Multi-Vehicle Adaptive Planning with Online Estimated Cost due to Disturbance Forces

Vishnu R. Desaraju*, Lantao Liu, and Nathan Michael

Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
{rajeswar, lantao, nmichael}@cmu.edu

Abstract. This paper proposes an adaptive planning architecture for multi-vehicle teams subject to an uncertain, spatially-varying disturbance force. Motivated by a persistent surveillance task, the planning architecture is designed with three hierarchical levels. The highest level generates interference-free routes for the entire team to monitor areas of interest that have higher uncertainty. The lower-level planners compute trajectories that can be tracked accurately along these routes by anticipating the effects of the disturbance force. To this end, the vehicles maintain an online estimate of the disturbance force, which drives adaptation at all planning levels. A set of simulation results validate the proposed method and demonstrate its utility for persistent surveillance.

Keywords: Adaptive Planning ·Vehicle Routing ·Persistent Surveillance

1 Introduction

Trajectory planning in the presence of significant, uncertain, external disturbance forces must be adaptive in order to successfully execute complex, multi-vehicle missions, such as persistent surveillance of a set of targets [1]. This is particularly true for micro aerial vehicles (MAVs) that have limited capabilities and cannot simply reject disturbances (e.g., due to HVAC systems or wind flow) via feedback control [2, 3]. A failure to compensate for these exogenous forces may lead to significant deviations from the planned trajectory, potentially leading to mission failure. Our recent work [4] addresses this problem for a single vehicle through a hierarchical planning framework that enhances trajectory tracking performance by adapting online to an uncertain, spatially-varying disturbance force.

However, the problem becomes more complex when multiple vehicles are involved. For the multi-vehicle persistent surveillance scenario, there are several objectives. We wish to generate exclusive trajectories such that each vehicle visits a set of target locations distinct from other vehicles to maximize coverage and reduce unnecessary overlap [5]. We also seek to maintain an estimate of any exogenous forces acting on the system and use this to select routes such that the trajectories generated can be tracked accurately. Additionally, since frequent re-planning is needed to leverage this estimate, the planning method must be amenable to running online.

* We gratefully acknowledge the support of ARL Grant W911NF-08-2-0004.

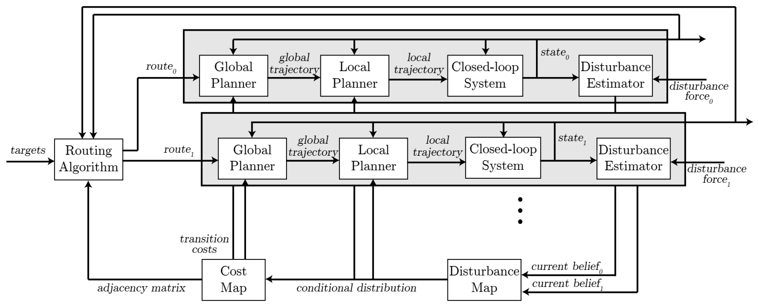


Fig. 1. Overview of the system architecture. Each gray block represents one vehicle.

Maximizing the number of important targets in the planned route can be approximated using heuristics to the Traveling Salesman Problem (TSP) [6]. However, for multi-agent systems, generating exclusive tours for different agents is extremely difficult [7]. Since the multi-vehicle routing problem belongs to the class of collective decision problems that must optimize the overall team performance, a natural way to address it is through task allocation/assignment. Strategies employing task allocation to generate routing paths have been proposed. Representative works include the incremental deployment approach [8], where paths are constructed in stages using dynamic programming, the finishing time constrained architecture [9], which embeds online path cost approximations, as well as auction-based schemes [10], where targets are auctioned/competed for by individual agents in a decentralized fashion. Different from these ideas, we recently proposed a multi-robot navigation method by adapting the graph matching variant of the Hungarian algorithm [11]—originally designed to solve the optimal assignment problem—to construct routing paths in a spatial topology [12]. The approach has several useful features including being particularly effective at generating multiple non-interfering paths and enabling online computation of routes due to its low computational complexity.

