2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Stereo vision-based obstacle avoidance for micro air vehicles using disparity space

Larry Matthies[1], Roland Brockers[1], Yoshiaki Kuwata[2], and Stephan Weiss[1]

***Abstract*— We address obstacle avoidance for outdoor flight of micro air vehicles. The highly textured nature of outdoor scenes enables camera-based perception, which will scale to very small size, weight, and power with very wide, two-axis field of regard. In this paper, we use forward-looking stereo cameras for obstacle detection and a downward-looking camera as an input to state estimation. For obstacle representation, we use image space with the stereo disparity map itself. We show that a C-space-like obstacle expansion can be done with this representation and that collision checking can be done by projecting candidate 3-D trajectories into image space and performing a z-buffer-like operation with the disparity map. This approach is very efficient in memory and computing time. We do motion planning and trajectory generation with an adaptation of a closed-loop RRT planner to quadrotor dynamics and full 3D search. We validate the performance of the system with Monte Carlo simulations in virtual worlds and flight tests of a real quadrotor through a grove of trees. The approach is designed to support scalability to high speed flight and has numerous possible generalizations to use other polar or hybrid polar/Cartesian representations and to fuse data from additional sensors, such as peripheral optical flow or radar.**

I. Introduction

With their ability for VTOL, hover, and low or high speed flight, rotorcraft micro air vehicles (MAV) have potential applications in a wide range of reconnaissance-oriented scenarios outdoors, indoors, and underground, in well-lit or dark conditions ranging from wide open to highly cluttered. This requires sensing, world representation, and motion planning architectures that support this range of speeds and clutter and that can fuse multiple obstacle detection sensors to span this range of conditions. This paper focuses primarily on sensing and representation methods for outdoors that are scalable to high speed flight.

Although considerable progress has been made lately on MAV navigation with active optical range sensors like lidar and structured light [1], [2], these sensors have limitations in terms of size, weight, power, and field of regard for outdoor operation in direct sunlight. Outdoor scenes typically have enough visual texture to enable camera-based perception systems, which have potential for wide field of regard in both azimuth and elevation, with very low size, weight, and power implementations. Representations used to date include image space data products used for reactive navigation [3], [4], [5], [6], Cartesian voxel data structures used with deliberative planners [1], [2], [7], and a few examples of polar data structures [8], [9]. For outdoor perception of large areas and flight that may reach high speed, image space and polar representations have advantages in compactness and matching the natural angular and range resolution of many sensors. Polar representations can have the radial axis parameterized by inverse range, which gives a compact representation of all space from some minimum range to infinity [10] and is a natural match to the range resolution characteristics of camera-based perception with stereo vision and optical flow. The combination of dense, forward-looking stereo, dense peripheral optical flow, and hybrid representations that combine advantages of image space, Cartesian grid, and polar grids appears to be an attractive synthesis for outdoor operation and could be augmented with other sensors to go inside or underground. To our knowledge, this synthesis has not been explored.

Fig. 1. Asctec Pelican quadrotor flying through a groove of trees.

This paper takes a step toward this vision by demonstrating autonomous, outdoor flight among trees with forward-looking stereo vision for 3D perception, inverse range image space representation (scaled disparity), and the first algorithm to combine deliberative motion planning with image space collision checking for MAVs. This approach has very low memory requirements and enables very efficient collision checking. It lays a foundation for future extensions to add peripheral optical flow and for several extensions to the representational framework. Subsequent sections elaborate on related work and motivation for our approach (section II), sketch the architecture of our system (section III), describe the approaches to collision checking, motion planning, and motion execution (sections IV-V), describe experimental evaluations done in simulation and on live flight through a grove of trees (section VI), and discuss conclusions and planned extensions (section VII).

[1]Larry Matthies, Roland Brockers and Stephan Weiss, are with the Jet Propulsion Laboratory, California Institute of Technology.
{lhm, brockers}@jpl.nasa.gov, stephan.weiss@ieee.org
[2] kuwata@alum.mit.edu

II. Related Work

Significant progress has been made recently on deliberative obstacle avoidance using active optical range sensors, such as single axis scanning lidar (e.g. Hokuyo) and RGB-D structured light [1], [2]. The lack of a second axis for scanning lidar is a significant limitation and structured light sensors are ineffective in sunlight. Very compact, electronically beam-steered radar with large maximum range is under development for this application, but again has a single-axis scan [11]. Optical flow can cover a wide field of view and has been used for reactive obstacle avoidance [4], [5]; however, this only provides information when the aircraft is moving, and poor estimation of time to collision near the focus of expansion limits the ability to perceive obstacles dead ahead. Stereo vision can provide 3D perception around the focus of expansion whether the aircraft is moving or not, but so far with relatively short lookahead distance [7]. Stereo and optical flow have been used together in purely reactive obstacle avoidance based on sparse perception with point features [12].

