Adaptive Path Planning in a Dynamic Environment using a Receding Horizon Probabilistic Roadmap: Experimental Demonstration

Thomas A. Frewen, Harshad Sane, Marin Kobilarov, Sanjay Bajekal, Konda R. Chevva

We demonstrate the application of a receding-horizon motion-planning method based on probabilistic roadmaps (PRM) that uses sampling from motion primitive maneuver-automata, for the problem of adaptive path planning in the case of a partially unknown obstacle field. Specifically, we consider trajectories followed by a helicopter equipped with finite-range obstacle sensors in unknown or partially unknown terrain. We provide a functional summary of our planning approach and an overview of the algorithmic architecture. The planner functionality has been demonstrated through several software-in-loop (SIL) scenarios including adaptation to newly discovered obstacles and vehicle deviation from planned paths. This paper presents experimental results from flight tests with an electric helicopter showing in-flight path adaptation to simulated obstacles.

Introduction

Consider a robotic vehicle navigating an obstacle-rich terrain with the objective to reach a prescribed goal subject to an optimality criterion e.g. in minimum time or distance traveled. The vehicle is subject to constraints arising from its underactuated dynamics, actuator bounds, and obstacles in the environment. Furthermore, the vehicle has partial information about its surroundings and is equipped with a sensor measuring the relative positions of obstacles in its field of view. This motion planning problem, although being practically very relevant, has no closed-form analytical solution (Refs. 1,2) since both the dynamics and constraints are nonlinear. The survey paper (Ref. 2) provides excellent background on the complexity of this problem and the lack of solutions with guarantees on completeness or optimality. Gradient-based and potential-field methods have had marginal success with this problem, however, they are strongly influenced by a good starting guess as existence of problem constraints yield many local minima and limit completeness. In addition, special differentiation (Ref. 3) is required to guarantee convergence due to the non-smooth nature of the constraints. The straightforward solution is to discretize the vehicle state space and perform discrete graph search. Such a technique is limited to very simple problems due to the exponential size of the search space (state dimension $\times$ trajectory epochs) also known as the *curse of dimensionality*.

This paper proposes an approach based on a recent methodology in the robotics community, known as sampling-based motion planning which includes the *rapidly-exploring random tree* (RRT) (Ref. 4) and the *probabilistic roadmap* (PRM) (Ref. 5). In our framework samples are drawn from the vehicle's continuous configuration space, and are connected through sequences of precomputed motion primitives (Ref. 6) that satisfy vehicle dynamics and constraints. These form a maneuver automaton that encodes vehicle dynamics in the form of trim states and inter-trim transition maneuvers, that may be stored for real-time sampling during planning. The strength of our approach lies in its ability to handle dynamic constraints and global planning in a unified framework, while respecting vehicle dynamics.

Construction of a fully connected dense PRM roadmap is conducted offline, while local modification of the existing roadmap is performed during real-time execution. In essence, the offline stage encodes the reachability space of the vehicle given prior knowledge about the environment. A dense PRM could be efficiently searched using classical graph search algorithms such as A* (Ref. 7) to provide an initial solution. New sensor information is subsequently reflected in online roadmap updates of graph nodes and edges and replanning is accomplished using dynamic A* (D*).

In order to adapt to tracking errors and dynamic environment, we implement a receding horizon (RHC) strategy for PRM update and search that incorporates common RHC concepts such as planning horizon, sensing horizon, and cost-to-go, where cost-to-go is derived directly from the current roadmap. In contrast to traditional RHC approaches, this provides a *consistent* and *unified* way of representing planning horizon cost (reactive plan) and cost-to-go (global plan). The maneuver automaton, using locally optimal motion primitives, is consistently used in the roadmap in conjunction with standard graph replanning and search methods. Uniform sampling and full reachability guarantee that an optimal solution can be asymptotically found.

T. Frewen, H. Sane (sanehs@utrc.utc.com), S. Bajekal and K. Chevva are with United Technologies Research Center, East Hartford, CT. M. Kobilarov is with California Institute of Technology. Submission to 2011 AHS Specialists' Meeting on Unmanned Rotorcraft, Phoenix, AZ.

Fig. 1. Maxi Joker Electric rotorcraft demonstration unit.

The paper is organized as follows. We first formulate the general adaptive path planning problem for a vehicle with dynamic constraints and an obstacle sensor. The probabilistic roadmap formulation and the computation of helicopter primitives are then described. Our recursive PRM approach for adaptation to newly discovered obstacles or deviations from a planned path is described. Next, we characterize the performance (feasibility, computational cost, objective cost) through multiple Monte Carlo simulations. Software-in-loop (SIL) studies explore the relationship between number of samples, environment clutter, and path optimality. Next, we discuss the SIL and experimental implementation of the proposed algorithms for the purposes of demonstration on a Maxi-Joker-3 electric helicopter (Fig. 1). Lastly, we present results from SIL and experimental flight tests that illustrate the capabilities of the recursive PRM algorithm. We conclude the paper by discussing opportunities for improvements and future work.

Planning Problem Formulation

Consider a vehicle with state $x \in X$ controlled using actuator inputs $u \in U$, where X is the state space and U denotes the set of controls. The vehicle's dynamics are described by the function $f : X \times U \to TX$ defined by

$$\dot{x} = f(x,u) \quad : \text{vehicle dynamics model.} \tag{1}$$

In addition, the vehicle is subject to constraints arising from actuator bounds and obstacles in the environment. These constraints are expressed through the m inequalities

$$F_i(x(t),u(t)) \geq 0 \quad : \text{constraints,} \tag{2}$$

for $i = 1,...,m$.