In this work, we extend our previous single vehicle adaptive planner [4] to the multi-vehicle case by introducing a high-level planning layer that is responsible for the multi-vehicle coordination. This new high-level planner is a centralized, multi-vehicle routing method built on a navigation graph and produces a set of interference-free routes for all vehicles. A hierarchical trajectory planner then generates trajectories for each vehicle to execute these routes, and the vehicles continuously update a shared estimate of the spatially-varying disturbance force encountered as they traverse the environment. This estimate drives adaptation in the trajectory planner, allowing the vehicles to avoid strong disturbance forces and ensuring feasibility of the trajectories. In addition, we update the edge costs in the navigation graph using this estimate, thereby adapting the routing assignment to the environment as well. We show that the high-level multi-vehicle routing and lower-level single-vehicle trajectory planning are tightly coupled and form a feedback loop—the high-level routing provides general navigation solutions while the lower-level planning computes the actual trajectory for the vehicle to execute and updates the navigation graph, which in turn facilitates the global routing. The general system architecture is illustrated in Fig. 1.

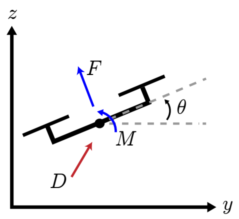


Fig. 2. The state space configuration for a 2-D quadrotor MAV in the y-z plane with control inputs F and M and external disturbance force D.

2 Preliminaries

We consider the multi-vehicle persistent surveillance task where we wish to plan trajectories for n vehicles to monitor a set of m fixed, known target locations in the environment ($n \leq m$). In addition, an unknown, spatially-varying disturbance force acts on the vehicles throughout the mission, affecting their motion.

2.1 Vehicle Model

The approach outlined in Fig. 1 is applicable to any dynamic system, but to simplify presentation, we consider a 2D quadrotor MAV whose pose in the world frame is given by position $p = [y, z]^T$ and pitch angle θ. The quadrotor has mass m, inertia J, and is subject to an additive disturbance force $D = [d_y, d_z]^T$, as shown in Fig. 2. We use a nonlinear backstepping controller [13] to track a smooth reference trajectory $p_d(t) = [y_d(t), z_d(t)]^T$, which implicitly defines a desired pitch $\theta_d(t)$ due to the coupling in the dynamics. Thrust and moment, F and M, are the control inputs, e_2 is a unit vector along the world z-axis, and $(k_p, k_v, k_R, k_\Omega)$ are the controller gains. The resulting closed loop dynamics are

$$\begin{aligned} m\ddot{y} &= -F\sin\theta + d_y \\ m\ddot{z} &= F\cos\theta - mg + d_z \\ J\ddot{\theta} &= M \\ F &= [-\sin\theta \;\; \cos\theta]\,(-k_p(p - p_d) - k_v(\dot{p} - \dot{p}_d) - mge_2 + m\ddot{p}_d) \\ M &= -k_R(\theta - \theta_d) - k_\Omega(\dot{\theta} - \dot{\theta}_d) \end{aligned} \tag{1}$$

2.2 Disturbance Cost Map and Navigation Graph

The disturbance force acting on the vehicle can be modeled as a spatially-varying, stochastic process, and the force observed at any location is a sample drawn from this process. Therefore, to drive adaptation in the planners, we must maintain an online estimate of this process. We select a discrete conditional probability distribution representation to facilitate online updates of this estimate with local observations [4]. A Kalman filter updates the belief $\tilde{D} \sim N(\mu_D, \Sigma_D)$ at each cell of the conditional distribution. The process and measurement models are

$$\begin{aligned} D_{k+1} &= D_k + \omega_k, \qquad \omega_k \sim N(0, \Sigma_\omega) \\ z_k &= \mu_{D_k} + \nu_k, \qquad \nu_k \sim N(0, \Sigma_{D_k}) \end{aligned}$$

where Σ_ω defines the rate at which the uncertainty about old estimates increases, and the measurement model is assumed to be Gaussian [14] with parameters given by $\tilde{D}$ in each cell. If a prior for this conditional distribution is available, e.g., constructed from previous missions, environment geometry, or other scenario-specific information [15], this can be used to guide the initial planning iterations.

From this disturbance force belief distribution, we can compute a discrete *cost map*, in which the cost corresponding to each cell of the discrete conditional distribution is defined as the magnitude of its current force estimate. This also allows us to build a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in which each vertex $v_i \in \mathcal{V}$ represents a target location and the weight of an edge between two locations captures information from the current cost map. Each vertex is only connected to its k nearest neighbors. Running A* search on the cost map yields estimates of the traversal cost between any two vertices connected in the graph.