The most common obstacle representations for MAV are image space data products that serve reactive obstacle avoidance [4], [5], [6] and Cartesian voxel data structures that serve deliberative planning [1], [7], [2]. Reactive obstacle avoidance with image space data structures use very little memory and computation, but have limited ability to reason about 3D structure and vehicle dynamics. Cartesian voxel data structures enable much greater 3D reasoning, have mature temporal fusion algorithms for error reduction, and have been used to plan high speed, aggressive maneuvers [13]; however, they use much more memory and computing time. Uniform voxel sizes are also problematic for representing both very near and very far objects, which can lead to more complex, multi-resolution data structures. Polar representations parameterized by azimuth and range have been used in a few efforts because they naturally capture range-dependent variations in angular and range resolution [8], [9]; however, these efforts were only tested in simulation and [9] only represented sparse, discrete obstacles. Some stereo vision-based navigation systems for ground vehicles have had characteristics that are interesting for MAVs. A simple version of collision testing and path planning with the stereo disparity image was done in [14]. A 2D polar grid-based representation in the ground plane was used in [10], where the radial axis was parameterized as inverse range; this matched the angular and range resolution characteristics of stereo and gave a compact representation of all space from a minimum range to infinity. This was used to represent and reason about distant obstacles, while a 2D Cartesian map was used for nearby obstacles. Inverse range is equivalent to nearness fields that have been used for reactive MAV obstacle avoidance with optical flow [5].

Reactive obstacle avoidance controllers have been based on image space nearness fields computed from optical flow [5] and trained from human behavior via imitation learning [6]. Recent deliberative planners have used techniques including anytime, incremental A* for non-symmetric vehicles moving slowly in cluttered spaces [15] and RRT* with path optimization [13] and lattice search with precomputed motion primitives [16] for fast, aggressive maneuvers.

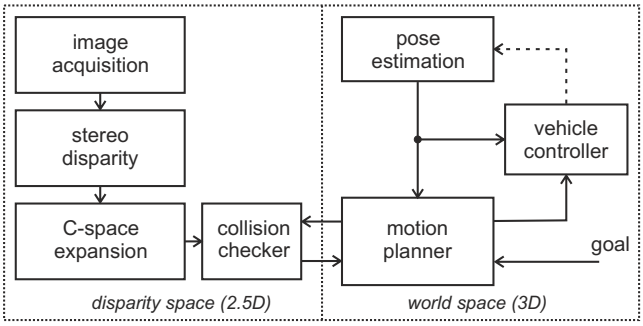


Fig. 2. System architecture

III. Vision based autonomous Navigation System

We use cameras for wide, two-axis fields of regard in outdoor operation. Our approach is loosely motivated by owl vision systems, which have two eyes with a wide area of binocular overlap and wide, monocular peripheral fields. Although it is unclear whether or to what extent owls have binocular depth perception [17], for MAVs this would provide forward-looking, stereo-based depth perception whether or not the aircraft is moving and peripheral depth perception from optical flow as long as the aircraft is moving. In principle, peripheral optical flow can provide long-range depth perception from long motion baselines generated by aircraft movement, but with a blind spot around the focus of expansion. Stereo vision is well-known to provide short-range depth perception; however, the range limit of stereo depth perception can actually be quite large, although with low range accuracy. Since high accuracy is not needed for distant objects, we expect that this will be useful to fill in long-range depth perception around the focus of expansion, as well as for short range depth perception and as an input to state estimation. In this paper, we have only implemented forward-looking stereo. We also include and have implemented a downward-looking camera for state estimation.

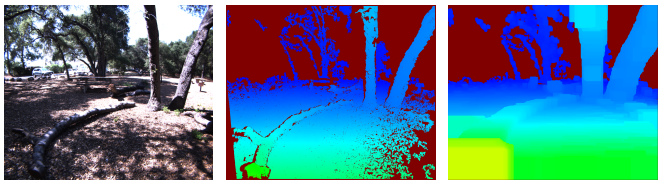
To address scalability large spaces and high speed, we are exploring polar representations. This is actually a family of possibilities, ranging from 2D image space of one camera, to 2D surfaces of virtual cylinders or spheres around the vehicle, to 2D azimuth-range cells, to 3D polar volumetric structures analogous to Cartesian voxel maps. Parameterization of the radial axis can be range or inverse range. In this paper, we instantiate the simplest case for a stereo system, which is an image space representation of inverse range (i.e. scaled stereo disparity). Ultimately, we expect that motion planning is best done in 3D space with a vehicle dynamics model as part of trajectory generation. As a matter of convenience, we have adapted the closed-loop RRT (CL-RRT) planner of [18] to quadrotor dynamics and full 3D search. RRTs have been used with 2D polar cellular

maps in [9], but we have not seen deliberative planning done with a disparity space obstacle representation except for the ground vehicle system in [14]. Here, we generalize their approach by generating 3D trajectories with CL-RRT, applying an obstacle expansion to the disparity map to create a form of image-based configuration space obstacle map, and projecting the 3D trajectories into image space. At this point, a z-buffer-like operation detects trajectories that are in front of objects, intersect objects, or appear to go behind objects, and trajectories can be pruned accordingly. This is illustrated in Figure 2. This concept can be generalized to a wrap-around virtual image space representation, for example on the surface of a virtual cylinder. This would enable fusion of inverse range from stereo and optical flow, as well as a persistent representation that enables temporal fusion and keeping track of objects that are no longer in view. It can also be generalized to various degrees of multi-layer representation.