The goal is to compute the optimal controls u^* and final time T^* driving the system from its initial state $x(0)$ to a given goal region $X_g \subset X$, i.e.

$$\begin{aligned} (u^*,T^*) &= \arg\min_{u,T} \int_0^T C(x(t),u(t))\mathrm{dt}, \\ \text{subject to } & \dot{x}(t) = f(x(t),u(t)), \\ & F_i(x(t),u(t)) \geq 0,\ i = 1,...,m. \\ & x(T) \in X_g \end{aligned} \tag{3}$$

where $C : X \times U \to \mathbb{R}$ is a given cost function, e.g. fuel consumption, time to destination, or threat exposure. A typical cost function includes a time component and a control effort component, i.e. $C(x,u) = 1 + \lambda \|u\|^2, \lambda \geq 0$.

Obstacle Constraints The vehicle operates in a workspace that contains a number of *obstacles* denoted by $\mathscr{O}_1,...,\mathscr{O}_{n_o}$ with which the vehicle must not collide. Typically, the vehicle state can be defined by (q,v) consisting of its configuration $q \in \mathscr{C}$ and velocity $v \in \mathbb{R}^{n_v}$ (Ref. 8). Assume that the vehicle is occupying a region $\mathscr{A}(q,v)$, and let $\mathbf{prox}(\mathscr{A}_1,\mathscr{A}_2)$ be the Euclidean distance between two sets $\mathscr{A}_{1,2}$ that is negative in the case of intersection. Obstacle avoidance constraints in (2) can be written as

$$F_1(q(t)) = \min_i \mathbf{prox}(\mathscr{A}(q(t)),\mathscr{O}_i), \text{ for all } t \in [0,T]. \tag{4}$$

Obstacle representation and proximity checking is a computationally intensive procedure with a variety of documented approaches (Refs. 4, 9). For simplicity in this paper we assume a $2.5D$ digital elevation map (DEM) representation and use standard packages (Proximity Query Package (Ref. 10)) to compute the proximity function **prox**.

Sensing We assume that the vehicle is equipped with a finite-range obstacle sensor measuring the relative positions of obstacles, producing a set of points lying on the surface of obstacles. The sensor parameters include sensor range, scan-rate, and field of view. In this paper, we assume an idealized on-board simultaneous localization and mapping (SLAM) algorithm for map updates. The terrain, represented using the $2.5D$ DEM is updated at a prescribed rate based on the new occupancy data.

Planning with Probabilistic Roadmaps

We propose a motion planning method that combines PRM (Refs. 4, 5, 8) for global planning with maneuver automata for incorporation of vehicle dynamics (Ref. 6). This solution approach generates feasible trajectories that satisfy the given dynamics (1) and general constraints (2). A maneuver automaton encodes vehicle dynamics in the form of trim states and inter-trim transition maneuvers that are stored for real-time sampling during planning. The vehicle maneuvers can be pre-computed either through nonlinear trajectory optimization (Ref. 11) or by recording the closed-loop trajectories of a simulated or real vehicle. A typical PRM path consisting of a sequence of maneuvers and trims is shown in Fig. 3.a. The general framework is suitable for systems such as aerial, underwater or ground robots if one ignores the pose-dependent external forces (wind, drag, friction).

PRM Algorithm

Fig. 2 provides an algorithmic overview of our approach for a roadmap $G = (V,E)$ defined by its vertices V and edges E. A key issue in the construction of a PRM is how to connect samples s; in Fig. 2 connections are made both into and from each state sample s with selection of closest neighbors

according to proximity function. In our framework samples are connected through sequences of locally optimized motion primitives. The primitives are computed offline and organized in a library for lookup during motion planning. Combining this strategy with heuristic graph search renders the overall approach suitable for real-time applications.

Algorithm: Construct-Roadmap

Input:

s_0: initial state
s_g: goal state
n: number of nodes in the roadmap
k_{in} : number of incoming closest neighbors to each sample
k_{out} : number of outgoing closest neighbors to each sample

Output:

roadmap $G = (V, E)$

1. $V \leftarrow \{s_0\}$
2. $E \leftarrow \emptyset$
3. **while** $|V| < n$ **do**
4. $s_s \leftarrow$ Sample(S, G);
5. $V \leftarrow V \cup \{s_s\}$
6. **for all** $s \in V$ **do**
7. $N_s \leftarrow k_{\text{in}}$ closest neighbors to s according to Dist
8. **for all** $s' \in N_s$ **do**
9. **if** $edge$ = Connect$(s', s) \neq$ NIL **then**
10. $E \leftarrow E \cup \{edge\}$
11. $N_s \leftarrow k_{\text{out}}$ closest neighbors from s according to Dist
12. **for all** $s' \in N_s$ **do**
13. **if** $edge$ = Connect$(s, s') \neq$ NIL **then**
14. $E \leftarrow E \cup \{edge\}$

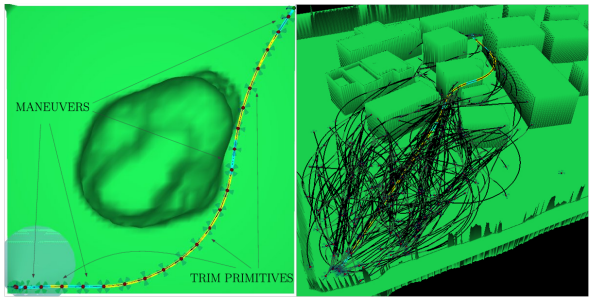
Fig. 2. Algorithm to construct a *motion planning roadmap*–a graph-based approximation of the reachable state space suitable for efficient path planning.

