2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Incremental Micro-UAV Motion Replanning for Exploring Unknown Environments

Mihail Pivtoraiko, Daniel Mellinger and Vijay Kumar

Abstract— **This paper describes an approach to motion generation for quadrotor micro-UAV's navigating cluttered and partially known environments. We pursue a graph search method that, despite the high dimensionality of the problem, the complex dynamics of the system and the continuously changing environment model is capable of generating dynamically feasible motions in real-time. This is enabled by leveraging the differential flatness property of the system and by developing a structured search space based on state lattice motion primitives. We suggest a greedy algorithm to generate these primitives off-line automatically, given the robot's motion model. The process samples the reachability of the system and reduces it to a set of representative, canonical motions that are compatible with the state lattice structure, which guarantees that any incremental replanning algorithm is able to produce smooth dynamically feasible motion plans while reusing previous computation between replans. Simulated and physical experimental results demonstrate real-time replanning due to the inevitable and frequent world model updates during micro-UAV motion in partially known environments.**

I. INTRODUCTION

Recent advances in micro-UAV technology, including quadrotors, are enabling the application of these vehicles to a number of relevant areas, such as exploration of unmapped structures in search, rescue and surveillance applications. One of the pre-requisites of such capability is efficient generation of flight trajectories in partially known, cluttered environments. We propose a novel approach to motion generation in this setting and describe attractive properties that it offers, in particular pre-computing motion primitives that encapsulate the dynamics of micro-UAV motion and efficient replanning by reusing previous computation between replans. At the cost of certain representation losses – inevitable in any continuum sampling process – the approach features real-time autonomous micro-UAV motion replanning, a key capability requried for navigating partially known cluttered environments.

A. Motivation

The problem of motion replanning in this domain is a difficult one for at least three reasons: complex dynamics of flight systems, high dimensionality of the search space that is needed in order to effectively represent system reachability, and limited computational capacity that is available onboard such vehicles. Although there is extensive literature on motion planning for systems with complex differential constraints, none of it is directly applicable to this domain, as further described in the following section.

The authors are with the School of Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA USA {mihailp, dmel, kumar}@seas.upenn.edu

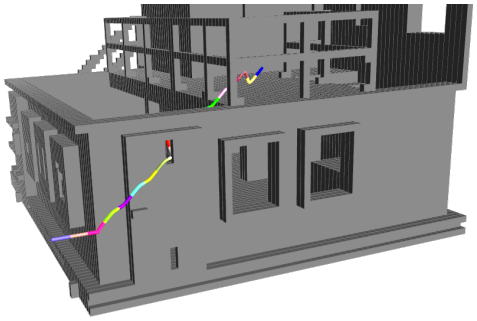


Fig. 1. **Motion plan example**. This paper studies micro-UAV maneuver generation in previously unknown, vast and cluttered spaces. In our experimental evaluation, we model urban scenes similar to the one pictured here. Search, rescue and surveillance in such scenarios are some of the applications of this work. Here, a quadrotor micro-UAV maneuver to fly into a building and localize a structure inside it was generated in real-time (216 ms runtime). This trajectory is guaranteed to be dynamically feasible and is directly executable on a quadrotor micro-UAV.

In many applications featuring physical robots, it is beneficial to perform *incremental replanning*: once a plan is computed, it is incrementally updated, by reusing previous computation, should new information about the environment invalidate it [15]. This capacity enables the planner to react quickly to the changes of the world model, including those due the to uncertainty and noise of the perception, localization and other systems – a frequent and inevitable phenomenon in robotics. Incremental replanning is a typical component of many fielded robotics systems, since plan repair can be vastly more efficient than planning from scratch [15]. To our knowledge, this work represents the first successful attempt to perform incremental replanning in micro-UAV domain.

B. Prior Work

Motion planning with kinematics and dynamics constraints has been extensively explored over the last several decades by [1, 3, 6, 7, 12, 14, 16], among many others. Application of these methods to micro-UAV planning have been limited, however, due to the challenges of this domain as suggested in Section I-A.

In addition, incremental replanning methods, motivated above, have traditionally featured search spaces with certain regular and repetitive structure that allows representing the search space as a *graph*, such as 2-D grids [8, 15]. Graph properties, such as edge in-degree ≥ 1, allow the reuse of previous computation in replanning. In contrast, *tree* based search spaces [1, 7] do not readily admit this type

978-1-4673-5643-5/13/$31.00 ©2013 IEEE

of computation reuse [13]. With this motivation, this paper explores the application of structured, graph-based search spaces to micro-UAV replanning.

