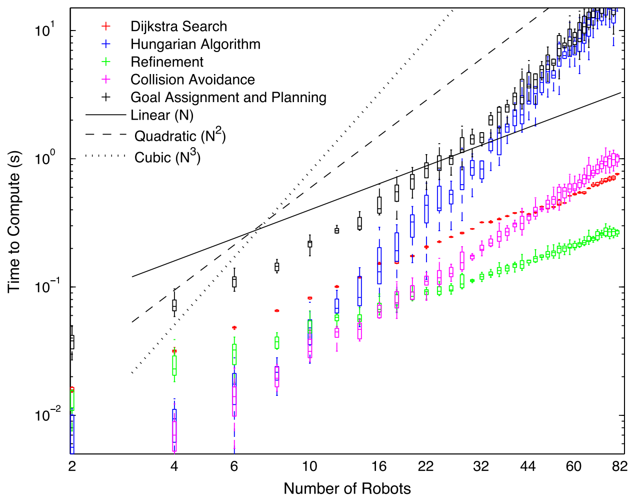


Fig. 15 100 simulation trials for each value of N to demonstrate time duration spent computing each stage of the algorithm applied to quadrotors. Note that both the planning (*red*) and refinement (*green*) grow roughly linearly in the number of robots and collision avoidance (*pink*) grows roughly quadratically. As the number of robots grows larger, the $\mathcal{O}(N^3)$ complexity of the Hungarian algorithm begins to dominate (Color figure online)

trajectories for 6 quadrotors through a tight space in MATLAB, however this could be optimized substantially to get better performance.

A much larger environment, similar to the maze-like structure in 11, was modeled in simulation to test the computational requirements as the number of robots grows much larger than the current experimental space allows. This simulated graph has $|V| > 10^4, |E| > 6\times10^5$, which for $N < 100$ are roughly constant in the number of simulated robots. The times of computation for each stage of the algorithm are displayed in Fig. 15.

This plot confirms the complexity analysis as a function of the number of robots in 3.8, where as the number of robots increases, solving the task assignment grows roughly cubically and begins to dominate the computation time at about 16 robots for the given parameters of the simulation. Of course, as $|V|$ or $|E|$ increase, the Dijkstra search will contribute more, moving the crossover of graph search and task assignment to higher N.

6 Conclusions and future work

This paper presents a resolution-complete computationally tractable trajectory generation method for a team of robots with complex dynamics. In-depth proofs of completeness and collision avoidance are included. The concept of trajectory refinement allows us to incorporate some robot dynamics while planning in a low dimensional space. A number of clarifying examples are provided and finally, the algorithm is applied to a team of quadrotor aerial vehicles.

In current work, we are pursuing distributed solutions as well as relaxing the assumption of completely interchangeable robots. It also appears this algorithm could be very useful in solving the dynamic vehicle routing problem (Psaraftis 1988), but would require additional consideration to handle time evolving goal states. Finally, the flexibility of the cost function may be useful in creating algorithms for designing controllers which could be used for exploration with teams of robots.

Acknowledgments Research supported by: ONR Grant N00014-07-1-0829, ONR MURI Grant N00014-08-1-0696, NSF CCF-1138847, ARL Grant W911NF-08-2-0004, ONR Grant N00014-09-1-1051, Matthew Turpin was supported by NSF Fellowship Grant DGE-0822.

Appendix: Extended proofs

Lemma 2 *The conditions in* (7), (8) *satisfy the antisymmetry property* $i \prec j \implies i \nsucc j$

Proof Assume $i \prec j$ and $i \succ j$. Without loss of generality, this requires one of the following cases:

$$\text{CASE 1:}\quad s_i \in P^\star(s_j, g_{\phi_j^\star}),\, s_j \in P^\star(s_i, g_{\phi_i^\star})$$
$$\text{CASE 2:}\quad g_{\phi_j^\star} \in P^\star(s_i, g_{\phi_i^\star}),\, g_{\phi_i^\star} \in P^\star(s_j, g_{\phi_j^\star})$$
$$\text{CASE 3:}\quad s_i \in P^\star(s_j, g_{\phi_j^\star}),\, g_{\phi_i^\star} \in P^\star(s_j, g_{\phi_j^\star})$$