The disturbance belief distribution can also be used to evaluate surveillance performance. Since uncertainty increases over time at each location, the persistent surveillance objective can be recast as minimizing uncertainty at each location vertex v_i, e.g., measured by $\mathrm{trace}(\Sigma_D(v_i))$. Therefore, for the high level routing, we wish to decrease the overall uncertainty across all vertices. This is achieved by selecting a set of representative vertices $V \in \mathcal{V}$ that are most uncertain (see Sect. 3.3) and routing the vehicles to transit them so as to reduce the uncertainty with updated observations. To do so, we construct a *navigation graph* $G = (V, E)$ to capture the topology of uncertain regions, where an edge $e(v_i, v_j) \in E$ with an edge weight $w_e(v_i, v_j) > 0$ denotes the traversibility from $v_i \in V$ to $v_j \in V$. Note that the navigation graph G is directed, so it is possible that $w_e(v_i, v_j) \neq w_e(v_j, v_i)$.

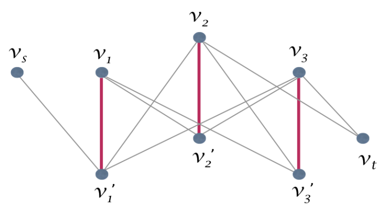
3 Multi-Vehicle Trajectory Planning with Dynamic Costs

With multiple vehicles, trajectories must be generated simultaneously for all vehicles in the team. By correlating the navigation graph to a bipartite graph, the multi-vehicle routing problem can be solved via the optimal task assignment mechanism. This provides a set of goals for each vehicle's trajectory planner.

3.1 Multi-Vehicle Route Planner Optimized by Task Assignment

We treat the current location of each vehicle as another vertex in the navigation graph and denote these vertices as the starting nodes V_s; we use V_g to denote the goal nodes. Sets V_s, V_g, V are disjoint and we generate the goal nodes online satisfying $|V_s| = |V_g| = n$ (see Sect. 3.3). Then the multi-vehicle routing problem becomes computing a set of routing paths starting from V_s, ending at V_g, and transiting some vertices in V.

Assume the underlying adjacency matrix for navigation graph G is $\mathbf{A}$. Normally, the diagonal entries $\mathbf{A}(i,i)$ are set to 0 and these entries usually convey no useful information. We have shown in our previous work [12] that by transforming G to a form of bipartite graph $\tilde{G} = (V, V', \tilde{E})$, the diagonal values can be set as non-zero and used to control various properties of the routing paths.



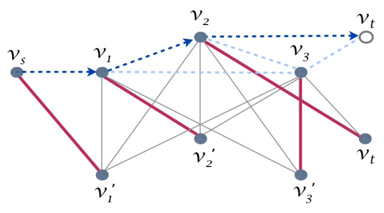


Fig. 3. Bipartite graph in the form of 3D mesh, where $V = \{v_1, v_2, v_3\}$, $V' = \{v_1', v_2', v_3'\}$, $V_s = \{v_s\}$, $V_g = \{v_g\}$. (a) Matched edges are in red bold, others are unmatched edges; (b) The number of matched edges increases by one after switching edge states of the augmenting path v_s—v_1'—v_1—v_2'—v_2—v_g. The projected routing path is v_s—v_1—v_2—v_g, the vertices of which are only in navigation graph G. The path is illustrated by dashed arrows in the top layer.

Specifically, the transformed bipartite graph $\tilde{G}$ has two sets of nodes V and V', where V' is simply a copy of V such that $|V| = |V'|$, and an edge $\tilde{e}(v, v') \in \tilde{E}$ connects the vertices $v \in V$ and $v' \in V'$. More formally, if there is an edge $e(v_i, v_j) \in E \in G$, we construct a pair of bipartite graph edges,

$$\begin{aligned} &\tilde{e}(v_i, v_j') \in \tilde{E} \in \tilde{G}, \quad v_i \in V, v_j' \in V' \\ &\tilde{e}(v_i', v_j) \in \tilde{E} \in \tilde{G}, \quad v_i' \in V', v_j \in V \end{aligned} \tag{2}$$

both of which are weighted the same as the counterpart edge $e(v_i, v_j) \in G$

$$w_{\tilde{e}}(v_i, v_j') = w_{\tilde{e}}(v_i', v_j) = w_e(v_i, v_j), \quad i \neq j. \tag{3}$$

For each pair of vertices with identical labels (i.e., v_i and v_i'), an edge is also created between them:

$$\tilde{e}(v_i, v_i') \in \tilde{E} \in \tilde{G}, \tag{4}$$

which is weighted by

$$w_{\tilde{e}}(v_i, v_i') = \mathbf{A}(i, i). \tag{5}$$

We then insert the starting and goal vertices into V and V', respectively,

$$V = V \cup V_s, \quad V' = V' \cup V_g, \tag{6}$$

and add edges following the rules described in (2). Fig. 3(a) illustrates the resulting 3-dimensional mesh with insertion of only one starting vertex and one ending vertex.