In other related work, Gillula et al. [4] develop control techniques for acrobatic maneuvers of quadrotors by studying reachability of the system in a number of specific regimes. Our approach is also based on reachability evaluation, however we focus on motion generation in cluttered environments over long ranges, while that work is mostly considering free-space motion. He et al. [5] utilized a micro-UAV as an aid to ground robots during exploration in adversarial environments. While this work pushed the envelope in micro-UAV capability in partially known environments, it did not specifically focus on generating micro-UAV maneuvers.

In previous work [10], we explored the application of mixed-integer quadratic programming to generate smooth trajectories that minimize a cost functional incorporating the dynamics in the presence of obstacles. However, the complexity of the method renders it challenging in complex environments because of the need to represent each polygonal facet with a binary variable, which leads to prohibitive growth in the dimensionality of the problem. We adapt earlier work on differentially-constrained planning [13] to develop a set of pre-computed motion primitives which allows us to synthesize complex motion plans for quadrotor micro-UAVs by concatenating these primitives.

The contribution of the paper is two-fold. First, we are able to enlarge the size of the workspace considered in micro-UAV motion planning domain to length scales that are several orders of magnitude larger than the size of the vehicles, in 3D environments with obstacles. Second, we demonstrate the reuse of previous computation via incremental replanning; such methods have been actively pursued in robot motion planning in partially known environments for over a decade [8, 15], though their application to systems with second and higher-order dynamics (which is necessary for micro-UAV) have been limited.

The paper is structured as follows. Section II formalizes the problem we address here, and Section III describes a set of state sampling rules that provide the desired search space structure, followed by an account, in Section III-C, of a control sampling process that is guided by the structure. Further, certain specifics of using these controls in the context of incremental replanning are described in Section IV. Lastly, Section V describes experimental results in micro-UAV motion planning as applied to the quadrotor platform. We demonstrate that the proposed approach compares favorably with the leading state-of-the-art methods.

II. PRELIMINARIES

A. Differential Flatness

The quadrotor is a smooth nonlinear underactuated dynamics system. It requires at least a six-dimensional state space description (3D position and orientation, although typically also their time derivatives) to represent its motion, while the *input* space is four-dimensional, $U \subset \mathbb{R}^4$ (corresponding to propeller thrusts). However, this system is also known to be *differentially flat* [11]. This implies that all the state and input variables (and therefore the constraints on these variables) can be written as functions of the so-called *flat outputs* and their derivatives, in particular the position of the quadrotor center of mass and its yaw angle in an inertial frame, $\mathbf{y} = [r_x,\ r_y,\ r_z,\ \psi]^T$. Quadrotor *flat state space* is a smooth manifold $X \subset \mathrm{SE}(2) \times \mathbb{R}^{4l+1}$ that includes the space of the flat outputs and their l time derivatives. Since there is a one-to-one correspondence between X and the original vehicle state and controls, we will be referring to flat space unless otherwise noted (a comprehensive account of this relationship and the details of quadrotor dynamics are given in [9], among others). The quadrotor is assumed to be a time-invariant system satisfying a *transition equation* $\dot{x} = f(x, u)$, which induces a vector field $\mathcal{X}$ on X.

B. Problem Definition

Assume $X_{\mathrm{free}} \subset X$ that is free of obstacles: either other objects in the environment, pre-defined "no-go" regions, dynamics constraints, etc. We seek to steer the system from an initial state $x_I \in X_{\mathrm{free}}$ to a final state $x_F \in X_{\mathrm{free}}$. Let $u : \mathbb{R}_+ \to U$ be a *control function* of time defined over the time interval $[0, T]$ (or simply *control*). Control Hilbert space $\mathcal{U}$ spans all feasible controls of the system. Suppose a control $u(t) \in \mathcal{U}$ is executed in free space without disturbances when the system is in $x \in X$. The resulting state space projection $\xi_{u,x} : \mathbb{R}_+ \to X$ is a *flow* of $\mathcal{X}$, or *trajectory* – a solution to the system's transition equation from x given $u(t)$ (time dependence was dropped for clarity).