The next two sections describe the details of our current implementation with a disparity map from one stereo camera pair.

IV. Image Based Collision Checking

For efficiency reasons, collision checking is performed directly in disparity space. When a new disparity image is obtained from stereo, C-space expansion is applied in the disparity domain, allowing to treat the UAV as a single point in space for planning purposes. During motion planning, small trajectory segments are verified by projecting them into disparity space and comparing the reconstructed disparity values along the segment with the corresponding C-space disparity values to detect collisions.



(a) (b) (c)

Fig. 3. C-space expansion in disparity space: original left view image (a), stereo disparity map (b), C-space expanded disparity map (c).

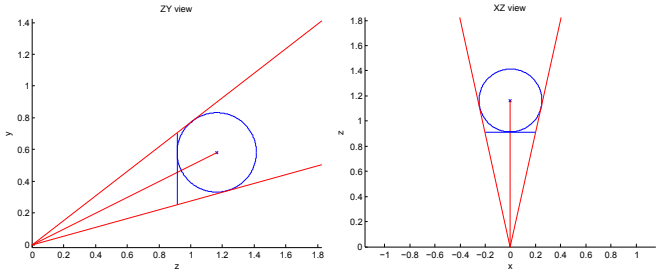
A. C-space Expansion

C-space expansion is implemented as an image processing function (Fig. 3). To illustrate this operation, we first project a pixel of the stereo disparity map $p(u, v, d)$ into world coordinates using the stereo base b_s and the focal length f (in pixels), assuming rectified images and a disparity map that corresponds to the left camera view:

$$z_w = -fb_s/d \quad (1)$$

$$P(x_w, y_w, z_w) = [uz_w/f, vz_w/f, z_w]^T \quad (2)$$

Considering an expansion sphere S around $P(x_w, y_w, z_w)$ with the expansion radius r_v, we calculate the position of the rectangle that perfectly hides the sphere from the viewpoint of the camera (Figure 4) and assign to it a disparity value that corresponds to the distance to the point on S that is closest to the camera origin.



(a) (b)

Fig. 4. C-space expansion example: The expansion square covers the expansion sphere from the camera view point (0,0) completely. It's disparity value is constant.

Technically, this expansion operation increases the expansion volume around a world point , since the correct projection of a sphere onto image coordinates is an ellipse, but this method has the advantage that the operation now is separable into two 1D operations – a horizontal expansion along image scan lines and a vertical expansion along image columns – reducing computational cost significantly.

The horizontal and vertical expansion limits depend on the viewing angle of each pixel and the expansion radius r_v. The horizontal viewing angle α to a point P in world coordinates is defined as

$$\alpha = \tan^{1}(z_w/x_w) \quad (3)$$

and the horizontal angular field of view γ of the expansion sphere S around P is defined by the distance of P from the camera origin and the expansion radius r_v

$$\gamma = 2\alpha_1 = 2\sin^{1}(r_v/\sqrt{z_w^2 + x_w^2}) \quad (4)$$

Projecting the two rays r_1 and r_2 along the viewing angles $\alpha+\alpha_1$ and $\alpha-\alpha_1$ back into the image defines the horizontal extension of the expansion sphere in the image

$$r_1 = [r_{1_x}, r_{1_y}, r_{1_z}] = [z_w/\tan(\alpha + \alpha_1), y_w, z_w] \quad u_1 = fr_{1_x}/r_{1_z} \quad (5)$$

$$r_2 = [r_{2_x}, r_{2_y}, r_{2_z}] = [z_w/\tan(\alpha - \alpha_1), y_w, z_w] \quad u_2 = fr_{2_x}/r_{2_z} \quad (6)$$

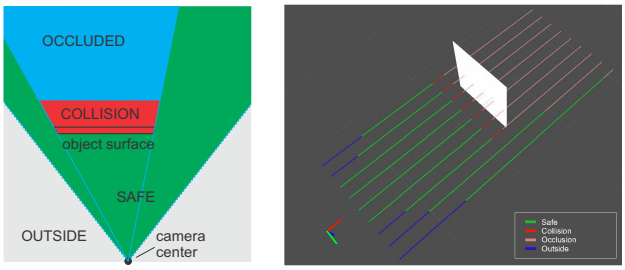
The vertical extension can be calculated similarly:

$$\beta = \tan^{1}(z_w/y_w) \quad (7)$$

$$\beta_1 = \sin^{1}(r_v/\sqrt{z_w^2 + y_w^2}) \quad (8)$$

$$r_3 = [r_{3_x}, r_{3_y}, r_{3_z}] = [x_w, z_w/tan(\beta + \beta_1), z_w] \quad v_1 = fr_{3_y}/r_{3_z} \quad (9)$$

$$r_4 = [r_{4_x}, r_{4_y}, r_{4_z}) = (x_w, z_w/tan(\beta - \beta_1), z_w] \quad v_2 = fr_{4_y}/r_{4_z} \quad (10)$$



(a) (b)

Fig. 5. Collision checking logic: (a) Trajectory segments inside the viewing volume of the camera are classified as: SAFE if unobstructed (green), COLLISION if directly behind an object surface (red), and OCCLUDED if behind on object; (b) Test trajectories in a simulation: Collision checking result for a set of 9 horizontal trajectories .

