2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Efficient Two-phase 3D Motion Planning for Small Fixed-wing UAVs

Myung Hwangbo James Kuffner Takeo Kanade

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA, 15213, USA
{myung, kuffner, tk}@cs.cmu.edu

***Abstract*— We present an efficient two-phase approach to motion planning for small fixed-wing Unmanned Aerial Vehicles(UAVs) navigating in complex 3D air slalom environments. A coarse global motion planner first computes a kinematically feasible obstacle-free path in a discretized 3D workspace which roughly satisfies the kinematic constraints of the UAV. Given a coarse global path, a fine local motion planner is used to compute a more accurate trajectory for the UAV at a higher level of detail. The local planner is iterated as the vehicle traverses and refines the global path as needed up to its planning horizon. We also introduce a new planning heuristic for 3D motions of fixed-wing UAVs based on 2D Dubins curves, along with precomputed sets of motion primitives derived from the vehicle dynamics model in order to achieve high efficiency.**

I. Introduction

Recent advances in sensor devices, communications, and battery technology have made fixed-wing Unmanned Aerial Vehicles (UAVs) smaller in size and cost-competitive. Small-size UAVs are becoming an increasingly attractive solution for a variety of scientific, civil, and military applications. While some autonomous UAVs are employed successfully in security and military services, urban applications such as infrastructure monitoring demands small or even micro UAVs to maneuver within complex obstacle-filled environments. Operating a UAV under these conditions poses a number of difficult challenges. Environments cluttered with buildings and overhangs require high maneuverability and fast adaptation to dynamic and unknown obstacles. Fixed-wing UAVs require a relatively high minimum forward velocity to maintain lift. Thus, in order to respond quickly to unknown obstacles, high-performance real-time motion planning that respects the complex dynamic constraints of the vehicle must be accomplished without any significant delays.

We have been developing an unmanned aerial vehicle system [1] designed to accomplish a 3D air slalom scenario in Fig. 1. In this scenario, several labeled gates are arranged in the environment. The gates are placed either on the ground or in the air, and the UAVs are instructed to pass through each of the gates sequentially in the order specified. Although autonomously computing the UAV trajectory to accomplish this task is the focus of this paper, many other system components comprise the complete solution. The UAV maintains up-to-date perception of the environment and its own state through onboard cameras, IMU and GPS. Each slalom gate requires the UAV to pass through the target hoop with the correct 3D position as well as aligned pitch and yaw angles.

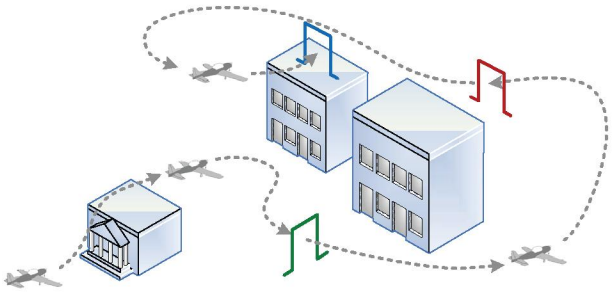

Fig. 1. 3D air slalom scenario for a small fixed-wing UAV: The vehicle needs to autonomously pass through each of the gates in the order specified.

Our goal is to build a real-time motion planner for 3D slalom scenarios that allow the UAV to operate reliably in the presence of fixed, moving or unknown obstacles. This paper presents a novel two-phase approach to motion planning for the 3D air slalom scenario. First, a coarse global planner computes a kinematically feasible obstacle-free path in a discretized 3D workspace which roughly satisfies the kinematic constraints of the vehicle. Then, a fine local motion planner computes a more accurate trajectory for the UAV at a higher level of detail. The local planner is iterated as the UAV traverses and refines the global path as needed up to its planning horizon. In order to achieve real-time performance, we have developed new planning heuristics for fixed-wing UAVs, and utilize precomputed sets of motion primitives derived from the UAV dynamics model. The result is an efficient planner that satisfies both the kinematic and dynamic constraints of the UAV while navigating in complex partially-known 3D environments.

II. Related Work

There have been a number of research efforts in both the robotics and computer animation literature related to planning 2D or 3D trajectories for guiding aircraft in known or unknown

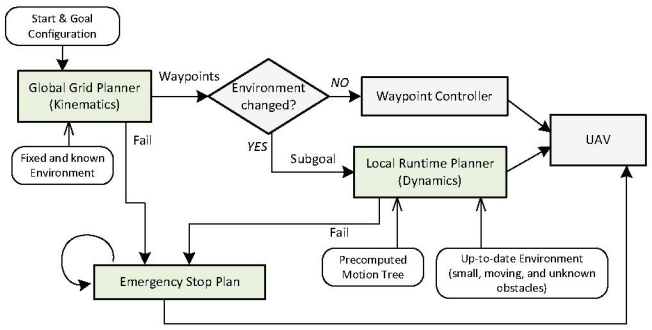

Fig. 2. Overview of two-phase planning architecture: The coarse global planning and the fine local planning compensate each other in terms of associated constraint, size of planning horizon, and the type of obstacles considered.

environments. Previous research for UAVs in robotics has primarily explored vision-based navigation for UAVs [2], or basic obstacle avoidance during flight [3]. A 3D local trajectory planner has been presented in [4] for autonomous navigation with a predefined global path. They also utilize a two-stage strategy consisting of a decision mode and a trace mode. The controller design in trace mode is heuristic, so applying the method to other dynamic systems can sometimes be difficult.