In addition, a function $c : X \times \mathcal{U} \to \mathbb{R}_+$ assigns to each control a value of *cost* that is related to a domain-specific notion of motion quality. Without loss of generality, this paper focuses on cost defined in terms of three practically important, though competing, objectives in micro-UAV domain: trajectory traversal time T, a measure of trajectory smoothness c_e (related to control effort and energy expenditure):

$$c_e(u, x) = \sum_{k=1}^{l} w_k \int_0^T \left| \frac{d^k \mathbf{y}}{dt^k} \right|^2 dt$$

where w_k terms optionally weigh the derivative terms, in addition to the workspace based term c_w that captures preference of traversing certain parts of the workspace over others (e.g. adversarial or poor visibility regions may be assigned higher cost):

$$c(u, x) = \mathbf{w} \cdot [T, c_e(u, x), c_w(u, x)]^T \tag{1}$$

where weight vector $\mathbf{w}$ additionally specifies relative preference between the cost components. Other measures of cost may be utilized, however.

The motion planning problem we address is the following constrained optimization program:

$$u^* = \arg\min_{u \in \mathcal{U}} c(u, \xi_{u,x}) \tag{2}$$

$$\begin{aligned} \text{s.t.} \quad \xi_{u,x}(0) &= x_I \\ \xi_{u,x}(T) &= x_F, T > 0 \\ \xi_{u,x}(t) &\in X_{\text{free}}, \forall t \in [0, T] \end{aligned}$$

for given initial and final states $x_I, x_F \in X_{\text{free}}$.

C. System Symmetries and Invariance

An array of systems of practical interest exhibit symmetries in system dynamics that can be exploited to reduce the computational complexity of (2) [2, 3, 12, 13]. These symmetries provide for existence of certain controls that are *invariant* with respect to certain transformations in some regions of X and therefore may be pre-computed, cached and reused in those regions, leading to potentially significant reduction in computation.

Specifically, we assume that X can be partitioned, at least locally, into a Cartesian product of two sub-manifolds $X = G \times Z$, where G is a Lie group with identity element e, and $\mathbf{g}$ is a Lie algebra on G. Any $x \in X$ is then a tuple $(g, z) \in G \times Z$. We denote Z as the *base space*, and G as the *fiber*, such that X is the *principal fiber bundle*. Furthermore, the system transition equation is said to be *invariant* with repect to the left action of $\gamma \in G$ onto $X \times \mathcal{U}$ if its solutions $\xi_{u,x}$ satisfy:

$$\gamma \xi_{u,(g,z)}(t) = \xi_{u,(\gamma g,z)}(t), \quad t \in [0, T], \quad T > 0 \qquad (3)$$

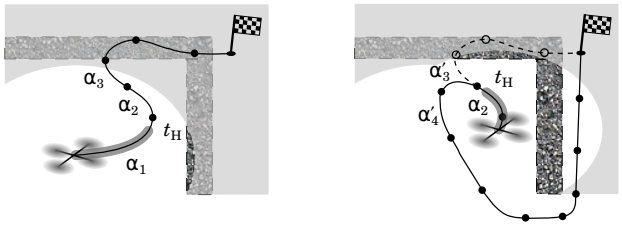
Note that, unlike [2], we cannot assume invariance of the cost functional (1) due to the workspace term c_w because the workspace is related to the fiber G and the left action of γ modifies its value, namely the equality (3) is not satisfied.

D. Motion Primitives

Invariance property allows us to develop a representation of the planning problem based on pre-computing controls at *fiber origin* $g_0 \in G$ and reusing them elsewhere else in G. We refer to such pre-computed controls as *motion primitives*. Their collection $\mathcal{A}$ forms a *motion description language*, whose *symbols* are motion primitives $\alpha = (u, z, c_{u,(g_0,z)}) \in \mathcal{U} \times Z \times \mathbb{R}_+$ consisting of pre-computed controls, the resulting trajectories and their free-space costs (omitting the c_w term), defined at a particular $z \in Z$ and fiber origin g_0 but valid anywhere else in G. A motion plan $\omega = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$ is a sequence of primitives. In general, not all primitives can be combined arbitrarily. Therefore, the feasible motion plans belong to a subset Ω^* of the free monoid over all possible primitives.

E. Incremental Replanning

Since we emphasize the micro-UAV motion in previously unknown environments, we assume that X_{free} changes quickly, e.g. perhaps many times over the interval $[0, T]$, with a frequency that depends on sensor measurement rate. This requires the vehicle to generate and transition to a new plan ω_{i+1} while executing ω_i. To maintain control continuity, we require the transition between the plans to occur at least t_H seconds (referred to as the *replanning horizon*) ahead along the current plan. Since we impose control quantization (Section II-D), this implies that the transition will take place at the end of the current or a future primitive that is part of ω_i being executed.