The bipartite graph is the main data structure used in the Hungarian algorithm [11]—an optimal assignment algorithm—which aims to find a *perfect matching* where each vertex in V is uniquely matched (assigned) to a vertex in V' and total cost is minimized. The Hungarian algorithm grows the matching by searching for a path, called an *augmenting path*, which consists of an alternating sequence of *matched* and *unmatched* edges but with free end nodes such that the number of unmatched edges in the path is one more than that of the matched

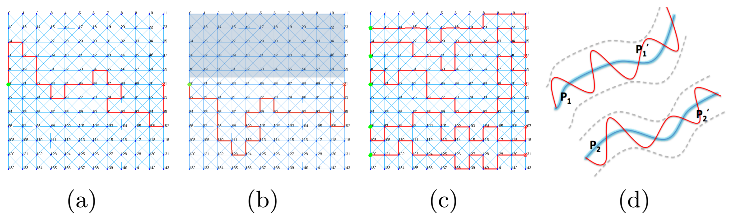


Fig. 4. Routing paths computed from adjacency matrix $\mathbf{A}$ of non-zero diagonal entries. (a) Single path: all diagonal entries weighted by non-zero costs; (b) Single path: only diagonal entries corresponding to vertices of bottom-half plane are weighted by non-zero cost; (c) Five interference-free paths are generated for five starting vertices; (d) With weighted diagonal entries, routing paths (red winding curves) attempt to cover more vertices within a bounded region. (Blue bold paths are the shortest paths.)

ones. Consequently, if the states of all edges in the augmenting path are switched, the set of matched edges, $\mathcal{M}$, are augmented whereas the number of unmatched edges are decreased. (Note, each vertex on the augmenting path is always on only one matched edge no matter how the edge states are switched.) The Hungarian algorithm iteratively grows $\mathcal{M}$ via searching for augmenting paths until $|\mathcal{M}| = |V|$, indicating that each vertex in V is matched/assigned to a unique vertex in V'. This also means that only the weights of matched edges contribute to the optimization objective f:

$$f = \min \sum_{\forall \tilde{e}(v_i,v_j)\in\mathcal{M}} w_{\tilde{e}}(v_i, v_j) = \min \sum_{i=1}^{n} \mathbf{A}(i, \phi(i)) \tag{7}$$

where $\phi(i)$ denotes the vertex that matches vertex v_i. Searching for an augmenting path requires a time complexity of $O(|V|^2)$. More specific algorithmic description and analysis can be found in [11, 16].

After an augmenting path $\tilde{P}$ starting from a vertex $v_s \in V$ and ending at a vertex $v_g' \in V'$ is computed by the Hungarian algorithm, a routing path P can be obtained by mapping $\tilde{G}$ back to G. Specifically, on $\tilde{P} \in \tilde{G}$ all $v' \in V'$ except v_g are removed and the remaining vertices are connected sequentially to form a path routing from v_s to v_g, as illustrated in Fig. 3(b). This shows that with the adjacency matrix and pre-determined v_s and v_g, a routing path connecting them can be computed by the optimal assignment method. For multiple starting and ending vertices, multiple paths can be obtained. Since each vertex can not simultaneously be on more than one augmenting path, the resulting routing paths are interference-free with no shared vertex.

Unlike the navigation graph G, when an adjacency matrix $\mathbf{A}$ is used to represent the bipartite graph $\tilde{G}$, different values of diagonal entries produce distinct augmenting paths that correspond to distinct routing paths. To analyze the overall path cost for an arbitrary set of augmenting paths, if we treat the on-path edges and off-path edges separately, Eq. (7) becomes,

$$f = \min \Big(\sum_{\tilde{e}(v_i,v_j)\in\tilde{P},\tilde{e}(v_i,v_j)\in\mathcal{M}} w_{\tilde{e}}(v_i, v_j) + \sum_{\tilde{e}(v_i,v_j)\notin\tilde{P},\tilde{e}(v_i,v_j)\in\mathcal{M}} w_{\tilde{e}}(v_i, v_j) \Big), \tag{8}$$

and if $\mathbf{A}(i,i) = 0$, $\forall i$, the second term of Eq. (8) becomes 0, and

$$f = \min \sum_{\tilde{e}(v_i,v_j)\in\tilde{P},\tilde{e}(v_i,v_j)\in\mathcal{M}} w_{\tilde{e}}(v_i,v_j). \tag{9}$$

which implies the overall cost of the paths must be the minimum, yielding a set of globally shortest routing paths.