To determine the new disparity of the rectangular defined by the image coordinates u_1, u_2, v_1 and v_2 the expansion radius r_v is subtracted from the z-component of P and transformed into a disparity value:

$$z_{\text{new}} = z_w - r_v \tag{11}$$

$$d_{\text{new}} = -fb_s/z_{\text{new}} \tag{12}$$

To calculate the full C-space map, (3) to (12) are applied to every pixel in the stereo disparity image. Each (u, v, d) triplet defines a rectangular image region (u_1, v_1, u_2, v_2) with a constant disparity d_{new}, that is written into an output map. Individual pixels are only updated if the new disparity is larger than the previous disparity value that was generated by a different (u, v, d) triplet.

1) Implementation Aspects: Equations (3)-(12) can be pre-calculated over the disparity space volume. Since the calculation of u_1 and u_2 in (3)-(6) is independent of the y coordinate it suffice to pre-calculate a look-up table to store the values of u_1 and u_2 for each defined (x,d) combination. Similarly, a look-up table for v_1 and v_2 is calculated for each (y,d) combination, and finally, the values for d_{new} can be pre-calculated for all valid disparities. Because of this separability, the C-space expansion can be implemented very efficiently. In a first step, an intermediate disparity image is generated by horizontally expanding all disparity values from the stereo disparity map using the u_1/u_2 look-up table. In a second step, each pixel in the intermediate disparity image is expanded vertically using the v_1/v_2 look-up table and the expansion column is stored with a new disparity value from the d_{new} look-up table in the final C-space disparity map.

B. Collision Checking in Disparity Space

The queries from the motion planner come as a sequence of short linear 3D trajectory segments. The collision checker module takes each segment that is defined through its start and end point, projects it into the current C-space disparity map D_{cspace}, and checks all pixels that are located on the straight line between the projected start and end point for collision. Collision checking itself depends on the reconstructed disparity value $d(p_s)$ of a point p_s on the segment and the actual disparity of the underlying pixel p_{cm} in the C-space map. If the disparity of p_s is larger than the disparity of p_{cm}, the pixel is classified as SAFE. If the disparity of p_s is smaller than p_{cm} the point on the trajectory is located behind an obstacle, and it is classified depending on the disparity difference

$$\begin{aligned} d(p_s) > d(p_{cm}) &: \text{SAFE} \\ d(p_s) < d(p_{cm}) \wedge d(p_s) - d(p_{cm}) < k &: \text{COLLISION} \\ d(p_s) < d(p_{cm}) \wedge d(p_s) - d(p_{cm}) \geq k &: \text{OCCLUDED} \\ p_s \notin D_{cspace} &: \text{OUTSIDE} \\ d(p_{cm}) = invalid &: \text{NO\_DATA} \end{aligned} \tag{13}$$

If the difference is smaller than a threshold, a collision occurred. If it is larger, the trajectory point is labeled as OCCLUDED as shown in Figure 5.

If the C-space map contains no valid disparity data at a checked pixel location p_{cm}, the trajectory segment is labeled as NO_DATA – which can be caused e.g. by a non textured surface when applying a real-time stereo approach.

How the planner uses these different trajectory classifications is explained in the following section.

V. Motion Planning over Closed-Loop Dynamics

Motion planning of the quadrotor has several challenges. First, it has a high dimensional state space – 6 DOF position and orientation, and their time derivatives position/orientation and velocity/angular velocity, resulting in at least 12 states. Second, the system is agile and consequently has poor stability, and naively propagating the open-loop vehicle dynamics quickly results in undesirable trajectories. Third, the dynamics become highly nonlinear especially when performing aggressive maneuvers. The focus and novelty of this work is in the collision checking in disparity space as explained above. For completeness, in the following we sketch our planning framework which is used to perform the real-world experiments described in Section VI.

We use a similar approach as in [18] but extend it to 6 DOF motion with agile vehicle dynamics. The planning is done over closed-loop dynamics, where a stabilizing low-level controller wraps the open-loop system. The planner uses the model described in [19] and designs a series of waypoint inputs to the low-level controller and forward-simulates the closed-loop dynamics. This effectively reduces the planning space to only 3D (i.e. waypoints).

The low-level controller consists of two layers: a linear *feedback controller* and a *waypoint tracker*. The *waypoint tracker* keeps track of which waypoint to visit next, and when to switch to the next waypoint segment (Fig. 6). It also computes the reference position/velocity and the position/velocity tracking errors. Given these tracking errors, the *feedback controller* computes the vehicle inputs $u_{\text{collective}}$, u_{roll}, u_{pitch}, and u_{yaw} to track the reference. For these four independent control loops we use regular PID controllers.