(a) Before replan (b) After replan

Fig. 2. **Replanning horizon**. The figure illustrates the process of replanning during to quadrotor motion in an unknown environment. The robot attempts to move to a goal state in upper right of the figure (marked with a flag icon). Initially (left figure), the robot's sensors have not yet observed the upper wall (the unknown parts of the workspace are in grey). An optimistic planning strategy allows computing a trajectory through unknown area, under the assumption that once the robot makes progress executing it, the planner will be able to modify the plan to avoid collision. To that end, the quadrotor commits the first t_H seconds of a plan for execution (thick grey highlight over the trajectory), and replans from the end of the affected motion primitive (α_2 in Figure b) onwards to the goal. Dashed line in Figure (b) shows the previous plan before replanning.

Figure 2 illustrates the replanning process, adapted to micro-UAV domain from [13, 15]. A quadrotor moves through a partially known area containing a Γ-shaped obstacle toward a goal on upper right of the figure. Unknown parts of the environment are in grey, and are "discovered" by the robot as it travels and its sensor perception ellipsoid moves over the unknown areas. In 2(a), robot's motion plan is $\omega_i = \{\alpha_1, \alpha_2, \alpha_3, \ldots\}$. Since the replan horizon extends through the primitive α_1 (thick grey highlight along the trajectory in the figure), the robot commits to executing α_1. Notice that the planner guarantees that this primitive is collision free [1] An *optimistic* planning strategy is employed: all trajectory segments are allowed to pass through unknown areas (though they may be assigned a higher cost c_w).

As the robot makes progress executing α_1, its replanning horizon enters α_2, thereby committing it for execution also, as shown in Figure 2(b). However, since the robot's perception horizon moves with the robot, it "discovers" the wall at the top of the picture. The planner performs a replan by setting x_I in (2) to the final endpoint of the last committed primitive, α_2, (overall motion goal remains x_F) and selects a different set of primitives after α_2 so that the obstacle is avoided, resulting in a new plan $\omega_2 = \{\alpha_2\} \cup \{\alpha'_3, \alpha'_4, \ldots\}$. We seek to perform replanning as frequently as computationally possible in order to keep the moving vehicle safe from collision while traversing a partially known environment.

[1] See Section IV-B for an extended replanning procedure which guarantees collision avoidance even under significant drift.

III. SEARCH SPACE DESIGN

A. Structured Search Space

We guarantee the graph structure of the search space by developing a sampled state space $\hat{X} = (\hat{G}, \hat{Z}) = \{g_0, g_1, \ldots\} \times \{z_0, z_1, \ldots\} \subset X$, where the samples are arranged as a multi-dimensional lattice. The differential flatness property of quadrotor micro-UAV's (Section II-A) allows pre-computation of free-space controls that steer the system between any two such state samples [9]. Connecting pairs $x_i, x_j \in \hat{X}$ in this manner induces a *state lattice* search space $\Omega^* \subset \mathcal{U}$, which inherits regular structure from $\hat{X}$ [13]. Since the original planning problem, as posed in Section II-B, is NP-hard, it is common practice to use sampled approximations, despite the inevitable losses in optimality and completeness [1, 3, 7, 12, 14, 16].

Given $z_i, z_j \in \hat{Z}$, let a set of primitives $\mathbf{a}_{z_i,z_j} = \{\alpha_{z_i,z_j,1}, \alpha_{z_i,z_j,2}, \ldots\}$ steer the system from (g_0, z_i) to $(g_1, z_j), (g_2, z_j), \ldots$, respectively, namely to a set of states that share the base value z_j. By symmetry invariance, $\mathbf{a}_{z_i,z_j}$ maintains regular lattice structure if applied at any other $(\gamma g_0, z_i)$. Therefore, $\mathbf{a}_{z_i,z_j}$ generates a subset of Ω^* that consists of motions from (g_i, z_i) to (g_j, z_j), $\forall g_i, g_j \in \hat{G}$. By induction, we define a *generator* $\hat{\mathcal{A}} = \{\mathbf{a}_{z_i,z_j} | \forall z_i, z_j \in \hat{Z}\}$ that generates the entire lattice Ω^*. For example, in the case of a 2D grid, the set of edges that connect some origin vertex to its four immediate neighbors generate the entire four-connected grid by translation and replication. Notice that this definition admits control sets $\mathbf{a}_{z_i,z_i}$ that keep the base value constant; such motions are known as *trim trajectories* in the aerospace community [2, 3].