Sampling In recent years considerable attention has been given to augmenting basic sampling-based motion planners (e.g. see (Ref. 5)) with strategies for accelerating the search. In particular, biased sampling has been employed to exploit either domain-specific knowledge such as obstacle boundaries or to monitor performance based on connectivity (Ref. 12). Such heuristic strategies are effective when combined with regular (e.g. uniform) sampling in order to retain the required probabilistic completeness (Ref. 13) - complex problems can only be solved effectively through a proper balance of *exploration and exploitation* (Refs. 14, 15). Note that in this work we employ uniform sampling in order to guarantee complete exploration of the state space. In general, the issue of *optimal sampling* is complex and considered an open problem.

Probabilistic Completeness Sampling-based methods are well established techniques for handling (through approximation) motion planning problems in constrained environments (Ref. 4). Since requiring algorithmic completeness in such settings is computationally intractable, a relaxed notion of completeness, termed *probabilistic completeness* has been adopted (see for example (Ref. 16)) - if a solution exists the probability that the algorithm will fail to find it approaches zero as the number of roadmap samples n grows to infinity. In addition to being probabilistically complete, our approach employs branch-and-bound and pruning techniques to speed convergence towards an optimum.

Helicopter Vehicle Model and Maneuver Automata

Helicopter Model For the purposes of building PRM maneuver primitive library, we use a simplified helicopter model where the vehicle is represented by a single underactuated rigid body with position $r \in \mathbb{R}^3$ and orientation matrix $R \in \text{SO}(3)$. Its *body-fixed* angular and linear velocities are denoted by $\omega \in \mathbb{R}^3$ and $v \in \mathbb{R}^3$, respectively. The vehicle has mass m and principal moments of rotational inertia J_1, J_2, J_3 forming the inertia tensor $\mathbb{J} = \text{diag}(J_1, J_2, J_3)$.



a) b)

Fig. 3. Computed optimal sequence of 4 maneuvers and 3 trim motions in a simple environment (left). An optimal trajectory in a more complex urban environment (right) requires more sophisticated sampling and pruning of a dynamic PRM.

The helicopter is controlled through a *collective* u_c (lift produced by the main rotor) and a *side-thrust* u_ψ (force produced by the rear rotor), while the direction of the lift is controlled by tilting the main rotor forward or backward through a *pitch* γ_p and sideways through a *roll* γ_r. The four control inputs then consist of the two forces $u = (u_c, u_\psi)$ and the two shape variables $\gamma = (\gamma_p, \gamma_r)$. The state space of the vehicle is $X = SE(3) \times \mathbb{R}^6 \times \mathbb{R}^2$ with $x = ((R, p), (\omega, v), \gamma)$.

The equations of motion are

$$\begin{bmatrix} \dot{R} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} R\,\widehat{\omega} \\ R\,v \end{bmatrix}, \tag{5}$$

$$\begin{bmatrix} \mathbb{J}\,\dot{\omega} \\ m\dot{v} \end{bmatrix} = \begin{bmatrix} \mathbb{J}\,\omega \times \omega \\ mv \times \omega + R^T(0,0,-mg) \end{bmatrix} + F(\gamma)u, \tag{6}$$

where the map $\widehat{\cdot}: \mathbb{R}^3 \to \mathfrak{so}(3)$ is defined by

$$\widehat{\omega} = \begin{bmatrix} 0 & -\omega^3 & \omega^2 \\ \omega^3 & 0 & -\omega^1 \\ -\omega^2 & \omega^1 & 0 \end{bmatrix},$$

while the control matrix is defined as

$$F(\gamma) = \begin{bmatrix} d_r \sin\gamma_r & 0 \\ d_r \sin\gamma_p \cos\gamma_r & 0 \\ 0 & d_t \\ \sin\gamma_p \cos\gamma_r & 0 \\ -\sin\gamma_r & -1 \\ \cos\gamma_p \cos\gamma_r & 0 \end{bmatrix},$$

where d_r, d_t are distances from the center of mass to the main and tail rotor thrust vectors.

Helicopter Primitives The local motion planning method is based on sequencing precomputed motion primitives which satisfy the dynamics (5)–(6). The sequence consists of trim primitives corresponding to motions with constant body-fixed velocity and of maneuvers used to transition between trims.

Trim conditions. Trim motions correspond to helices, circles, or straight lines in position space. Such curves are traced using velocity components $(\omega'_z, v'_x, v'_y, v'_z)$ which are related to the body-fixed velocities v and ω through

$$\omega = R^T_{(\phi,\theta,0)}\omega', \quad v = R^T_{(\phi,\theta,0)}v', \tag{7}$$

where $R_{(\phi,\theta,\psi)}$ denotes the rotation corresponding to roll ϕ, pitch θ, and yaw ψ and where $v' = (v'_x, v'_y, v'_z)$ and $\omega' = (0, 0, \omega'_z)$.