However, if we add a value $\delta_i > 0$ to each $\mathbf{A}(i,i)$, Eq. (8) becomes

$$f = \min \big(\sum_{\tilde{e}(v_i,v_j)\in\tilde{P},\tilde{e}(v_i,v_j)\in\mathcal{M}} w_{\tilde{e}}(v_i,v_j) + k\delta_i \big), \tag{10}$$

where k is the number of vertices (and the associated matched edges) that are not on $\tilde{P}$. Intuitively, if δ_i is large, more vertices on the path can reduce k and thus improve the minimization objective. However, the path that involves more vertices may also introduce extra path cost as the paths become more winding. Thus, the vertex v_i with non-zero δ_i may be transited by a routing path only if such change decreases the value of Eq. (10). See Fig. 4 for an illustration.

3.2 Single-Vehicle Adaptive Trajectory Planner

Once routes have been established for each vehicle, we switch to planning at the individual vehicle level in order to compute trajectories. Specifically, we wish to compute trajectories along the specified routes which satisfy all constraints, such as vehicle dynamics and sensing limitations, while also anticipating and compensating for the disturbance forces present along the trajectory. Therefore, we use a hierarchical, adaptive trajectory planner based on our previous work [4] which ensures constraint satisfaction while adapting the trajectories to the disturbance force belief as it is updated online.

As shown in Fig. 5, the routing algorithm from Sect. 3.1 provides a route, defined as a sequence of target locations, to each vehicle's global trajectory planner. The global trajectory planner then selects the first target as the trajectory planning goal and computes a nominal trajectory to it from the vehicle's current position using an A* path fit with a polynominal spline to form a smooth trajectory. The global planner guides the vehicle around obstacles and other environmental features that are not in the vehicle's immediate vicinity. In particular, we take the cost map into account when computing the A* path so that the global planner avoids regions that are believed to have high cost due to strong disturbance forces. As this belief evolves, the global planner will adapt and identify better trajectories between targets that favor low-disturbance regions. However, this planner does not ensure constraint satisfaction.

To enforce this requirement, we use a local planner based on the Closed-loop RRT (CL-RRT) algorithm [17]. CL-RRT simulates a closed-loop model of the vehicle, e.g., (1), to grow a tree of potential trajectories toward randomly sampled reference points. Since the closed-loop model embeds the vehicle dynamics, as well as other constraints, the trajectories generated will satisfy all constraints

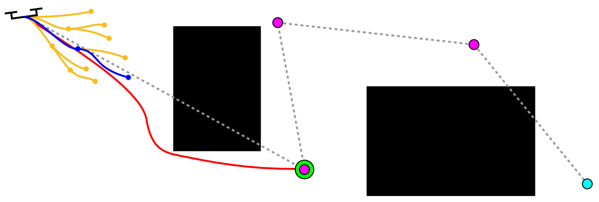


Fig. 5. Overview of the hierarchical trajectory planner. The dashed line indicates the computed route through the targets (magenta) to the routing goal (cyan). The global trajectory (red) guides the vehicle around obstacles to the first target (green) and the local planner selects the best trajectory (blue) from the set of feasible options (orange).

by construction. This property is preserved even if we add exogenous forces to the closed-loop model. Therefore, we sample the disturbance force belief distribution at each step of the simulation and propagate the dynamics accordingly. The resulting branches in the tree then represent expected trajectories of the vehicle subject to the disturbance force, and these trajectories will automatically adapt to the disturbance force estimate as the local planner replans online. We limit the local planner to explore the region around the global trajectory by biasing the sampling distribution accordingly. As each vehicle executes its local trajectory, it gains new observations of the disturbance force [14]. We can then update the shared conditional probability distribution in the areas it traverses. Predicting future motion using this continually updated distribution allows each vehicle to plan feasible trajectories that it can execute accurately and reliably. Furthermore, this approach enables online trajectory planning since the global A* planner can be run quickly on a sparse grid, and CL-RRT is designed to run online [17].