Because the cost functional (1) does not possess fiber invariance property (Section II-C), we are not able to utilize the motion primitive automaton framework [2]. Instead, the syntax of the motion language is formalized as a *roadmap* graph $\mathcal{G} = (E, \hat{X})$, where edges $E = \bigcup_{\gamma \in \hat{G}} \gamma \hat{\mathcal{A}}$ is the set of generator copies replicated throughout $\hat{G}$ and vertices are sampled state values in $\hat{X}$.

B. State Space Sampling Rules

While the specifics of state sampling that result in $\hat{X}$ depend on the details of the system transition equation, the robot's perception components and its environment, several general sampling strategies are worth noting.

To maintain symmetry and regularity motivated in Section III-A, we sample the quadrotor fiber space $\hat{G} \subset \mathrm{SE}(2) \times \mathbb{R}$ as a regular lattice (with a rectangular, hexagonal or other repeating unit). The sampling resolution C is chosen to be commensurate with the trajectory following accuracy of the robot controller and the *feature size* of the robot's perception system (e.g. for occupancy grid approaches, the resolution of this grid). Intuitively, performing motion planning at position grid resolution that is vastly different from that quantity is likely to lead to excessive computation (if planning resolution is much higher than that of the world model) or excessive losses in motion quality (if planning resolution is too low).

The quadrotor base space $Z \subset \mathbb{R}^{4l}$ captures the dynamics of the system and therefore has less intuitive basis for sampling. A reasonable approach is to choose a coarse sampling of each dimension and enhance it as demanded by the application. Thus, for a quadrotor an initial sampling choice may consist of extremal values of flat space rate of change along with a value of zero (steady-state): $\{\dot{\mathbf{y}}_{\min}, \mathbf{0}, \dot{\mathbf{y}}_{\max}\}$ and so on for further derivatives [2]. If needed, this interval is further subsampled.

C. Generating Motion Primitives

The discussion so far described the search space design principles and motivated the development of a set of motion primitives $\hat{\mathcal{A}}$ that serves as the state lattice generator. Algorithm 1 suggests a greedy approach for developing this generator set automatically.

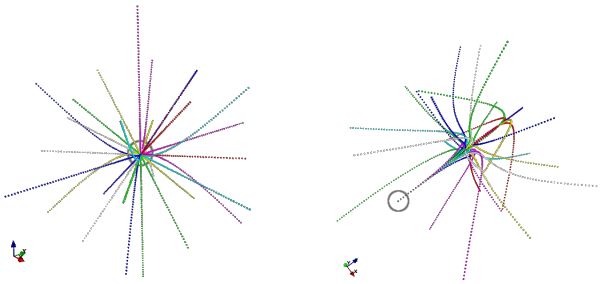
The inputs to the algorithm are: N the maximal number of motion primitive variants that are allowed, ρ the distance metric over $\hat{X}$, and R the radius of a *representation region*, a ball in $\hat{X}$ that is considered for generating motion primitives. The greater N and R are, the better representation quality of the resulting motion primitive set, although on-line planning computation also increases; the values are chosen as large as computationally tractable for real-time operation. Line 2 attempts to produce a set of motion primitives emanating from all discrete base values $z_i \in \hat{Z}$, and line 4 iterates over all base values once again to make sure that the resulting generator subset $\mathbf{a}_{z_i}$ has primitives that can connect to other generator subsets $\mathbf{a}_{z_j}$.

Once the terminal base values are fixed, line 7 sets up a loop over the fiber values (within the representation region)

[2]Though the related increases in trajectory following error and control effort may be of concern in some applications.

Input: N, ρ, R
Output: $\hat{\mathcal{A}}$
1 $\hat{\mathcal{A}} = \emptyset$;
2 **foreach** $z_i \in \hat{Z}$ **do**
3 $\mathbf{a}_{z_i} = \emptyset$;
4 **foreach** $z_j \in \hat{Z}$ **do**
5 initialize PQ;
6 $x_I = (g_0, z_i)$;
7 **foreach** $g_j \in \hat{G}$ **do**
8 $x_F = (g_j, z_j)$;
9 **if** $\|x_I - x_F\|_\rho < R$ **then**
10 α_{z_i,z_j} = generate_primitive(x_I, x_F);
11 PQ.push(α_{z_i,z_j});
12 **end**
13 **end**
14 $\mathbf{a}_{z_i}$ = PQ.pop(N);
15 **end**
16 $\hat{\mathcal{A}} \leftarrow \mathbf{a}_{z_i}$;
17 **end**

Algorithm 1: A greedy algorithm for automatic generation of motion primitives.