A more complicated nonlinear and high-dimensional hybrid system control architecture [5] was developed for an autonomous helicopter. This planner utilized libraries of trim trajectories, along with an efficient online replanning framework based on Rapidly-exploring Random Trees (RRTs) [6], which has been broadly used to plan motions for dynamic systems subject to dynamic constraints.

In computer animation, reactive steering behaviors for flocking, grouping and avoiding obstacles are presented in [7]. Although highly efficient, these techniques only use local information. An autonomous vehicle animation and planning system using online search and trajectory precomputation was developed in [8]. Precomputed search trees were also used to assemble planned sequences of human motion for animated characters navigating in complex virtual environments [9].

In our work, we focus on developing an efficient motion planner for solving the 3D air slalom task, with particular attention to computing global paths that satisfy both the kinematic and dynamic constraints of the UAV. The rest of the paper is organized as follows: an overview of our two-phase motion planning method is discussed in Section III; the details for the global grid planner and the local runtime planner with motion primitives are described in Section V and Section IV respectively; experimental results in simulation is presented in Section VI.

III. Two-phase motion planning

Two-phase scheme in motion planning is a natural and popular method to overcome the planning complexity of high order dynamical systems [10] [11]. At the first stage constraints imposed on a system are relaxed to take sidestep from its original planning complexity. For examples, limited range of state, system dynamics, or nonholonomic constraint is more likely to be ignored at this stage. The relaxation is taken up at a level where the reduced order system can take advantage of well-established planning algorithms [12]. The relaxed constraints are recovered at the second stage so that a final solution is feasible to the original system. Tge plan found at the first stage makes an initial guess or confines search space to its vicinity for a more sophisticated planner at the next stage.

Two main components of our planning architecture in Fig. 2 are the coarse global planner and the fine local planner. They compensate each other in terms of associated constraint, size of planning horizon, and the type of obstacles considered; The global planner takes kinematic constraints of the vehicle into account but neglects dynamic constraints which are considered in the local planner at the subsequent stage. Then the local planner selects a subgoal within its planning horizon from the optimal path the global planner have found over all workspace.

The global planner works in 3D grid cells. The size of each grid cell as well as allowable connections between grid cells are elaborately designed to capture UAV's kinematics in workspace discretization process. As a result its output neither violate given kinematic constraints such as a minimum radius of curvature and a maximum climbing angle nor be trapped in local minima by virtue of A* search. The local planner is built on the top of the global planner. It complements the drawbacks of the global planning, which are coarse discretization of the configuration space($\mathcal{C}$-space) and negligence of small or moving obstacles. With densely sampled motion primitives a finer level of connection between two configurations is achievable. This computationally intensive planning method, however, limits the size of the planning horizon. In obstacle-free regions from the global planner a general waypoint controller is enough to follow the path. On the other hand, when environmental change happens nearby the vehicle at a sudden the local planner is switched to run so that vehicle dynamics is taken into account to generate a new dynamically feasible path associated with aggressive vehicle motions required for avoiding unexpected obstacles.

When either planner fails to find a path with any reason an emergency stop plan is called up immediately to escape the critical situation. This backup plan moves the vehicle to a pre-defined safe region(usually up to the sky). It must run all the time at the background whenever either vehicle state or environment is updated.

IV. Global grid planning on a kinematically feasible search tree

One of basic techniques in robot navigation is planning a motion on a discretized workspace like 2D grid representation for mobile robots. For an air vehicle we first perform motion planning on a discretized 3D grid. Instead of extending the conventional 4 or 8-connected neighborhood in 2D grids to 3D grids, a search graph for 3D grids is built along with a new node definition and node connection rules which capture