A. Incremental RRT-based Planning with Safety Invariance

The vehicle model and the low-level controller described above are incorporated as a simulator in the RRT planner,

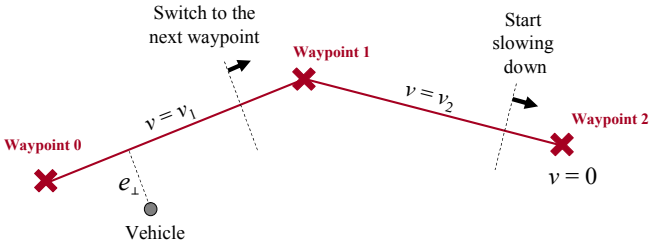


Fig. 6. Waypoint tracking logic. Waypoints are marked with $\times$. The cross-track controller minimizes the cross-track tracking error $e_\perp$, while the along-track controller maintains a commanded velocity along the lines that connect waypoints. The tracker switches waypoints when the along-track distance to the next waypoint becomes smaller than a threshold.

which is called CL-RRT (Closed-loop RRT) [18]. Below, we summarize the main steps of this RRT algorithm.

1) Randomly sample the 3D world space.
2) Select a node in the tree.
3) Form a controller input from the selected node to the sample.
4) Forward simulate the closed-loop dynamics from the selected node, using the controller input generated in step 3, and obtain a dynamically feasible trajectory.
5) Check if the predicted trajectory collides with or is occluded by obstacles (see section IV). If there is a collision, or the occlusion is in a close range, discard the sample and go to step 1.
6) Add a sample and associated propagated trajectory to the tree. Also generate intermediate nodes on the trajectory, so that it can branch off to another trajectory for future samples.

Note, that this RRT simultaneously grows a tree of controller inputs (straight lines connecting the controller input of a selected node to a sample, which forms an input to the forward simulation) and a tree of collision-free dynamically feasible trajectories (output of the forward simulation). Figure 7 illustrates the tree generated with this algorithm.

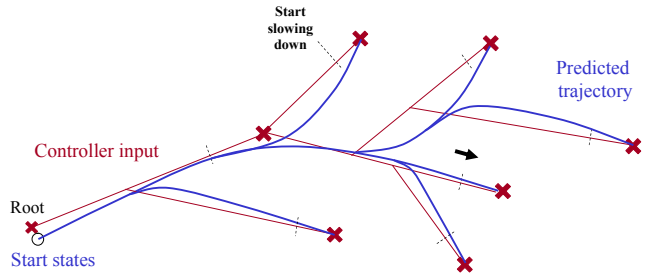


Fig. 7. RRT with closed-loop dynamics. The red lines represent the input to the controller, which are constructed by connecting a sample (marked with $\times$) to the tree. The blue lines represent the predicted vehicle trajectory, which is computed by the forward simulation.

Another unique feature of this RRT planner is that it maintains a safety invariance while the vehicle is flying [20]. In this paper, we say the vehicle is safe when the vehicle can come to a hover from the current state without collision. Note, that hovering is an invariant state of the quadrotor – once the vehicle is in a hover state, it can theoretically remain in hover without collision indefinitely in a static environment.