3.3 Online Adaptive Routing

Updating the disturbance belief online also enables adaptation at the routing level by informing the routing goal selection process. We select a set of vertices as the current goals and update the set as vehicles complete their routes. Specifically, for update iteration k, we select the set of vertices $V_g^{(k)}$ such that

$$V_g^{(k)} = \{v_i \mid \mathrm{trace}(\Sigma_D(v_i)) \geq \mathrm{trace}(\Sigma_D(v_j))\}, \quad \forall v_j \in V_g^{(k-1)}, \quad |V_g^{(k)}| = n$$

The set $V_g^{(k)}$ then corresponds to the locations without recent, accurate observations, as described in Sect. 2.2.

We can also update the adjacency matrix with the current cost map. As in Sect. 2.2, the edge weights in the navigation graph are updated by running A* between the corresponding vertices. We also scale the vertex weights, defined by the diagonal elements of the adjacency matrix, to down-weight the vertices with low uncertainty (i.e., that have been visited recently). The scale factor for v_i is

$$\lambda_i = \frac{\mathrm{trace}(\Sigma_D(v_i))}{\sum_{j=1}^{n} \mathrm{trace}(\Sigma_D(v_j))}$$

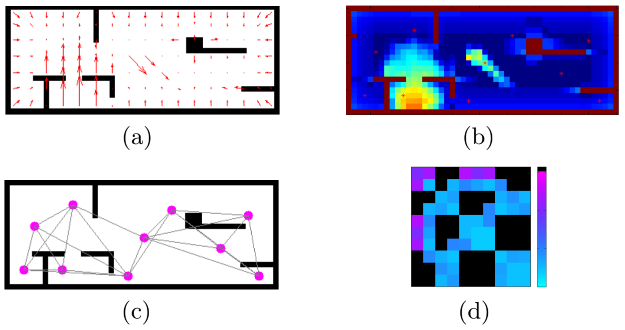


Fig. 6. Environment maps. (a) True spatially-varying disturbance force; (b) Cost map derived from the true disturbance force; (c) Navigation graph for the ten targets; (d) Adjacency matrix with true edge costs computed via A* (black indicates no edge).

and effectively prioritizes the vertices by the relative uncertainty in the belief distribution at each location.

Down-weighting based on uncertainty also enables the team of vehicles to recompute routes more frequently, further leveraging the online updated disturbance belief and cost map. During any update period, if n' vehicles complete their routes and reach their goals, $V_{g'}^{(k)}$, the routing algorithm is re-run to determine new goals and routes for those vehicles. The new set of goals is then

$$V_g^{(k)} = \left(V_g^{(k-1)} \setminus V_{g'}^{(k-1)}\right) \cup \bar{V}_g^{(k)}$$

where $\bar{V}_g^{(k)} = \{v_i \mid \mathrm{trace}(\Sigma_D(v_i)) \geq \mathrm{trace}(\Sigma_D(v_j))\},\ \forall v_j \in V_g^{(k-1)},\ |\bar{V}_g^{(k)}| = n'$.

Although the routing algorithm will compute new routes for all vehicles, this definition of $V_g^{(k)}$ maintains any unvisited goals from the previous routes, thus minimizing disruption to other vehicles. In addition, the new routes will automatically avoid revisiting locations that have been visited recently (by any vehicle) due to the low uncertainty at those locations producing a small scale factor.

4 Results

To evaluate this adaptive planning approach in simulation, we consider a persistent surveillance scenario in which a team of three quadrotor MAVs are deployed to monitor a set of ten known target locations. The vehicles must operate in the presence of an unknown, spatially-varying disturbance force, shown in Fig. 6(a) with the corresponding cost map in Fig. 6(b). However, the team is only provided a prior on the disturbance force belief distribution. Figure 6(c) shows the locations of the ten target sites (vertices) and edges that define the navigation graph. The corresponding edge costs are shown in the adjacency matrix (Fig. 6(d)). Note that these edges do not avoid obstacles or high-cost regions; that is left to the trajectory planners.