An invariant trajectory has a constant body-fixed velocity. Therefore, in order to determine the invariance conditions for given ω' and v' one sets $\dot{v} = 0$ and $\dot{\omega} = 0$ in (6) and substitutes (7) into (6). Assuming two identical inertia components $J_2 = J_3$ and $v'_z = 0$ a closed-form solution can be found directly as

$$\begin{aligned} &\theta = 0, \quad u_y = 0, \quad \gamma_p = 0, \quad \gamma_r = 0, \\ &\phi = \arctan(-w'_z v'_x / g), \\ &u_c = m(g\cos\phi \ - w'_z v'_x \sin\phi). \end{aligned} \tag{8}$$

Assume that the helicopter is at configuration $(R_{(\phi_0,\theta_0,\psi_0)}, p_0)$ and is about the execute a trim motion with velocity $(\omega'_z, v'_x, v'_y, v'_z)$. Its state after t seconds will evolve according to

$$x(t) = (\omega, v, R_{(\phi_0,\theta_0,\psi(t))}, p(t), \gamma), \tag{9}$$

where the constant $\omega, v, \theta, \phi, \gamma$ are computed using (7) and (8). The yaw and position evolve according to

$$\psi(t) = \psi_0 + t\omega'_z, \ p(t) = p_0 + \begin{pmatrix} \cos\psi_0 \Delta p_x - \sin\psi_0 \Delta p_y \\ \sin\psi_0 \Delta p_x + \cos\psi_0 \Delta p_y \\ t v'_z \end{pmatrix},$$

where

$$\Delta p_x = \begin{cases} (v'_x \sin t\omega'_z - v'_y(1-\cos t\omega'_z))/\omega'_z, & \text{if } \omega_z \neq 0 \\ t v'_x, \text{ otherwise,} \end{cases}$$

$$\Delta p_y = \begin{cases} (v'_x(1-\cos t\omega'_z) + v'_y \sin t\omega'_z)/\omega'_z, & \text{if } \omega_z \neq 0 \\ t v'_y, \text{ otherwise.} \end{cases}$$

In this paper we consider $v'_z = 0$ which yields planar trims but as will be explained next maneuvers evolve in the complete 3-D position space.

Maneuvers. Maneuvers are computed to connect two trimmed states. Let the map $\pi : X \to X$ subtract out the invariant coordinates from a given state, i.e. $\pi((\omega, v, R_{(\phi,\theta,\psi)}, p, \gamma)) = (\omega, v, R_{(\phi,\theta,0)}, 0, \gamma)$. Then given two trims, the first one ending with state x_1 and the second one starting with state x_2, we compute a maneuver trajectory x using the following optimization procedure:

Compute:	$T;\ x : [0,T] \to X;\ u : [0,T] \to U$
minimizing:	$J(x,u,T) = \int_0^T (1 + \lambda \|u(t)\|^2)\, \mathrm{d}t, \lambda > 0$
subject to:	$\pi(x(0)) = x_1, \pi(x(T)) = x_2,$ dynamics eq. (5),(6) for all $t \in [0,T]$.

In our case, the continuous optimal control formulation was computationally solved through the discrete mechanics methodology (Refs. 11,17) which is particularly suitable for systems with nonlinear state spaces and symmetries. The computations were performed offline and optimal maneuvers solutions were assembled in a library for lookup during run-time. With trims and maneuvers organized in a library a motion plan in the absence of obstacles can be computed in closed form through inverse kinematics (Ref. 6) of a minimal number of primitives.

Recursive PRM Algorithm

The proposed algorithm can naturally handle unmapped obstacles and recover from trajectory tracking failures. This is accomplished by removing obstructed edges in the roadmap or adding new edges from the current state and performing efficient graph *replanning*. Fig. 4 sketches a prototypical planning scenario in which the vehicle must plan an optimal route from an initial state x_0 to a goal region X_f in an obstacle-rich environment with only partial prior knowledge of the environment. Fig. 4 a) shows initial roadmap construction based on the prior environment information (e.g. obstacles $\mathcal{O}_{1,2,3}$). Trajectory regeneration is utilized to plan around locally sensed, unknown obstacles obstructing the vehicle's initial path ($\mathcal{O}_4$ in Fig. 4 b) ; imperfect vehicle tracking (Fig. 4 c) of the planned path is addressed by the insertion of new roadmap nodes matching the current vehicle state(Fig. 4 d); node insertion is activated when

the error resulting from sensing and actuation noise, and environmental disturbances exceeds vehicle tracking capabilities.

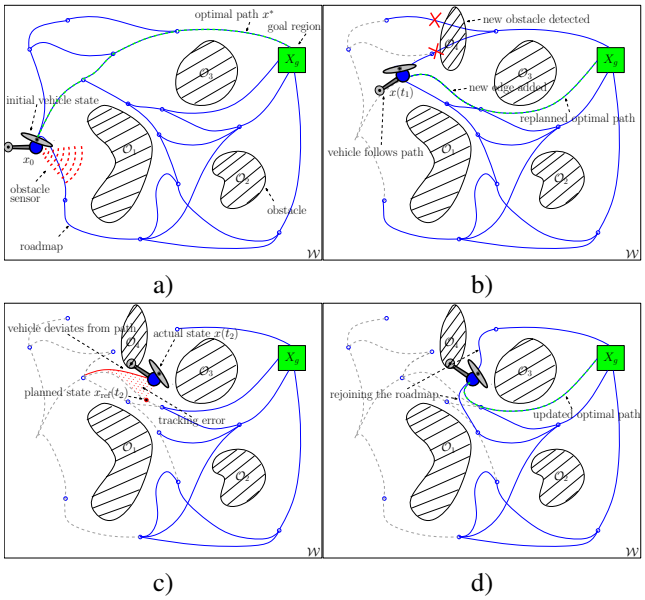


Fig. 4. A sketch of a typical planning scenario where a vehicle replans when new obstacles are discovered or vehicle deviates from prescribed planned path. Recomputing a new trajectory in all cases is performed efficiently using D^* search.