(a) Hover state (b) High velocity in Z-direction

Fig. 3. **Pre-computed motion primitives**. The proposed method seeks to encapsulate much of the complexity of dynamics and control of micro-UAV's with motion primitives, off-line pre-computed controls that represent feasible motions of the system. By representing system reachability with a carefully designed set of control samples, the need to reason about system dynamics at planning time is eliminated, allowing ultra-low power CPU's (perhaps even lacking floating-point units) to compute elaborate motion plans in real-time. The figure shows a set of primitives of a quadrotor (gray circle) while it is hovering in place (a) and moving at a certain velocity along the Z-axis (b).

that generates motion primitives (line 10) via a boundary value problem (BVP) solver [9] and stores them in a priority queue. Line 14 pops N primitives from the priority queue (in increasing order of free-space cost) and adds them to the partial generator set of primitives emanating from z_i, which then is added to $\hat{\mathcal{A}}$ on line 16.

IV. INCREMENTAL REPLANNING WITH CONTROL SETS

This section describes the specifics of micro-UAV replanning in partially known environments when using motion primitives developed above.

A. Trajectory Swaths

It is widely accepted that one of the most computationally intensive procedures in planning is collision detection and estimation of motion cost. This computation involves convolving the vehicle frame along the workspace trajectory in question. Since we pre-compute motion alternatives, their workspace paths can be also cached in the form of invariant trajectory *swaths*, a set of voxels that are occupied by the robot body during motion (red cells in Figure 4). Hence, motion cost computation is reduced to iterating over an array of voxels resulting in potential orders of magnitude speed-up.

B. Inevitable Collision State Rejection

When the micro-UAV is moving through a partially known environment, the motion planner must guarantee that the robot is able to stop in the event that the unknown area is determined to be an obstacle. In general, verifying that a state along the computed micro-UAV trajectory is not an *inevitable collision state* (ICS) [17] is difficult, however we may once again leverage the motion sampling structure established in Section III in order to simplify the problem.

As Figure 5 illustrates, ICS rejection may be performed by adding a layer of look-ahead to each vertex expansion during planning. The motion primitive generation method, proposed here, naturally includes *stop motions*, primitives that bring the system to rest (micro-UAV hover state). The stop motions of each successor state are checked for entering the unknown region. If there is no available stop motion entirely inside the known region, we consider the state an ICS.

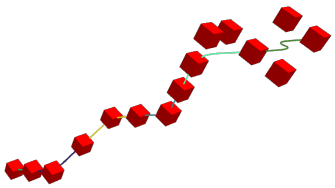


Fig. 4. **Trajectory swath**. One of the quadrotor motion primitives is shown along with the pre-computed *swath* of the motion, a set of workspace voxels that the vehicle traverses when executing a motion. During on-line planning, the evaluation of the cost of this maneuver with respect to a cost map is reduced to a summation operation over a memory array – system simulation is avoided, resulting in significant runtime speed-up.

C. Incremental Search

Compatibility with incremental search was one of the motivations for the present search space design, as described in Sections I-A and III-A. Due to the proposed search space structure, any standard incremental search algorithm ([8, 15] among others) may be used in order to implement reactive replanning while navigating partially known cluttered environments. Pre-computed lookup table heuristics and invalidation regions [13] may further reduce replanning computation.

V. EXPERIMENTAL RESULTS

A model of the KMel "*nano*" quadrotor was used throughout the experiments. The search space feature size C (Section III-B) was set to 20cm, roughly the size of the vehicle. The world model was a 3D occupancy grid with resolution C, and the position dimensions of $\hat{G}$ were similarly sampled as a regular 3D grid with resolution C. The robot was assumed equipped with perception sensors that operate within a *perception radius* R_P around the vehicle; in the experiments, we expressed this value in terms of the feature size, R_P/C.

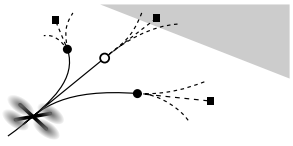


Fig. 5. **Inevitable collision state rejection**. The motion primitive generation method, proposed here, naturally includes *stop motions*, controls that bring the system to rest (micro-UAV hover state). This allows fast checking whether a state is potentially an inevitable collision state (ICS) that must be rejected. The quadrotor icon in the figure denotes the vertex being expanded during search. The motion alternatives (solid lines) lead to successor states (circles). White circle is an ICS because it has no stop motion (black squares) that entirely lies inside the known region (unknown area is in grey). Detecting and rejecting ICS is a fast operation because it benefits from the pre-computation used in the other aspects of the proposed approach.