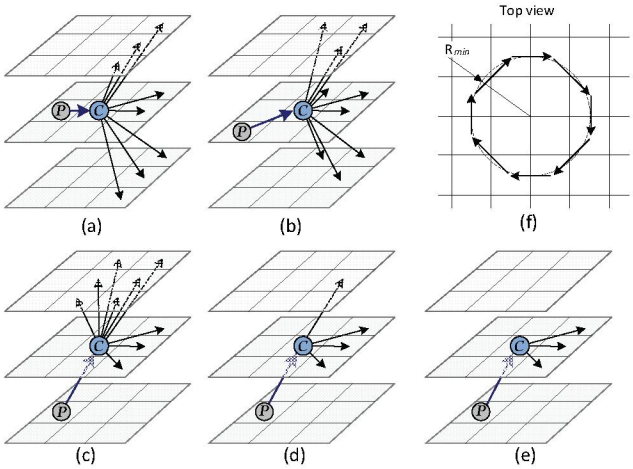

Fig. 3. Node definition and node expansion comprising a search tree for the global planner: A node is defined by the location in 3D and *parent direction*. One node corresponds to a discretized point in the configuration space $\mathcal{C} = (x, y, z, \theta, \phi)$. The given vehicle kinematic constraints, $R_{h,min}$ and μ_{max}, can be captured through the design of node expansion rule in 3D grids. For instance, vehicle forwarding motion enforces *horizontal forward-only* expansion in (a) and (b). The vehicle maneuverability decides one of possible vertical expansion in (c),(d), and (e).

the vehicle kinematics. Then A* algorithm on the search tree guarantees to find an optimal path between starting and goal nodes in terms of its designed cost. Consequently the output path automatically satisfies given kinematic constraints of the vehicle because the search tree is already kinematically feasible in construction. Note that moving obstacles or obstacles relatively smaller than the grid size are ignored at this stage.

A. Node definition and node expansion

From 3D grid representation of the workspace a node used in the construction of a search tree is defined by the location in 3D space as well as the *parent direction* indicating where the node comes from. The parent direction is the directional vector connecting a current grid location from one of 26 possible locations of a parent node. So each node can be considered as a point in the discretized configuration space $\mathcal{C} = (x, y, z, \theta, \phi)$. In other words a node corresponds to the vehicle flying at (x, y, z) with pitch angle θ and yaw angle ϕ. The vehicle attitude is discretized very coarsely with no mention of roll angle ψ which cannot be taken into account in this grid representation.

Two vehicle kinematic constraints are considered in this planning stage, that is, horizontal minimum turning radius $R_{h,min}$ and vertical maximum climbing angle μ_{max}. As shown in Fig. 3 the proper choice of node expansion rules and grid cell size can produce kinematically constrained vehicle motions in 3D grids. For example, *forward-only* expansion for the horizontal vehicle motion in Fig. 3(a)(b) illustrates the fact that a fixed-wing air vehicle needs to keep a minimum forwarding velocity. The vertical movement of the vehicle can be one of Fig. 3(c)(d)(e) according to the level of vehicle maneuverability. Especially when the straight upward or downward motion is allowed as in Fig. 3(c) the grid planner can find a path that performs a *looping* feat.

Note that node expansion rules should be designed to encode the given maneuverability of the vehicle of interest into a discretized grid space. For the rest of the paper we focus on the expansion rules shown in Fig. 3(a)(b)(d) which best describes the behavior of a small fixed-wing UAV.

B. Setting grid cell sizes

The grid cell sizes in x, y and z directions can be set so that a directed search tree is kinematically feasible for given constraints $R_{h,min}$ and μ_{max}. It is obvious to set the grid sizes in x and y equal. Fig. 3(f) shows that repeating horizontal forward-only node expansion generates a circle of which radius is one and half times the grid size in x or y. And the ratio between grid sizes in x and z decides the climbing angle of the vehicle when straight upward or downward motion is prohibited as in Fig. 3(d). The grid size in each direction is set by

$$S_x = S_y = \frac{1}{1.5} R_{h,min}, \quad S_z = S_x \tan^{-1}(\mu_{max}) \qquad (1)$$

where S_i is the grid size in i direction. Finally the found path from A* is always kinematically feasible because every branch in the search tree satisfies the given kinematic constraints.

V. Local runtime planning with motion primitives

There are several demanding situations that the global planner cannot cope with mainly due to its coarse configuration space representation. For example, grid sizes given by Eq. (1) are definitely larger than that of motion resolution required in general. Usually sub-meter level accuracy is demanded in control and motion planning for UAV urban application. Moreover the obstacles smaller than the grid size cannot be dealt with appropriately. The local planner is, therefore, built on the top of the global planner to overcome these problems. A finer level of motion planning between two configurations is achievable via interconnecting motion primitives which are densely sampled from the vehicle dynamics. Because it needs to response to local environmental changes as swiftly as possible the local planning should be involved with vehicle dynamics beyond the kinematics to generate aggressive motions as required.

The local planner we propose is sampling-based motion planning under differential constraints [15]. Searching for a feasible path via Dynamic Programming is vulnerable to the exponential growth of a search tree. The use of a good and computationally efficient heuristic function, an estimate distance to a goal state, is quite essential to prevent intractability. Two new heuristic functions based on 2D Dubins curve are proposed for motion planning in different $\mathcal{C}$-spaces.