First, consider CASE 1. This implies both of the following equalities:

$$||P^\star(s_j, g_{\phi_j^\star})|| = ||P^\star(s_j, s_i)|| + ||P^\star(s_i, g_{\phi_j^\star})||$$
$$||P^\star(s_i, g_{\phi_i^\star})|| = ||P^\star(s_i, s_j)|| + ||P^\star(s_j, g_{\phi_i^\star})||$$

Applying these to Eq. (5) yields:

$$\max\Big(||P^\star(s_i, s_j)|| + ||P^\star(s_j, g_{\phi_i^\star})||, ||P^\star(s_j, s_i)|| + ||P^\star(s_i, g_{\phi_j^\star})||\Big)$$
$$\leq \max\Big(||P^\star(s_i, g_{\phi_j^\star})||, ||P^\star(s_j, g_{\phi_i^\star})||\Big)$$

This is clearly a contradiction and therefore CASE 1 is not possible.

It can be similarly shown that CASE 2 is also impossible using a change of variables.

Next, consider CASE 3. Due to the optimal graph search, either

$$P^\star(s_i, g_{\phi_i^\star}) \subset P^\star(s_j, g_{\phi_j^\star}) \quad \text{OR} \quad ||P^\star(s_j, g_{\phi_j^\star})||$$
$$= ||P^\star(s_j, g_{\phi_i^\star})|| + ||P^\star(g_{\phi_i^\star}, s_i)|| + ||P^\star(s_i, g_{\phi_j^\star})||$$

Regardless of which of these conditions is present, it is relatively straightforward to show that Eq. (5) is violated in much the same manner as CASE 1 was handled.

As there are no conditions which allow for both $i \prec j$ and $i \succ j$, it is clear that $i \prec j \implies i \nsucc j$ □

Lemma 3 *The conditions in* (7), (8) *satisfy the transitivity property* $i \prec j$ *AND* $j \prec k \implies i \prec k$

Proof Assume $i \prec j$ AND $j \prec k$ AND $k \prec i$

Without loss of generality, there are the following conditions which result in this:

$$\begin{aligned}
&\text{CASE 1: } s_i \in P^\star(s_j, g_{\phi_j^\star}),\ s_j \in P^\star(s_k, g_{\phi_k^\star}),\ s_k \in P^\star(s_i, g_{\phi_i^\star})\\
&\text{CASE 2: } s_i \in P^\star(s_j, g_{\phi_j^\star}),\ s_j \in P^\star(s_k, g_{\phi_k^\star}),\ g_i \in P^\star(s_k, g_{\phi_k^\star})\\
&\text{CASE 3: } s_i \in P^\star(s_j, g_{\phi_j^\star}),\ g_k \in P^\star(s_j, g_{\phi_j^\star}),\ g_i \in P^\star(s_k, g_{\phi_k^\star})\\
&\text{CASE 4: } g_j \in P^\star(s_i, g_{\phi_i^\star}),\ g_k \in P^\star(s_j, g_{\phi_j^\star}),\ g_i \in P^\star(s_k, g_{\phi_k^\star})
\end{aligned}$$

First, consider CASE 1. The following conditions result:

$$\begin{aligned}
||P^\star(s_j, g_{\phi_j^\star})|| &= ||P^\star(s_j, s_i)|| + ||P^\star(s_i, g_{\phi_j^\star})||\\
||P^\star(s_k, g_{\phi_k^\star})|| &= ||P^\star(s_k, s_j)|| + ||P^\star(s_j, g_{\phi_k^\star})||\\
||P^\star(s_i, g_{\phi_i^\star})|| &= ||P^\star(s_i, s_k)|| + ||P^\star(s_k, g_{\phi_i^\star})||
\end{aligned} \tag{10}$$

it can be shown that the equalities (10) cannot be a solution to the LexBAP and:

$$\left(\sum_{l\in\{i,j,k\}} ||P^\star(s_l, g_{\phi_l^\star})||^p\right)^{\frac{1}{p}} > \left(||P^\star(s_i, g_{\phi_j^\star})||^p + ||P^\star(s_j, g_{\phi_k^\star})||^p + ||P^\star(s_k, g_{\phi_i^\star})||^p\right)^{\frac{1}{p}}$$

Therefore, CASE 1 is not a valid set of conditions.