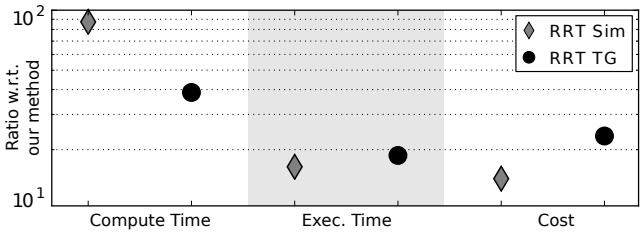


Fig. 6. **Planning comparison**. The figure compares two variants of the classical RRT algorithm with the approach presented here in terms of planning runtime, exploration execution time and the cost of overall motion. Note the vertical semilog axis. Horizontal axis represents quality metrics, namely computation runtime, execution (flight) time and overall cost of motion. Results are given as ratios with respect to our method, averaged over 30 simulated planning episodes with random initial and final states in the environment in Figure 1, with $R_P/C = 5$.

We evaluate the replanning performance w.r.t. the following three metrics that are relevant in applications we consider:

- *Compute time*: the amount of (wall) time, in seconds, the planner takes to generate a motion,
- *Execution time*: the amount of (wall) time, in seconds, required by the micro-UAV to execute the computed motion,
- *Cost* of the computed motion (1).

In the experiments, the robot traverses a previously unknown environment (an urban scene illustrated in Figure 1) between random start and goal states. Three sets of experiments are presented: a comparison of the approach with the RRT algorithm, a replanning performance study, and a review of the effects of the robot's perception radius on replanning.

A. RRT Comparison

Figure 6 presents a comparison of the classical RRT algorithm [7] adapted to this domain with a *baseline* configuration of our method: the state lattice roadmap coupled with AD* search algorithm [8]. Results are given as ratios with respect to baseline, averaged over 30 simulated planning episodes with random initial and final states in the environment in Figure 1, with $R_P/C = 5$. The average runtime of the baseline replanner in these experiments was 0.23 sec. with standard error 0.14 sec. The motions computed in these trials were successfully executed on physical "*nano*" quadrotors in our micro-UAV lab in order to confirm their flight feasibility.

Two variants of the RRT were used for fair comparison. One of the variants, labeled as "RRT Sim", is identical to the original RRT algorithm: it uses dynamics simulation during planning, which requires significant computation. The other variant, labeled "RRT TG", grows its tree using a differential flatness BVP solver [9]. While avoiding dynamics simulation, it tends to lead to increased cost of traversal by steering the system to the random state samples exactly.

We note that a comparison of our method to the alternative deterministic approaches, such as [1] and their successors, would be similar to "RRT Sim" result due to the dynamics simulation requirement.

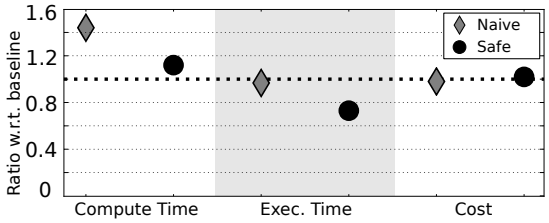


Fig. 7. **Replanning performance comparison**. The figure compares two modes of replanning with the baseline configuration, as averages over 30 simulated replanning experiments with random initial and final states in the environment in Figure 1, with $R_P/C = 5$.

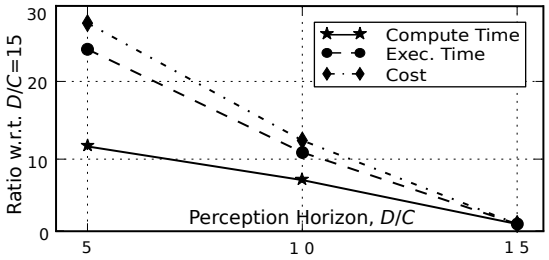


Fig. 8. **Effect of perception radius**. Measured dependence of replanning performance on the radius of perception radius is shown. Horizontal axis shows three different values of R_P/C. Vertical axis represents ratios of the three performance quantities compared.