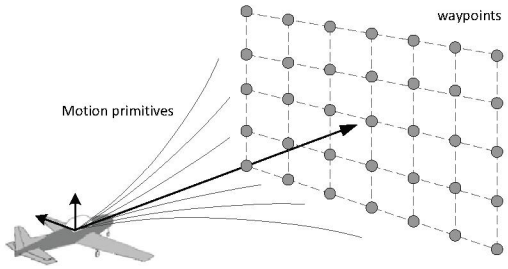

Fig. 4. Control-based action sampling: A grid of waypoints in front of the vehicle at x_o is given to a discretized feedback dynamic system $f_d(x,u)$ as a set of inputs $\mathcal{U}$. The resulting motion segments comprise the sampled reachable set $R(x_o,\mathcal{U})$ from the state x_o and recorded in a look-up-table.

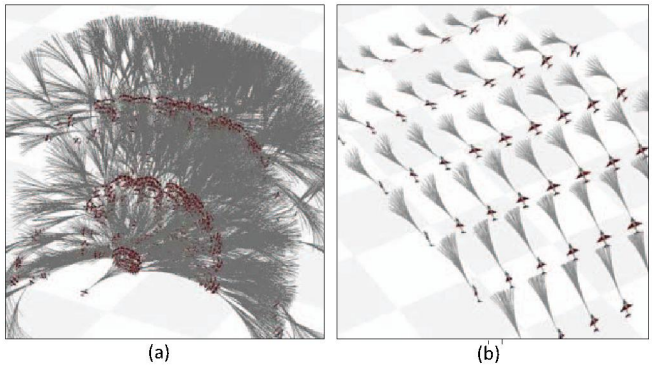

Fig. 5. Generation of motion primitives: (a) A reachability tree for a fixed-wing UAV stopped at the third depth (b) Precomputed motion primitives at different starting states

A. Generating motion primitives

The motion primitives in Fig. 5(b) is a sampled set of feasible vehicle motions which hold the dynamic characteristics of the vehicle at a given state. More complex dynamic behaviors of the vehicle can be produced via interconnecting these motion primitives. We are assumed to have both a dynamic model of the vehicle and a waypoint controller to track a point in 3D which can be expressed as $x_{k+1} = f_d(x_k, u_k)$ in a discretized feedback dynamic system. Otherwise motion libraries from real experiments would be needed. A grid of waypoints in front of the vehicle is given as a set of inputs to generate motion primitives. Some of waypoints cannot be achievable because they are neither controllable state nor dynamically feasible. See Fig. 4 for detail.

The motion primitives are precomputed and saved in a look-up-table for the sake of fast computation. Since the size of memory is easily blown up in terms of state dimension we record only a final vehicle state and a middle point location for collision checking. Fig. 5 shows a reachability tree for a fixed-wing UAV stopped at the third depth and precomputed motion primitives at different vehicle states.

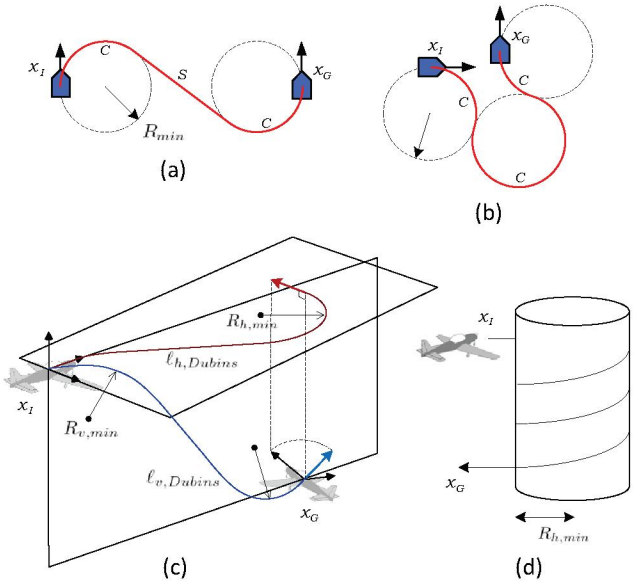

Fig. 6. Two heuristic functions to estimate the optimal cost-to-go: (a)-(b) Examples for 2D Dubins curve which is the shortest length path between $\mathbf{x}_I, \mathbf{x}_G \in SE(2)$, (c) h_2 in $\mathcal{C} = \{x, y, z, \theta, \phi\}$, (d) h_1 in $\mathcal{C} = \{x, y, z, \phi\}$

B. Forward dynamic programming: greedy best-first search

In discrete planning one may use any graph search algorithm such as depth-first or A* to find a path from a starting state to a goal state. A search tree is incrementally constructed through state transition which corresponds to a set of motion primitives. A specific choice among search algorithms comes with the fact that in our case the branching factor b, the same as the number of motion primitives being used at state transition, at each node is quite big ($b = 40$ on average). And a goal for the local planner is placed at d steps ($d = 10 \sim 15$) away from a start in order to give enough chance in avoiding obstacles. Since the goal is at the depth of d at a search tree, if we build an exhaustive tree like Fig. 5(a), the tree size grows exponentially with respect to $d(O(b^d))$. Hence, in real time application a greedy search method is preferred while the path optimality is sacrificed.