Figure 5 shows simulation of helicopter flight with finite range sensor, where graph edges are updated as new obstacles are discovered, and, if required, new roadmap samples (corresponding to trim primitives) are added and connected within the planning horizon. In either case, dynamic obstacles and obstacle map changes are handled automatically through fast replanning in the graph structure.

The PRM performance with respect to the number of roadmap nodes and the environment difficulty has been assessed in simulation. In order to limit the scope of this paper, we only present analysis of the algorithm computation time (Fig. 6). In general, initial offline construction takes several seconds while replanning during execution required less than a second and can be accomplished in real-time. Optimality of computed solutions improves with increasing number of nodes.

Further details of our PRM algorithm analysis will be the subject of future work; the results of these simulation studies were used to guide the selection of algorithm parameters used in the experimental validation described next.

Simulation and Experimental Setup

Software-in-Loop (SIL) Implementation

Functional Layout and SIL Operation The SIL layout depicted in Fig. 7 combines a flight dynamics and controls

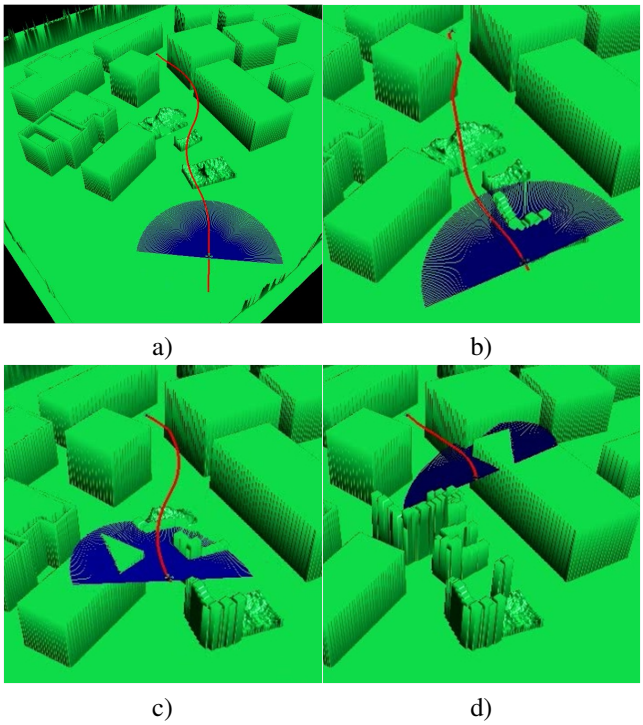


Fig. 5. Example of receding horizon map updating and dynamic replanning with a finite range sensor (blue semicircle).

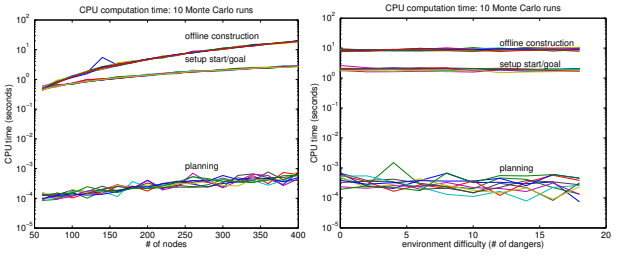


Fig. 6. Monte Carlo runs on a simulated city terrain (Fig. 3.b) showing computation time as a function of a) number of roadmap nodes ; b) environmental difficulty.

simulator in closed-loop with path-planner, with the intention to incorporate as many aspects of actual flight test as possible. Consequently, as opposed to the model described in the previous section, the helicopter simulator is a detailed closed-loop helicopter model that includes aerodynamics, actuator models, implemented flight controller and associated modes, software interfaces and communication delays. The path planner initially creates an "offline" roadmap based on the vehicle start state (communicated by the simulator) and a selected goal state. Once the offline roadmap is complete the planner switches to "online" mode and trajectory commands are sent to the simulator based on the current best planned path. The actual vehicle state, returned by the simulator, is fed back to the planner in order to adapt and update the best planned path at regular intervals (replanning frequency). The path planner has multiple threads at different time-scales that allow for separate handling of roadmap updates and trajectory commands to vehicle sim-

ulator. Synchronized access by each thread to shared resources is ensured so that deadlocks are avoided.

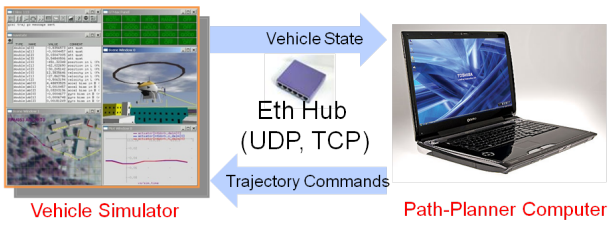


Fig. 7. Schematic of SIL functional layout depicting dedicated PRM-based path planner computer connected via an ethernet hub to a helicopter simulator (on a separate laptop). Trajectory commands are communicated via UDP packets from the path planner to the vehicle simulator; vehicle state information is communicated by the simulator to the path planner also through UDP.