Next, consider CASE 2:

$$\begin{aligned}
||P^\star(s_j, g_{\phi_j^\star})|| &= ||P^\star(s_j, s_i)|| + ||P^\star(s_i, g_{\phi_j^\star})||\\
||P^\star(s_k, g_{\phi_k^\star})|| &= ||P^\star(s_k, s_j)|| + ||P^\star(s_j, g_{\phi_i^\star})||\\
&\quad + ||P^\star(g_{\phi_i^\star}, g_{\phi_k^\star})||
\end{aligned} \tag{11}$$

Applying Eq. (5) for robots k and j while utilizing Eq. (11) yields:

$$\begin{aligned}
&\max(||P^\star(s_k, s_j)|| + ||P^\star(s_j, g_{\phi_i^\star})||\\
&\quad + ||P^\star(g_{\phi_i^\star}, g_{\phi_k^\star})||, ||P^\star(s_j, s_i)|| + ||P^\star(s_i, g_{\phi_j^\star})||)\\
&\le \max(||P^\star(s_k, s_j)|| + ||P^\star(s_j, s_i)||\\
&\quad + ||P^\star(s_i, g_{\phi_j^\star})||, ||P^\star(s_j, g_{\phi_i^\star})|| + ||P^\star(g_{\phi_i^\star}, g_{\phi_k^\star})||)
\end{aligned} \tag{12}$$

Make the following definitions:

$$\begin{aligned}
a &= ||P^\star(s_k, s_j)|| \quad & b &= ||P^\star(s_j, g_{\phi_i^\star})|| \quad & c &= ||P^\star(g_{\phi_i^\star}, g_{\phi_k^\star})||\\
d &= ||P^\star(s_j, s_i)|| \quad & e &= ||P^\star(s_i, g_{\phi_i^\star})|| \quad & f &= ||P^\star(s_i, g_{\phi_j^\star})||
\end{aligned} \tag{13}$$

Then, rewrite Eq. (12) using these definitions:

$$\max((a+b+c), (d+f)) \le \max((a+d+f), (b+c))$$

If $b + c \ge a + d + f$, it is clear from the previous equation that $b + c \ge a + b + c$, which implies $c \le 0$. As c is defined as $||P^\star(g_{\phi_i^\star}, g_{\phi_k^\star})||$, this value must be strictly greater than zero. Therefore, the fact that $a + d + f \ge b + c$ is used:

$$\max((a+b+c), (d+f)) \le a + d + f$$

Clearly $a + d + f \ge d + f$, so this can be simplified to:

$$a + b + c \le a + d + f$$

and further to:

$$d + f \ge b + c \tag{14}$$

Applying Eq. (5) for robots i and j and using Eq. (11):

$$\max\left(||P^\star(s_i, g_{\phi_i^\star})||, ||P^\star(s_j, s_i)|| + ||P^\star(s_i, g_{\phi_j^\star})||\right) \le \max\left(||P^\star(s_i, g_{\phi_j^\star})||, ||P^\star(s_j, g_{\phi_i^\star})||\right)$$

Using the definitions in (13), this can be rewritten as:

$$\max(e, (d+f)) \le \max(b, f)$$

It can be shown through simple contradiction that $b \ge f$ to simplify this equation to:

$$\max(e, (d+f)) \le b$$

Which means that $b \ge d + f$. Incorporating the knowledge from (14), this implies that:

$$b \ge d + f \ge b + c \implies 0 \ge c$$

However, c is defined to be strictly greater than zero and therefore it is impossible for Eq. (A) to be satisfied and therefore CASE 2 is impossible.