B. Reuse of Computation in Replanning

A replanning comparison of two configurations of the presented approach with respect to the baseline configuration above are shown in Figure 7. First compared configuration, referred to as *naïve*, is based on $\mathcal{G}$ coupled with A* search and cannot reuse computation between replans. This results in significantly higher (almost 50%) computation time with respect to the baseline (left side of the figure). Execution time and cost of motion (middle and left) are about the same, as expected. The second compared configuration, labeled as *safe*, utilizes AD* search with ICS rejection to guarantee safety (Section IV-B). Figure 7 demonstrates small additional motion cost and runtime losses due to additional successor processing, while execution is significantly faster due to higher velocity motion. These results suggest that the latter is the preferred configuration of the method when applied to replanning in partially known environments.

C. Variation of the Perception Radius

The effect of the perception radius during motion in unknown environment is studied here. The same experimental setup and *baseline* planner was chosen as in earlier experiments. Execution time and overall cost of motion degrades almost exponentially with the reduction of the perception radius. However, replanning computation time degradation is significantly more graceful – nearly linear thanks to reuse of previous computation in the proposed approach. Naïve replanning without the reuse would result in similar, near-exponential degradation in compute time as well.

VI. CONCLUSIONS

Recent advances in micro-UAV's, such as quadrotors, have set off new research efforts in a number of areas related to autonomous flying robots. These vehicles are particularly difficult from the point of view of motion planning because of severely limited power and on-board computation, and complex dynamics motion constraints. We discussed efficient motion generation in partially known environments under the constraints of micro-UAV systems. The strengths of the approach include its ability to manage runtime-quality trade-off by virtue of pre-computing motion primitives that encapsulate the motion dynamics. We demonstrate autonomous micro-UAV motion in partially known, cluttered environments on the scale of at least three orders of magnitude greater than vehicle size, in real-time. Future work includes further improving motion sampling and search techniques in this framework, demonstrating the efficacy of the approach in autonomous micro-UAV exploration in search and rescue applications, as well as studying the applicability of these results to collaborating robot teams.

ACKNOWLEDGMENT

We thank ARL (grant W911NF-08-2-0004) and ONR (grant N00014-09-1-1031) for their support of this research. In addition we are grateful to Rattanachai Ramaithitima, Alex Kushleyev, Koushil Srinath and others for helpful discussions and their help with implementations and experiments.

REFERENCES

[1] J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10(2-4):121–155, 10 1993.

[2] E. Frazzoli and F. Bullo. On quantization and optimal control of dynamical systems with symmetries. In *Proceedings of the IEEE Conference on Decision and Control*, 2002.

[3] E. Frazzoli, M.A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control and Dynamics*, 25(1), 2002.

[4] J. H. Gillula, Haomiao Huang, M. P. Vitus, and C. J. Tomlin. Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1649–1654, 2010.

[5] R. He, A. Bachrach, M. Achtelik, A. Geramifard, D. Gurdan, S. Prentice, J. Stumpf, and N. Roy. On the design and use of a micro air vehicle to track and avoid adversaries. *International Journal of Robotics Research*, 29:529–546, 2010.

[6] J.-P. Laumond, S. Sekhavat, and F. Lamiraux. Guidelines in nonholonomic motion planning. *Robot motion planning and control*, 1998.

[7] S.M. LaValle and Jr. J.J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.

[8] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun. Anytime Dynamic A*: An anytime, replanning algorithm. In *AAAI*, 2005.

[9] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011.

[10] D. Mellinger, A. Kushleyev, and V. Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 477–483, 2012.

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

[12] S. Pancanti, L. Pallottino, and A. Bicchi. Motion planning through symbols and lattices. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.

[13] M. Pivtoraiko and A. Kelly. Kinodynamic motion planning with state lattice motion primitives. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2172–2179, 2011.

[14] E. Plaku, L. E. Kavraki, and M. Y. Vardi. Motion planning with dynamics by a synergistic combination of layers of planning. *IEEE Transactions on Robotics and Automation*, 26(3):469–482, 2010.

[15] A. Stentz and M. Hebert. A complete navigation system for goal acquisition in unknown environments. *Autonomous Robots*, 2(2):127–145, 1995.

[16] I.A. Sucan and L.E. Kavraki. Kinodynamic motion planning by interior-exterior cell exploration. In *Workshop on Algorithmic Foundations of Robotics*, pages 449–464, 2009.

[17] T.Fraichard and H.Asama. Inevitable collision states – a step towards safer robots? *Advanced Robotics*, 18 (10):1001–1024, 2004.