Test Input Data Each SIL test run requires a start and goal state for the vehicle, a terrain map, a maneuver automaton characterizing the vehicle dynamics, and a number of PRM parameters that determine the behavior of the recursive PRM algorithm (e.g. the maximum number of roadmap vertices).

Primitives A motion primitive library for this specific vehicle is constructed based on vehicle simulation models, while experimental data is used for library validation. The baseline maneuver automaton, for dynamics symmetry group $G = SE(2) \times \mathbb{R}$, consists of 31 trim primitives (nodes) and 961 connecting maneuvers (edges). Each trim primitive corresponds to a different set of vehicle velocities (ω_z, v_x, v_y) with a maximum absolute linear velocity of 10m/s and a maximum angular velocity of 30 degrees/s that corresponds to Maxi Joker 3 electric helicopter limits (actual g-limits are higher). For comparison purposes, we also construct a pruned version of this maneuver automaton by removing the faster primitives (nodes, edges) containing 19 trims and the 361 maneuvers.

Obstacles and Map update The terrain is represented using a DEM map loaded from a file. The map corresponds to the experimental test scenario. The obstacle proximity function **prox** (see (4)) between the helicopter trajectory and the terrain is computed using the Proximity Query Package (Ref. 10) that can compute closest distance between two arbitrary meshes. The finite-range obstacle sensor parameters include sensor range, scan-rate, and FOV. The $2.5D$ DEM map is updated at a prescribed frequency.

Trajectory Controls The planner output consists of a time-parametrized trajectory consisting of a sequence of trims and maneuvers. The on-board aircraft flight control system (FCS) is designed to follow a full-state trajectory command and typically operates at 100Hz or more. The planner-to-FCS interface relays the trim-portions of the planned trajectory alone, while relying on the FCS to generate the stitching maneuvers in real-time. Although doing this introduces discrepancy between the maneuvers in the primitive library and the FCS-generated ones, this approach results in a smoother flight performance.

Test Output Data For the duration of each test run we record both the actual and commanded vehicle states and the associated timestamps $(t, \omega_x, \omega_y, \omega_z, v_x, v_y, v_z, \phi, \theta, \psi, x, y, z)$. For the actual flight tests, we record video from onboard and ground cameras.

Experimental Implementation

The PRM approach was validated through flight tests using a Maxi Joker 3 RC electric helicopter. The Maxi Joker 3 variant used in the flight tests had a flybar with a torque tube driven tail. The helicopter has gross weight capability of 20lb with a flight time of approximately 12 minutes. A schematic of the experimental set-up is shown in Fig. 8). An onboard embedded processing board hosts a trajectory tracking controller that receives trajectory commands for the current control horizon from the ground-based planner via a dedicated radio datalink. The datalink also relays current vehicle state and actuator and sensor status as feedback to the ground-based planner for the receding horizon planner update. In the event of an emergency, a human safety pilot can override and take over control of the vehicle via a dedicated datalink.

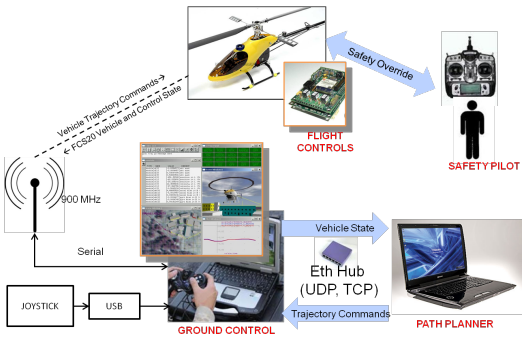


Fig. 8. Schematic of the experimental setup.

The flight tests were conducted at a baseball field approximately 500 ft long and 300 ft wide. Fig. 9 shows an aerial photo of the experimental test site with *virtual* obstacles (organized in the form of a set of city buildings) superimposed. Since the obstacles are simulated, the ranging sensor output and obstacles locations are provided to the planner as simulated inputs with appropriate delays, while the vehicle executes real flight.

Fig. 9. Aerial view of the test site with virtual obstacles superimposed

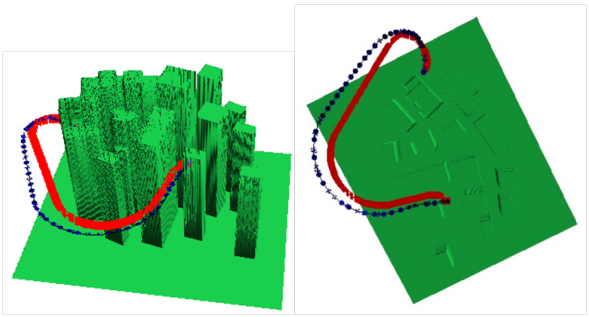
Results

SIL Tests

Our first set of SIL tests were performed using an obstacle map with city buildings (Fig. 10 a) with the path planner having complete a priori knowledge of the environment. These tests allowed for separate assessment of path planning, using the selected functional architecture with the simulator, unencumbered by the recursive planner routines required when the environment is only partially known and a finite range sensor is used. Our second set of tests, described further below, probes the latter scenario for different obstacle maps - the planner is provided a partial obstacle map (with numerous buildings from the actual environment map omitted) and only locally senses unknown terrain based on the range of obstacle sensor.