CASE 3 is equivalent to CASE 2 with a change of variables and is not possible.

CASE 4 is equivalent to CASE 1 with a change of variables and is not possible.

As there are no conditions which allow $i \prec j$ AND $j \prec k$ AND $k \prec i$, then it is clear that $i \prec j$ AND $j \prec k \implies i \prec k$. Therefore the conditions in Eqs. (7) and (8) satisfy the transitivity property. □

References

Burkard, R. E., & Rendl, F. (1991). Lexicographic bottleneck problems. *Operations Research Letters*, *10*(5), 303–308.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271.

Enright, J., & Wurman, P. R. (2011). Optimization and coordinated autonomy in mobile fulfillment systems. In *Automated action planning for autonomous mobile robots*.

Erdmann, M., & Lozano-Perez, T. (1987). On multiple moving objects. *Algorithmica*, *2*(1–4), 477–521.

Fredman, M. L., & Tarjan, R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, *34*(3), 596–615.

Gerkey, B. P., & Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, *23*(9), 939–954.

Kant, K., & Zucker, S. W. (1986). Toward efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research*, *5*(3), 72–89.

Kloder, S., & Hutchinson, S. (2006). Path planning for permutation-invariant multirobot formations. *IEEE Transactions on Robotics*, *22*(4), 650–665.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, *2*(1–2), 83–97.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge: Cambridge University Press.

LaValle, S. M., & Hutchinson, S. A. (1998). Optimal motion planning for multiple robots having independent goals. *IEEE Transactions on Robotics and Automation*, *14*(6), 912–925.

Liu, L., & Shell, D. A. (2011). Multi-level partitioning and distribution of the assignment problem for large-scale multi-robot task allocation. In *Proceedings of the robotics: Science and systems*, Los Angeles, CA, June 2011.

Liu, L., & Shell, D. A. (2012). A distributable and computation-flexible assignment algorithm: From local task swapping to global optimality. In *Robotics: Science and systems*.

Loizou, S. G., & Kyriakopoulos, K. J. (2008). Navigation of multiple kinematically constrained robots. *IEEE Transactions on Robotics*, *24*(1), 221–231.

Luna, R. & Bekris, K. E. (2011). Push and swap: Fast cooperative path-finding with completeness guarantees. In *Proceedings of the twenty-second international joint conference on artificial intelligence-volume volume one* (pp. 294–300). Menlo Park: AAAI Press.

Mellinger, D., & Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 2520–2525), Shanghai, China.

Michael, N., Mellinger, D., Lindsey, Q., & Kumar, V. (2010). The GRASP multiple micro UAV testbed. *IEEE Robotics and Automation Magazine*, *17*(3), 56–65.

O'Hara, I., Paulos, J., Davey, J., Eckenstein, N., Doshi, N., Tosun, T., et al. (2014). Self-assembly of a swarm of autonomous boats self-assembly of a swarm of autonomous boats into floating structures. In *Proceedings of the IEEE international conference on robotics and automation*, Hong Kong.

Peasgood, M., Clark, C. M., & McPhee, J. (2008). A complete and scalable strategy for coordinating multiple robots within roadmaps. *IEEE Transactions on Robotics*, *24*(2), 283–292.

Psaraftis, H. N. (1988). Dynamic vehicle routing problems. *Vehicle Routing: Methods and Studies*, *16*, 223–248.

Schüpbach, K., & Zenklusen, R. (2011). Approximation algorithms for conflict-free vehicle routing. In *Algorithms-ESA 2011* (pp. 640–651). Berlin: Springer.

Sharon, G., Stern, R., & Felner, A., Sturtevant, N. R. (2012). Meta-agent conflict-based search for optimal multi-agent path finding. In *SOCS*.

Sokkalingam, P. T., & Aneja, Y. P. (1998). Lexicographic bottleneck combinatorial problems. *Operations Research Letters*, *23*(1), 27–33.