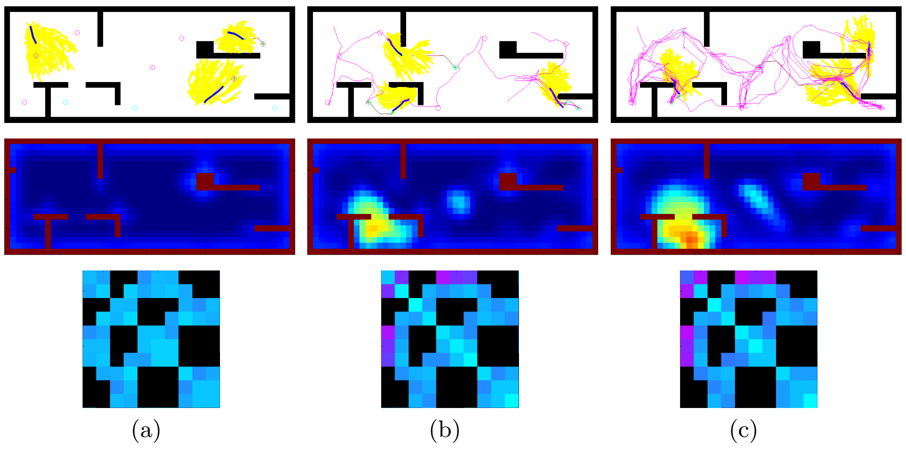


Fig. 7. Snapshots of the 3-vehicle adaptive planning scenario showing evolution of the trajectories, cost map, and adjacency matrix. (a) Initial conditions with costs from prior on disturbance force; (b) Executed trajectories help update belief and corresponding costs; (c) Vehicles find preferred routes (trajectory-dense regions) as belief converges.

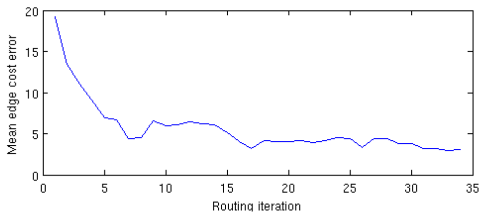


Fig. 8. Mean edge cost error (difference between current edge costs and costs computed with perfect information) decreases over successive runs of the routing algorithm.

Figure 7 shows three snapshots from the simulation. The first column represents the initial positions of the vehicles with plans computed using the cost map prior (current goals are cyan circles, CL-RRT in yellow, current trajectory in blue, executed trajectory in magenta). The vehicles execute these plans while updating the disturbance force belief, as indicated by the updated cost map and adjacency matrix in the second column. Over time, the trajectories tend to settle into certain regions as the belief distribution approaches the true distribution. Consequently, the cost map and elements of the adjacency matrix also approach the values computed in the perfect information case (Fig. 6). Figure 8 shows the mean error decrease as the vehicles continue updating their routes.

To quantify the performance of this adaptive planning approach, we look at the mean uncertainty at each target, given by trace($\Sigma_D(v_i)$), over the course of a fixed-duration mission. Figure 9 shows the mean uncertainty for three routing options: randomly selecting routing goals, using a manually prescribed tour of the vertices, and using the adaptive routing approach presented in Sect. 3.3.

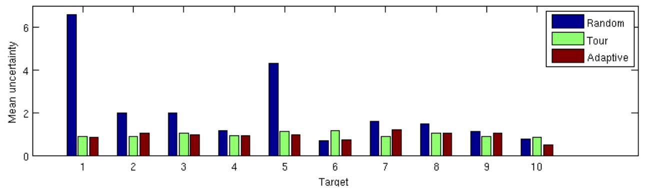


Fig. 9. Comparison of the average uncertainty per target over the course of the mission, measured by trace($\Sigma_D(v_i)$). The adaptive routing approach performs far better than randomly selecting goals for the routing algorithm and is comparable to the manually prescribed tour, but with the advantage of being completely autonomous.

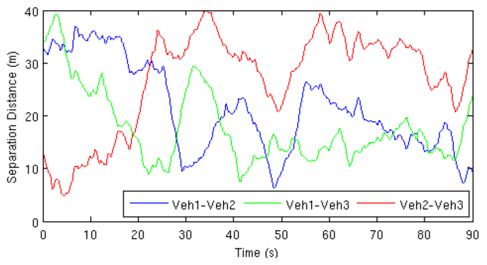


Fig. 10. Distance between vehicles following interference-free routes

Both the tour and adaptive routing perform comparably and are able to keep the uncertainty low over all vertices, while randomly selecting routing goals may leave certain targets unvisited longer. However, these scenarios assume Σ_ω is the same at all locations. If instead the uncertainty grows faster in certain regions, the difference between the tour and adaptive approach is more pronounced. If the uncertainty in the high-disturbance region at the center increases three times as quickly as in the rest of the environment, the mean uncertainty at the center vertex is 2.367 for the tour and 1.635 for adaptive routing, a 31% reduction.