A greedy best-first search sorts the queue according to a heuristic function so that a node which is estimated to be closer to a goal is extended first. The path obtained in this way is not necessarily optimal. In many cases, however, far fewer nodes are explored, which results in much faster running times. Therefore, we invent new heuristic functions which are more probable to avoid the worst case $O(b^d)$ in time complexity and space complexity.

C. Heuristic functions based on 2D Dubins curve

Dubins curve [13] as a search heuristic is useful because the kinematics of Dubins car is very close to that of a fixed-wing air vehicle except for the dimensionality. Dubins curve is a shortest length path between two configurations $\mathbf{x}_I, \mathbf{x}_G \in$

$SE(2)$ when the car is constrained by a minimum turning radius ρ_{min}. Fig. 6(a)(b) shows this shortest path is composed of no more than three motion primitives which are a straight line and a circular path. The path length, called *Dubins metric*, is obtainable without expensive computation thanks to phase partitions [14]. In practice the discontinuity in Dubins metric needs appropriately dealt with when a sampling-based motion planning is used. A small change in configuration cause a sudden jump in Dubins metric when crossing partitions. To alleviate this problem we allow a small margin for the heading angle of a goal when computing optimal cost-to-go.

Two new heuristic functions, h_1 and h_2, are proposed based on the Dubins curve. They differ by whether the pitch angle θ of the vehicle is considered or not in their configuration space. h_1 is for the case that vehicle motion is relatively smooth so pitch angle at a goal can be thought as zero all the time.

1) h_1 in $\mathcal{C} = \{x, y, z, \phi\}$: First consider a simple kinematic equation [15] for a fixed-wing air vehicle. Two inputs are the maximum yaw rate u_w and the altitude change u_z.

$$\dot{x} = \cos\theta, \quad \dot{y} = \sin\theta, \quad \dot{\theta} = u_w, \quad \dot{z} = u_z \tag{2}$$

The constant inputs to Eq. (2) generate a helical motion as seen in Fig. 6(d). We use the length of the path as a distance function between two configurations with two parameters, $R_{h,min}$ and μ_{max}. The path length can be easily approximated as follows; First compute 2D Dubins metric $\ell_{h,Dubins}$ on the horizontal plane by ignoring altitude change in z. Then, keep adding a full circular motion of radius $R_{h,min}$ until getting enough horizontal traveling distance to satisfy the maximum climbing angle μ_{max}.

1 $h_1 \leftarrow \ell_{h,Dubins} \quad \Delta z \leftarrow |z_I - z_G|$
2 While $\Delta z / h_1 > \sin(\mu_{max}), \quad h_1 \leftarrow h_1 + 2\pi R_{h,min}$
3 $h_1 \leftarrow \sqrt{h_1^2 + \Delta z^2}$

2) h_2 in $\mathcal{C} = \{x, y, z, \phi, \theta\}$: The 3D vehicle trajectory between two configuration is decomposed onto two orthogonal planes. 2D Dubins curves are computed from both planes with projected configurations as in Fig. 6(c). The horizontal plane parallel to the ground and the vertical plane is oriented toward the goal position. The horizontal motion is a Dubins curve that aligns the yaw angle ϕ with a minimum radius $R_{h,min}$. The vertical motion is also one that aligns the pitch angle θ with a minimum radius $R_{v,min}$. An estimate distance to a goal is computed by combining both paths as

$$h_2(x_I, x_G) = \sqrt{{\ell_{h,Dubins}}^2 + w^2 {\ell_{v,Dubins}}^2} \tag{3}$$

where $w = |\sin(\Delta z / \sqrt{\Delta x^2 + \Delta y^2})|$ accounts for the climbing angle between two vehicle positions. Geometrically w makes the vertical Dubins curve projected onto the gravity z direction. Therefore, in case of no altitude change, $\Delta z = 0$, h_2 regards the air vehicle as the same as the Dubins car.