Solovey, K., & Halperin, D. (2014). *k*-Color multi-robot motion planning. *The International Journal of Robotics Research*, *33*(1), 82–97.

Turpin, M., Michael, N. & Kumar, V. (2012). Trajectory planning and assignment in multirobot systems. In *Algorithmic foundations of robotics X* (pp. 175–190). Boston, MA: Springer.

Turpin, M., Michael, N., & Kumar, V. (2013a). Concurrent assignment and planning of trajectories for large teams of interchangeable robots. In *Proceedings of the IEEE International Conference on Robotics andAutomation*.

Turpin, M., Mohta, K., Michael, N., & Kumar, V. (2013b). Goal assignment and trajectory planning for large teams of aerial robots. In *Robotics science and systems*, Berlin, Germany.

Van Den Berg, J, Guy, S. J., Lin, M., & Manocha, D. (2011). Reciprocal n-body collision avoidance. In *Robotics research* (pp. 3–19). Berlin: Springer

van Den Berg, J., Snoeyink, J., Lin, M. C., & Manocha, D. (2009). Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In *Robotics: Science and systems*, Citeseer (Vol. 2, pp. 2–3).

Van Den Berg, J. P., & Overmars, M. H. (2005). Prioritized motion planning for multiple robots. In *IEEE/RSJ international conference on intelligent robots and systems, 2005. (IROS 2005)* (pp. 430–435).

Van Nieuwstadt, M. J., & Murray, R. M. (1998). Real-time trajectory generation for differentially flat systems. *International Journal of Robust and Nonlinear Control*, *8*(11), 995–1020.

Wagner, G., & Choset, H. (2011). M*: A complete multirobot path planning algorithm with performance bounds. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA (pp. 3260–3267).

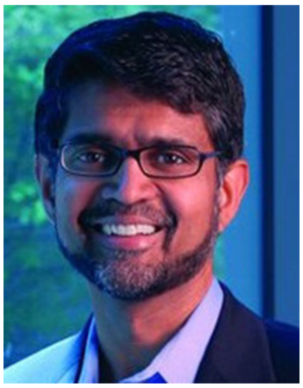
Wagner, G., Kang, M., & Choset, H. (2012). Probabilistic path planning for multiple robots with subdimensional expansion. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2886–2892).

Yu, J., & LaValle, S. M. (2012). Distance optimal formation control on graphs with a tight convergence time guarantee. In *2012 IEEE 51st annual conference on decision and control (CDC)* (pp. 4023–4028).

Matthew Turpin is a PhD candidate in the Department of Mechanical Engineering and Applied Mechanics at the University of Pennsylvania working with Vijay Kumar. He works on concurrent assignment and planning of trajectories as well as formation control for swarms of quadrotor microaerial vehicles.

Kartik Mohta is a Ph.D. candidate in the Department of Mechanical Engineering and Applied Mechanics at the University of Pennsylvania working with Vijay Kumar. He works on computer vision and control of quadrotor micro-aerial vehicles.

Nathan Michael is an Assistant Research Professor in the Robotics Institute at Carnegie Mellon University in Pittsburgh, Pennsylvania. Previously, he received a Ph.D. from the Department of Mechanical Engineering at the University of Pennsylvania (2008) before transitioning into a Research Faculty position in the same department (2010). His research focuses on enabling autonomous ground and aerial vehicles to robustly operate in uncertain environments with emphasis on robust state estimation, control, and cooperative autonomy.



Vijay Kumar is the UPS Foundation Professor and the Deputy Dean for Education in the School of Engineering and Applied Science at the University of Pennsylvania. He received his Ph.D. in Mechanical Engineering from The Ohio State University in 1987. He has been on the Faculty in the Department of Mechanical Engineering and Applied Mechanics with a secondary appointment in the Department of Computer and Information Science at the University of Pennsylvania since 1987. He is a Fellow of the American Society of Mechanical Engineers (ASME) and the Institution of Electrical and Electronic Engineers (IEEE).