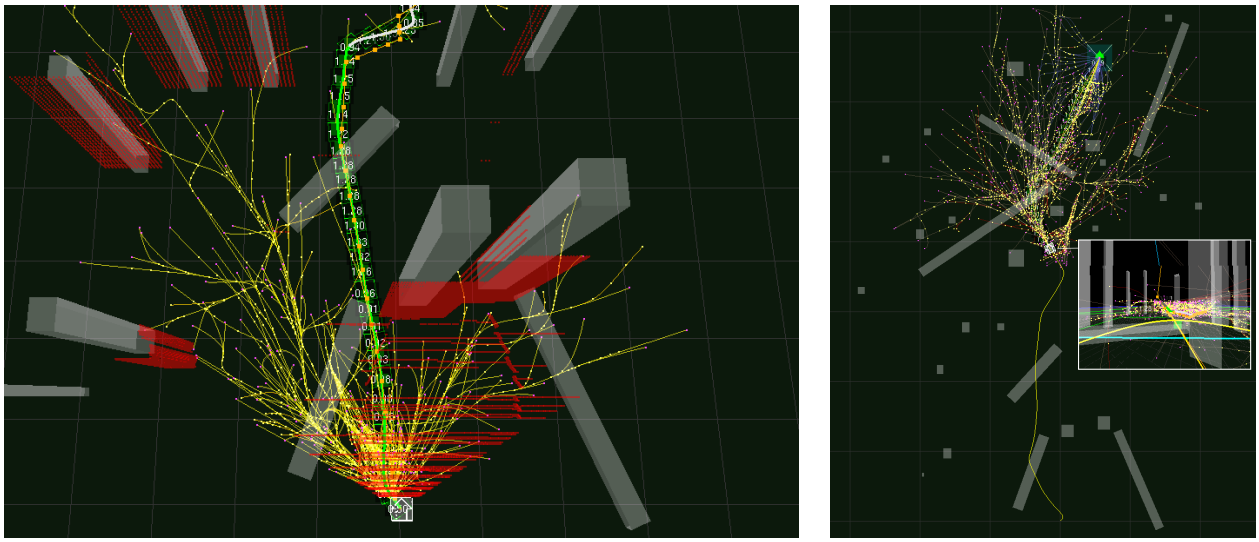
B. Motion Planner Execution Strategies

When the low-level controller executes the planned waypoints, different real-world considerations require specific execution strategies. First, paths planned a certain distance behind a perceived obstacle are considered as feasible. This allows to plan into potentially free space behind obstacles. The maintained safety invariance ensures an early re-planning should this space not be free. Space with no available data is treated similarly. Second, the predicted trajectory by the planner and the actual trajectory flown are generally close [21]. To account for non-zero prediction errors and changes in the environments, the planner re-propagates the latest states and changes the trajectory accordingly to ensure collision free flight from the current position. Last, if our proposed collision checker rejects a trajectory because it ends in occluded space or outside the field of view, we retain the feasible portion of such a trajectory in the tree, so that RRT can quickly connect future samples and grow trees from it.

VI. Experimental Results

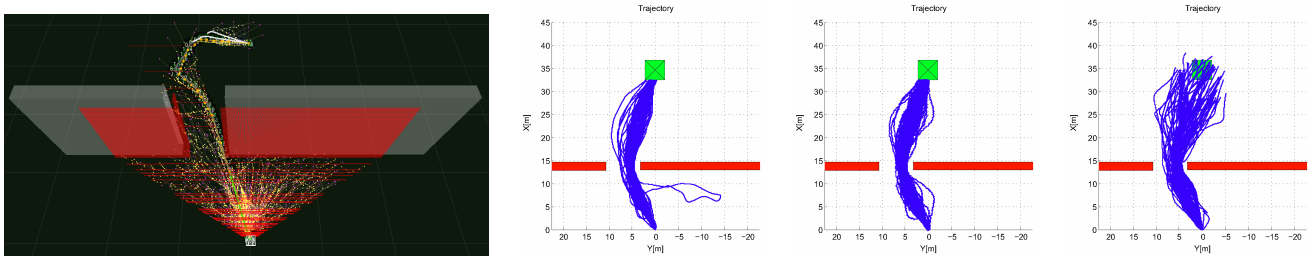
To evaluate our approach, we implemented our navigation algorithm both in a simulation environment and on a real quadrotor system. The simulation environment mimics a quadrotor within a virtual 3D world that is composed of cuboids. At the beginning of a simulated flight, the quadrotor is placed at a starting position above the ground and the planner is executed a few seconds ahead of the controller to allow the construction of an initial tree of decent size. Figure 8 gives an example of such a step, where the stationary vehicle has planned trajectories towards the goal at the top of the scene. When the controller is started, the vehicle begins to execute the planned trajectory, re-planning simultaneously during flight. As the position of the vehicle changes, new parts of the scene come into view, and trajectories are updated accordingly, until the goal is reached.

To verify the robustness of our approach, we repeatedly executed a simulated flight for several example scenes. One standard scenario for performance evaluation of planning stability is the flight through a vertical opening in a wall (Fig. 9), which we use to measure the influence of a corrupted pose on the planning approach. In a Monte Carlo simulation, we commanded the vehicle to pass the vertical opening with noise added to the pose estimate and recorded the flown flight paths for 100 flights for each experiment. With no pose noise added, the vehicle was able to fly safely through the opening on each run (Fig. 9b). Figure 9c illustrates the same flight experiment, where limited white noise ($\eta \in [-15\text{cm}, 15\text{cm}]$) was added to the position of the collision sensor (stereo camera system) prior to each planning cycle. When treating the pose of the collision sensor (with noise overlay) as true vehicle pose, all node positions on the RRT-tree become erroneous, simulating the effect of a noisy pose estimate on the planning process. In this case, collision checking invalidates additional tree branches, due



(a) (b)

Fig. 8. Simulated flight through virtual forest: (a) Initial flight trajectory from the vehicle position at the bottom towards the goal at the top with overlaid 3D C-space point positions (red); the horizontal red lines in front of the vehicle correspond to C-space points above the ground; (b) top down view during traverse with overlaid current view of the vehicle.