VI. Simulation Results

The nonlinear F16 dynamic model available at [16] is used for simulation model of a fixed-wing UAV. The low fidelity model which does not include the effect of the leading edge flap gives a complete decoupling between longitudinal and lateral direction. A simple PD waypoint controller that can track a given waypoint makes the UAV considered as a feedback dynamic system. To fit the F16 model to a small air vehicle all units are scaled down by 100. By inspecting behaviors of the dynamical system all kinematic constraints required for motion planning are given as follows: vehicle speed v = 2m/s, minimum turning radii $R_{h,min} = R_{v,min} =$ 15m, maximum climbing angle μ_{max} = 45°.

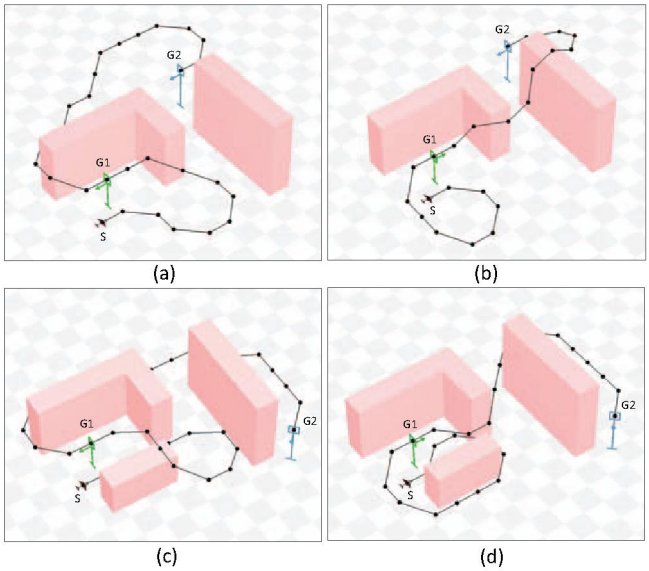

Fig. 7. Obstacle-free paths found from the global grid planner: Two air slalom gates are placed for the UAV to pass through in order. The arrow drawn at the gate indicates the gate direction. The first gate $G1$ has opposite orientations in rows while the second gate $G2$ is placed at different locations in columns. The vehicle is allowed to move horizontally with forward-only expansion and move vertically with no straight upward or downward motion. See Fig. 3(a),(b) and (d).

According to Eq. (1) grid cell sizes for the global planning are set to $S_x = S_y = S_z =$ 10m, which is two-thirds of $R_{h,min}$. Fig 7 shows how the global grid planner works in the air slalom scenario. The arrow on the gate indicates which way the vehicle needs to pass through. As expected the grid planner returns totally different paths as the gate has different directions or is placed at different locations. The vehicle is allowed to move horizontally with forward-only expansion and move vertically with no straight upward or downward motion as in Fig. 3(a),(b) and (d).

The Euclidean distance is no longer a good heuristic even in the grid planner because the node in its search tree contains a direction. Precomputation of the optimal cost-to-go h^* over the region around a node helps much reduce planning time. The h^* is precomputed by the Dijkstra algorithm over a 9×9×9 grid region centered at a node. If the goal N_G is outside of the

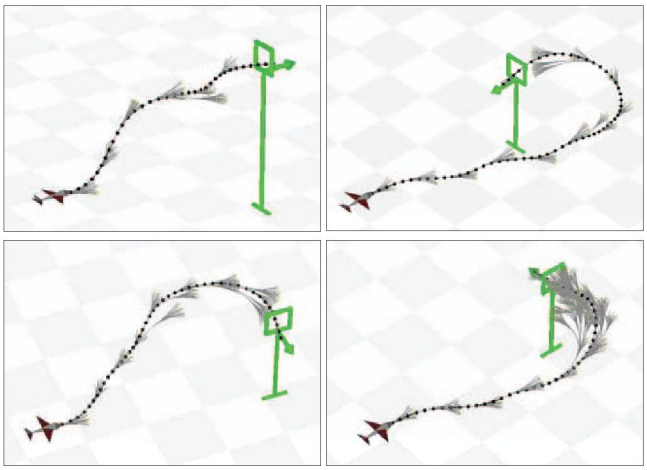

Fig. 8. Examples for the local path planner in free of obstacles: The greedy best-first search explores motion nodes toward the gate in a greedy way. The heuristic function h_2, the extended 3D Dubins metric, guides the node expansion efficiently toward the goal state. All motion expansions queued in the search tree are displayed in order to see which node has a best cost-to-go at every iteration until it reaches the goal region. The gate has different yaw angles ϕ in figures.