For the selected scale parameters, the simulated city building map is densely populated with buildings (Fig. 10 a) and the planner, with complete a priori knowledge of the environment, prefers a path close to the perimeter of the map where the free space volume is larger. (Fig. 10 b) shows the same SIL commanded (blue) and actual (red) vehicle trajectories from an overhead viewpoint. Note that the roadmap sampling here is not closely coupled to the layout of obstacles (other than samples in obstacles being rejected) and so the likelihood of generating a "direct" path from start to goal state through the building corridors is markedly reduced. The 200 roadmap samples used are drawn from a uniform distribution. The vehicle does deviate somewhat from the commanded planned path in Fig. 10. This is caused in part by path planning with an idealized maneuver set (maneuvers are not commanded) that is not entirely representative of the maneuvers the vehicle actually performs.

Fig. 11 shows SIL results for scenario where the planner is provided only a partial obstacle map and local sensing occurs as the vehicle moves. Only 2 of the 7 obstacles, corresponding to the obstacles nearest to the vehicle starting position are known a priori to the planner with the remainder sensed when they are within vehicle range and field of view. The figure shows the sequence of map-building and replanning snapshots as the helicopter successfully navigates the entire obstacle set.



a) b)

Fig. 10. Example of SIL path planning for simulated city building obstacle map. Two perspectives of the same simulation run are shown (panel (a) clearly indicates obstacle sizes and vehicle navigation around buildings) with commanded (red curve) and actual vehicle trajectories shown (blue line).

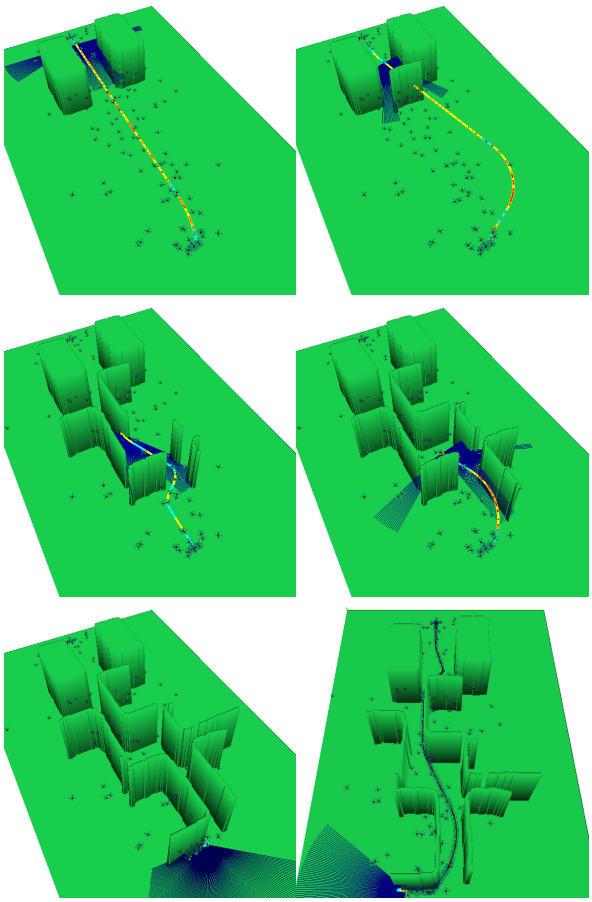
Flight-Tests

Fig. 12 shows the results of flight tests for real-time trajectory planning with a virtual city building map, corresponding to the SIL runs described previously; the obstacles are now represented "virtually" by the path planner. For the same start and goal locations as in the SIL test shown in Fig. 10, two different paths are successfully navigated by the vehicle. Both planned paths are valid but the variability in the route proposed by the planner raises some interesting questions with regard to the number of roadmap nodes required, the randomness of the PRM algorithm, and the need for appropriate metrics of path suitability other than minimum flight time that demand further investigation.

Fig. 13 shows flight test results for scenarios where the planner is provided with only a partial obstacle map and local sensing occurs as the vehicle moves. The vehicle successfully navigates from start to destination state for a scenario where a single unknown obstacle obstructs the straight line path connecting these states. As the number of obstacles unknown a priori to the planner increase so does the computational complexity associated with map updates and route replanning. As a future step, streamlining the computational expense associated with the replanning routines in our recursive planner is essential for real-time solution of more complicated scenarios.

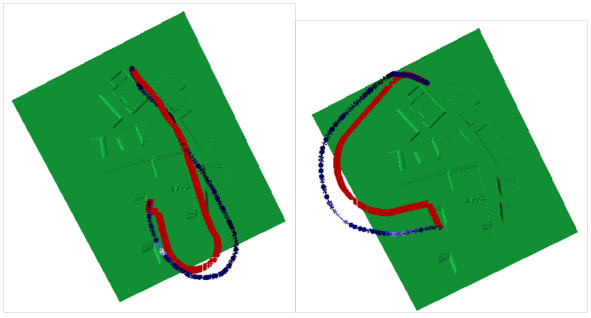
Conclusions

The strength of our robust motion planning approach using probabilistic roadmaps lies in its ability to handle dynamic constraints and perform global planning in a unified framework respecting vehicle dynamics. The method is shown to



a) b)