The routing algorithm also provides a basic deconfliction strategy as it produces interference-free paths. This results in spatial separation of the vehicles for the duration of the mission, as shown in Fig. 10, thereby avoiding potential collisions and reducing interference effects due to the thrust from other vehicles.

5 Conclusions and Future Work

In this work, we have presented a hierarchical, adaptive planning algorithm for multi-vehicle teams that enables reliable operation in environments with uncertain, spatially varying disturbance forces acting on the vehicles. An online estimate of the disturbance force drives the planners to adapt how they select routes and trajectories in order execute a persistent surveillance mission while maintaining trajectory feasibility. Simulation results demonstrate the utility of this approach in comparison to alternate routing strategies. This provides a foundation to further investigate the interaction between disturbance belief and the routing algorithm and to consider other factors, such as prioritizing target locations.

References

1. Bethke, B., Bertuccelli, L.F., How, J.P.: Experimental demonstration of adaptive MDP-based planning with model uncertainty. In: Proc. of the AIAA Guidance, Navigation, and Control Conf., Honolulu, Hawaii (2008)
2. Garau, B., Alvarez, A., Oliver, G.: Path Planning of Autonomous Underwater Vehicles in Current Fields with Complex Spatial Variability: an A* Approach. In: Proc. of the IEEE Intl. Conf. on Robot. and Autom. (2005) 194–198
3. Ceccarelli, N., Enright, J.J., Frazzoli, E., Rasmussen, S.J., Schumacher, C.J.: Micro UAV Path Planning for Reconnaissance in Wind. In: Proc. of the Amer. Control Conf. (July 2007) 5310–5315
4. Desaraju, V.R., Michael, N.: Hierarchical adaptive planning in environments with uncertain, spatially-varying disturbance forces. In: Proc. of the IEEE Intl. Conf. on Robot. and Autom., Hong Kong, China (May 2014)
5. Jones, P.J.: Cooperative area surveillance strategies using multiple unmanned systems. In: PhD thesis, Georgia Institute of Technology. (2009)
6. Sundar, K., Rathinam, S.: Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots. CoRR **abs/1304.0494** (2013)
7. Bullo, F., Frazzoli, E., Pavone, M., Savla, K., Smith, S.: Dynamic vehicle routing for robotic systems. Proceedings of the IEEE **99**(9) (2011) 1482–1504
8. Howard, A., Matarić, M.J., Sukhatme, G.S.: An incremental self-deployment algorithm for mobile sensor networks. Auton. Robots **13**(2) (2002) 113–126
9. Bellingham, J., Tillerson, M., Richards, A., How, J.P.: Multi-task allocation and path planning for cooperating UAVs. In: Cooperative Control: Models, Applications and Algorithms at the Conference on Coordination, Control and Optimization. (November 2001) 1–19
10. Lagoudakis, M.G., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A., Koenig, S., Tovey, C., Meyerson, A., Jain, S.: Auction-based multi-robot routing. In: Proc. of Robot.: Sci. and Syst. (2005) 343–350
11. Kuhn, H.W.: The Hungarian Method for the Assignment Problem. Naval Research Logistic Quarterly **2** (1955) 83–97
12. Liu, L., Shell, D.A.: Physically routing robots in a multi-robot network: Flexibility through a three-dimensional matching graph. **32**(12) (2013) 1475–1494
13. Lee, T., Leok, M., McClamroch, N.H.: Geometric tracking control of a quadrotor UAV on SE(3). In: Proc. of the IEEE Conf. on Decision and Control, Atlanta, GA (December 2010) 5420–5425
14. Shen, S., Michael, N., Kumar, V.: Autonomous multi-floor indoor navigation with a computationally constrained MAV. In: Proc. of the IEEE Intl. Conf. on Robot. and Autom., Shanghai, China (May 2011)
15. Powers, C., Mellinger, D., Kushleyev, A., Kothmann, B., Kumar, V.: Influence of aerodynamics and proximity effects in quadrotor flight. In: Proc. of the Intl. Sym. on Exp. Robot., Quebec City, Canada (June 2012) 289–302
16. Lawler, E.: Combinatorial Optimization: Networks and Matroids. Dover Publications, Mineola, NY (2001)
17. Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., How, J.: Real-time motion planning with applications to autonomous urban driving. IEEE Trans. Control Syst. Technol. **17**(5) (2009) 1105–1118