TABLE I

USE OF THE PRECOMPUTED HEURISTIC IN A* GLOBAL PLANNING

	Euclidean dist	Precomputed h^*	Speed-up
Iteration	65504	12038	5.44
No. of nodes visited	96230	23142	4.15
No. of nodes revisited	483562	83207	5.81
Avg. runtime(sec)	0.475	0.135	3.50

The optimal cost-to-go h^* is precomputed over 9×9×9 grid region. The average values are taken after 100 experiments with randomly selected initial and goal nodes. The dimension of 3D grids for the whole workspace is 50×50×50 and the same obstacle environment in Fig. 4 is used.

region, the cost is the sum of the precomputed cost at N^* and the Euclidean distance between N_G and N^* where N^* is a closest node to N_G within the region. Due to the symmetries of the 3D grids one of 8 octants is enough. Table-I shows the speed-up in the global grid planning when the precomputed heuristic is used instead of the Euclidean distance. On average it makes the planner run 3.5 times faster.

For the local motion planning 40 motion primitives on average are sampled at every 5 degrees for the pitch θ and rolling ψ angle of the vehicle. Fig. 8 shows the basic performance of the local runtime planner in finding a feasible path between two configurations when the heuristic function h_2 in Eq.3 is used. The greedy best-first search explores motion nodes toward the goal state in a greedy way while the vehicle also heads on toward the goal direction. More node expansions occurs near the gate in effort to get into the goal region, which is fine adjustment to compensate discrete nature of the sampled motion. All motions queued in the search tree are displayed in order to see which node has a best cost-to-go at every iteration until it reaches the goal region. The small number of expanded

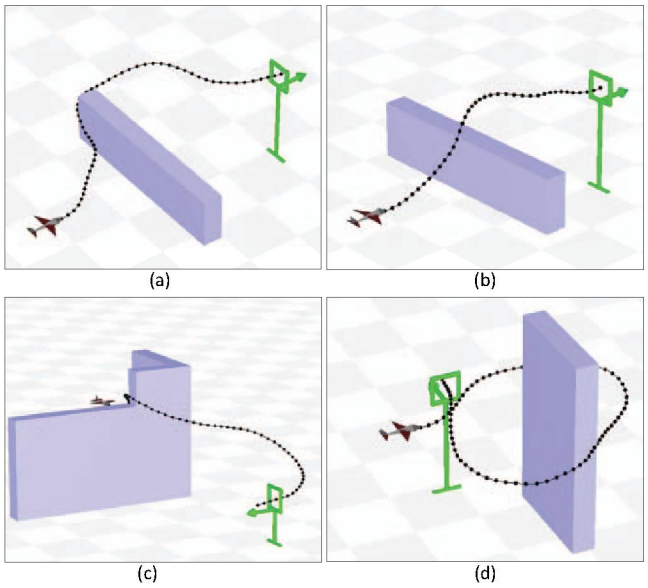

Fig. 9. Obstacle avoidance of the local runtime planner: Two different heuristic functions, h_1 and h_2, are used respectively in (a) and (b) under the same environment. In (a) the vehicle moves in a horizontal way to avoid the obstacle because h_1 needs a enough traveling distance for the change of altitude. In (b), on the other hand, the vehicle moves vertically in obstacle avoidance because h_2 can account for the pitch angle θ. In (d) the gate is too close for the vehicle to make the gate direction so the heuristic function h_2 drives the vehicle to take a P-turn.

nodes during search proves the effectiveness of the heuristic function we proposed.

Fig. 9 demonstrates how the local planner works in the presence of obstacles. In Fig. 9(a) and (b) two different heuristic functions, h_1 and h_2, are used respectively under the same environment. With h_1 the vehicle moves horizontally to avoid the obstacle because h_1 needs a enough traveling distance for the change of altitude. On the other hand the vehicle moves vertically during obstacle avoidance because h_2 can takes the pitch angle θ into account. If the gate is too close for the vehicle to make the gate direction the heuristic function h_2 drives the vehicle to take a P-turn like Fig. 9(d).

In the context of the 3D air slalom scenario we simulate the performance of the two-phase planner which combines the global grid planning with the local runtime dynamic planning in presense of unknown obstacles. In Fig. 10 the global grid planner finds the kinematically feasible path drawn by a dotted line with *a priori* known obstacles and gates. While the vehicle follows the given global path by a waypoint controller new unknown obstacles appear at points A and C respectively. Subgoals are set at B and D which are on the global path and 10 steps away from the current vehicle position. The local planner generates the new obstacle-free paths via interconnecting motion primitives.