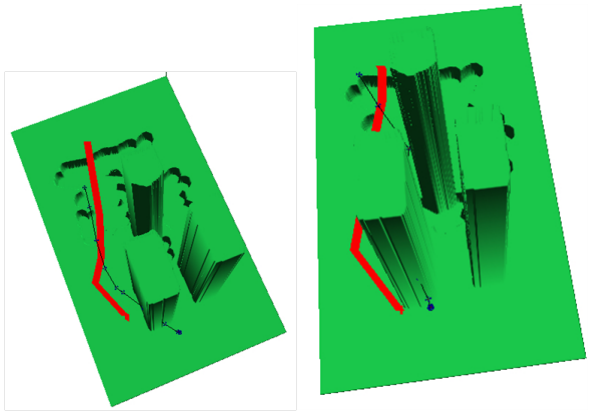
Fig. 11. Example of SIL path planning for uncertain a priori obstacle map with finite range sensor. Sequence of snapshots showing map-building and replanning are shown, with the final snapshot showing the entire trajectory. The blue ray-cone represents the ranging sensor. The small helicopter icons represent the samples placed on the planning map.



a) b)

Fig. 12. Example of real path planning in a virtual obstacles map with city building. Two instances of an experiment with the same start and goal locations (as in Fig.10) are shown (commanded (red curve) and actual vehicle trajectories shown (blue line)).

rapidly determine a feasible flight trajectory with solution optimality improving with increasing run-time. We have provided algorithmic details and schematics describing the planner logic and illustrated the applicability of this method



a) b)

Fig. 13. Example of real path planning using virtual obstacles for partially known obstacle map with finite local sensing (corresponding to simulation result in Figure 11). Two instances of an experiment with the same start and goal locations (as in Fig.10) are shown (commanded (red curve) and actual vehicle trajectories shown (blue line)).

to several simulated scenarios.

We also successfully flight-tested our path planning approach in an outdoor environment with numerous virtual obstacles. The onboard flight controller can be improved (Ref. 18) to be more compatible with velocity-primitive commands issued by the PRM planner. In order to address the computational burden of map update and obstacle checking, it will be necessary to explore efficient obstacle representations and related algorithms. In general, refinements and extensions to our approach in the area of robust planning strategies, faster computation and sampling techniques will be the subject of future development efforts and technical publications.

Acknowledgement

The authors would like to thank Chaohong Cai, Emilio Frazzoli, and Suresh Kannan for useful discussions and feedback. We also thank our test pilots, Joe Acosta and Evan Richey, for their flight test support.

References

[1]Canny, J. F., *The Complexity of Robot Motion Planning*, MIT, Cambridge, 1988.

[2]Goerzen, C., Kong, Z., and Mettler, B., "A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance," *Journal of Intelligent and Robotic Systems*, Vol. 57, 2009, pp. 65–100.

[3]Clarke, F. H., Ledyaev, Y. S., Stern, R. J., and Wolenski, P. R., *Nonsmooth Analysis and Control Theory*, Springer, 1998.

[4]LaValle, S. M., *Planning Algorithms*, Cambridge University Press, Cambridge, U.K., 2006.

[5]Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., Burgard, W., Kavraki, L. E., and Thrun, S., *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, ISBN 0-262-03327-5, June 2005.

[6]Frazzoli, E., Dahleh, M. A., and Feron, E., "Maneuver-Based Motion Planning for Nonlinear Systems with Symmetries," *IEEE Transactions on Robotics*, Vol. 21, (6), dec 2005, pp. 1077–1091.

[7]Russel, S. and Norvig, P., *Artificial Intelligence A modern Approach*, Pearson, 2010.

[8]Latombe, J.-C., *Robot Motion Planning*, Kluwer Academic Press, 1991.

[9]Ericson, C., *Morgan Kaufmann Publishers*, Springer-Verlag, New York, San Francisco, CA, 2005.

[10]Gottschalk, S., Lin, M. C., and Manocha, D., "OBBTree: A Hierarchical Structure for Rapid Interference Detection," *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, Vol. 30, 1996, pp. 171–180.

[11]Kobilarov, M., *Discrete Geometric Motion Control of Autonomous Vehicles*, PhD thesis, University of Southern California, 2008.

[12]Hsu, D., Sanchez-Ante, G., and Sun, Z., "Hybrid PRM Sampling with a Cost-Sensitive Adaptive Strategy," Proc. IEEE Int. Conf. Robotics and Automation ICRA 2005, 2005.
doi: 10.1109/ROBOT.2005.1570712

[13]Bekris, K. E. and Kavraki, L. E., "Informed and Probabilistically Complete Search for Motion Planning under Differential Constraints," First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR), July 2008.

[14]Powell, W., *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, Wiley Series in Probability and Statistics, 2007.

[15]Rickert, M., Brock, O., and Knoll, A., "Balancing exploration and exploitation in motion planning," Proc. IEEE Int. Conf. Robotics and Automation ICRA 2008, 2008.
doi: 10.1109/ROBOT.2008.4543636

[16]Ladd, A. and Kavraki, L., "Measure theoretic analysis of probabilistic path planning," *IEEE Transactions on Robotics and Automation*, Vol. 20, (2), April 2004, pp. 229–242.
doi: 10.1109/TRA.2004.824649

[17]Marsden, J. and West, M., "Discrete mechanics and variational integrators," *Acta Numerica*, Vol. 10, 2001, pp. 357–514.

[18]Cunha, R. and Silvestre, C., "A 3D Path-Following Velocity-Tracking Controller for Autonomous Vehicles," Proceedings of the 16th IFAC World Congress, Vol. 16, July 2005.