(a) (b) (c) (d)

Fig. 9. Flying through a door opening with added pose noise: Scene view from the starting position with overlaid c-space point cloud (red) (a), top down view of Monte Carlo simulation with 100 runs with (b) no pose noise, (c) additive white noise in x and y ($\leq$ 15cm), (d) random walk bias in x and y (13.4cm/s, random direction for each run).

to the added noise, forcing the vehicle to re-plan a valid trajectory when needed. As shown in Figure 9c, additive noise only affects the planning result marginally. The vehicle was able to execute all runs properly and pass the opening at all times, since the added noise was zero mean (no drift). This changed when a drifting pose estimate was simulated as shown in Figure 9d. To simulate a random walk bias on the position estimates, a fixed position offset of 1.3cm (a 10% drift when operating at maximum speed of 1.3m/s with a 100Hz simulation rate) in a random direction within the x/y plane was defined at the beginning of each run, and added as a pose offset prior to each planning step. In this experiment, the vehicle kept a stable orientation to maintain correct heading, which we assume closely related to real flight experiments, since roll and pitch would be stabilized around a drift free gravity vector and yaw can be assumed to be measured locally drift free by an on-board magnetometer or a vision-aided pose estimation approach. Since all world point coordinates drift with the amount of bias on pose, the goal cannot be reached and serves as a desired flight direction indicator.

The vehicle was able to pass the obstacle without a collision in 96% of all cases. Collisions only occurred when the vehicle was already within the opening and drifting sideways, so that the camera could not see the closing in obstruction.

Figure 10 illustrates the performance of our navigation system in a real world scenario. We implemented our algorithm on-board an Asctec Pelican quadrotor which was equipped with an Intel Core2Duo, 1.86 GHz processor and conducted flight experiments in a test forest (Fig. 1). Our system setup used the sensor fusion approach from [22] for pose estimation. It fuses IMU and position updates from a visual SLAM algorithm (Parallel Tracking and Mapping (PTAM)[23]) that uses images from a downward looking camera (752x480, grayscale, 110 degree FOV). To generate stereo disparity maps, we mounted a stereo camera system on top of the quadrotor that included an OMAP3730 to calculate real-time disparity maps (320x240, 25Hz, 12cm baseline, 110 degree FOV) [24] to off-load stereo calculations from the main processor, which only performed post-processing of disparity maps within the stereo vision pipeline. In this

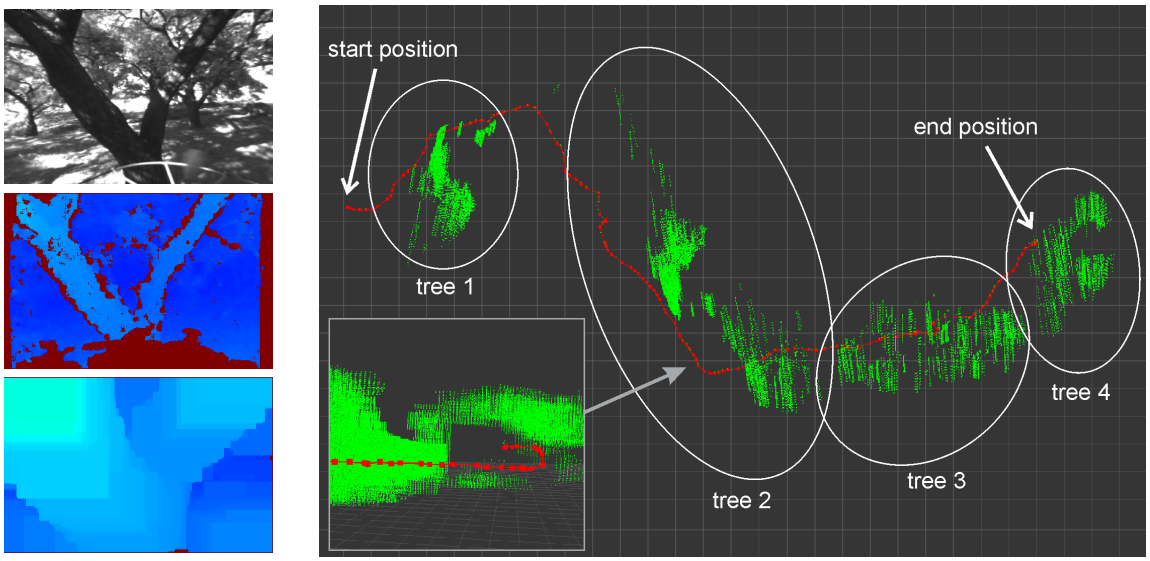


Fig. 10. Flying under tree canopy: (a) top to bottom: left rectified stereo camera view; disparity map; C-space map with expansion size of 60cm; (b) top down view of flown trajectory (red) with overlaid accumulated C-space point clouds (green) that are generated from sub-sampled C-space disparity maps; only points are shown that are more than 2.5m above the ground and not farther away from the vehicle than 2m. The inset shows a view from behind the vehicle in an upright view, illustrating flight bellow a tree branch.

setup, the pose estimation framework used 59% of the total resource (camera and PTAM 50%, pose filter 9%), stereo post-processing used 9%, and motion planning used 88%, generating motion plans at 2Hz. In total, the navigation frame work used 160% of the available 200% processing power of our 2 core CPU. Figure 10a gives an example of the viewing volume that is used for motion planning when approaching a tree. Figure 10b illustrates the flight trajectory of a representative experiment in a top down view. The vehicle started at the left of the image and was commanded to fly to the right, avoiding four different trees on its traverse. For the flight experiments, we started generating motion plans at an altitude of 4m and restricted the planning volume sample 3D points between 2m and 5m altitude. The vehicle was able to avoid obstacles autonomously and plan its motion around tree branches. This included a portion of the trajectory where the vehicle flew correctly below an overhead branch (tree 2, Fig. 10b).