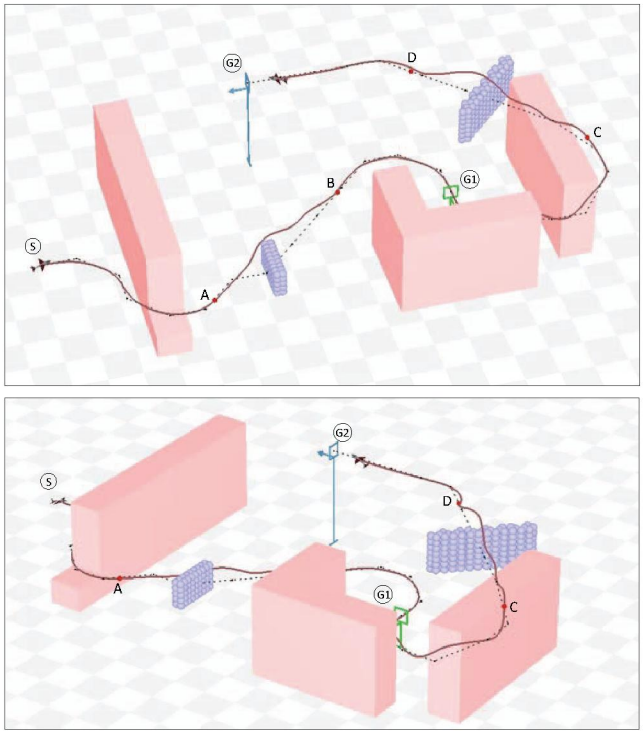

Fig. 10. An example of the two-phase motion planning which combines the global grid planning with the local runtime dynamic planning: The solid boxes in light red are fixed and known obstacles which the global planner considers. Two sets of spheres in light blue are unknown for global planning and can be detected when the vehicle arrives at points A and C respectively. A dotted line is a path found from the global grid planner. In $\overline{AB}$ and $\overline{CD}$ the local planner generates new obstacle-free paths. The actual path the vehicle have taken on is drawn by a solid brown line. Top and bottom figures shows the same result in different view angles.

VII. Summary and Future Work

We have presented an efficient motion planning method for small fixed-wing UAVs in complex 3D environments. Two-phase planning (global and local planners) are used to plan motions in large cluttered areas while respecting the UAV's kinematic and dynamic constraints. By adding directionality to the conventional 3D grid cell, a search tree in 3D space can be constructed only with feasible node connections which take into account for UAV's forward-only motion constraints. The path found in this search tree satisfies two important kinematic constraints of the UAV: the minimum turning radius and the maximum climbing angle. We also proposed two new heuristics to estimate the optimal cost-to-go during local planning, involving an approximation of the shortest path of the UAV in 3D by projection of the full configuration space into two planes which enable the application of Dubins curves. Utilizing precomputed sets of available motion primitives for discretized vehicle states, a best-first search local planner finds obstacle-free paths that satisfies the dynamic constraints. We have demonstrated the efficiency and effectiveness of the two-phase planner in complex simulated 3D air slalom scenarios. Our future work involves integrating the perception, planning, and control systems and demonstrating our result on the actual small fixed-wing UAV platform.

References

[1] T. Kanade, O. Amidi, and Q. Ke, "Real-time and 3d vision for autonomous small and micro air vehicles," *IEEE Conf. on Decision and Control*, pp. 1655–1662, Dec. 2004.

[2] B. Sinopoli, M. Micheli, G. Donata, and T. Koo, "Vision based navigation for an unmanned aerial vehicle," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2001.

[3] R. Zapata and P. Lepinay, "Flying amoong obstacles," in *European Workshop on Advanced Mobile Robots*, 1999.

[4] J. Sasiadek and I. Duleba, "3d local trajectory planner for uav," *Journal Journal of Intelligent and Robotic Systems*, vol. 29, no. 2, pp. 191–210, Oct. 2000.

[5] E. Frazzoli, M. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *AIAA Journal of Guidance, Control and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[6] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.

[7] C. Reynolds, "Steering behaviors for autonomous characters," in *Game Developers Conference*, 1999.

[8] J. Go, T. D. Vu, and J. Kuffner, "Autonomous behaviors for interactive vehicle animations," *Graphics Models*, vol. 68, no. 2, pp. 90–112, 2006.

[9] M. Lau and J. Kuffner, "Precomputed search trees: Planning for interactive goal-driven animation," in *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2006.

[10] J. E. Bobrow, "Optimal robot path planning using the minimum-time criterion," *IEEE Trans. on Robotics and Automation*, vol. 4, no. 4, pp. 443–450, 1988.

[11] Y. Kuwata and J. How, "Three dimensional receding horizon control for uavs," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Aug. 2004.

[12] J. C. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer Academic Publishers, 1991.

[13] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, pp. 497–516, 1957.

[14] X.-N. Bui, P. Soueres, J.-D. Boissonnat, and J.-P. Laumond, "Shortest path synthesis for dubins nonholonomic robot," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1994.

[15] S. M. LaValle, *Planning Algorithms*. Cambridge University Press (also available at http://msl.cs.uiuc.edu/planning/), 2006.

[16] R. S. Russell, *Nonlinear F-16 Simulation using Simulink and Matlab*. http://www.aem.umn.edu/people/faculty/balas/darpasec/SEC.Software.html, 2003.