VII. Conclusions and Future Work

We are developing an autonomous navigation system for outdoor operation of micro air vehicles. Given the highly textured nature of most outdoor scenes, we choose cameras for scene perception because they provide wide field of view with long-term packagability in very small size, weight, and power. In this paper, we instantiated the system with a forward-looking stereo camera pair for obstacle detection and a downward-looking camera to contribute to state estimation. To provide obstacle detection with low memory and computation requirements, we use an inverse range representation of the world in image space (scaled stereo disparity). Collision checking of trajectories generated by a CL-RRT planner is done very efficiently by performing a C-space-like expansion of the disparity map, projecting candidate trajectories into image space, and performing a z-buffer-like visibility test.

This approach was effective in simulations of flight through a virtual forest and through a narrow opening in a virtual wall; in simulation it was also robust to perturbations in vehicle state. We also implemented the approach on an Asctec Pelican quadrotor and tested it successfully in flight through a grove of trees. Some non-optimality of trajectories was seen due to short-term compromises that constrained trajectory search to the visible field of view and limited yaw excursions to maintain pointing toward the goal. Nevertheless, the results bear out the potential of the approach. The visual SLAM system experienced difficulty with feature tracking when flying within a couple of meters of the ground, due to dynamic range issues from the mixed sunlit and shaded terrain; although state estimation was successful at altitudes above 2m, this points to a need for greater robustness in this part of the system.

We plan to extend the perception system to include peripheral optical flow to both sides and to extend the representation to the surface of a virtual cylinder wrapping around the vehicle. This will facilitate fusion of inverse range data from stereo and optical flow, persistent memory of objects that pass out of view to the side of the aircraft, and temporal fusion by warping and overlaying inverse range maps with new measurements from frame to frame. We will also explore alternatives to the current CL-RRT motion planner. Longer term work will involve hybrid representations with other polar and Cartesian voxel data structures to support broader variation in clutter and flight speed.

ACKNOWLEDGMENTS

This work was funded by the Army Research Laboratory under the Micro Autonomous Systems & Technology Collaborative Technology Alliance program (MAST-CTA). JPL contributions were carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

REFERENCES

[1] S. Shen, N. Michael, and V. Kumar, "3d indoor exploration with a computationally constrained mav," in *Robotics: Science and Systems*, 2011.

[2] A. Bachrach, S. Prentice, R. He, P. Henry, A. S. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Estimation, planning and mapping for autonomous flight using an RGB-D camera in GPS-denied environments," *Int. J. Robotics Research*, vol. 31, no. 11, pp. 1320 – 1343, 2012.

[3] A. Beyeler, J.-C. Zufferey, and D. Floreano, "Vision-based control of near-obstacle flight," *Auton. Robots*, vol. 27, no. 3, pp. 201–219, Oct. 2009.

[4] J. Conroy, G. Gremillion, B. Ranganathan, and J. Humbert, "Implementation of wide-field integration of optic flow for autonomous quadrotor navigation," *Auton. Robots*, vol. 27, no. 3, pp. 189–198, 2009.

[5] A. Hyslop and J. Humbert, "Autonomous navigation in three-dimensional urban environments using wide-field integration of optic flow," *Guidance, Control, and Dynamics*, vol. 33, no. 1, 2010.

[6] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *ICRA*, 2013, pp. 1757 – 1764.

[7] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor MAV," in *IROS*, 2012, pp. 4557 – 4564.

[8] E. Bakolas and P. Tsiotras, "Multiresolution path planning via sector decompositions compatible to on-board sensor data," in *AIAA Guidance, Navigation, and Control Conference*, 2008.

[9] H. Yu and R. Beard, "A vision-based collision avoidance technique for miniature air vehicles using local-level frame mapping and path planning," *Auton. Robots*, vol. 34, no. 1-2, pp. 93 – 109, 2013.

[10] M. Bajracharya, A. Howard, L. Matthies, B. Tang, and M. Turmon, "Autonomous off-road navigation with end-to-end learning for the LAGR program," *Field Robotics*, vol. 26, no. 1, 2009.

[11] K. Sarabandi, M. Vahidpour, M. Moallem, and J. East, "Compact beam scanning 240 GHz radar for navigation and collision avoidance," in *SPIE vol. 8031*, 2011.

[12] S. Hrabar, G. Sukhatme, P. Corke, K. Usher, and J. Roberts, "Combined optic-flow and stereo-based navigation of urban canyons for a uav," in *IROS*, 2005.

[13] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for quadrotor flight," in *RSS Workshop on Resource-Efficient Integration of Perception, Control and Navigation*, 2013.

[14] M. Otte, S. Richardson, J. Mulligan, and G. Grudic, "Path planning in image space for autonomous robot navigation in unstructured outdoor environments," *Field Robotics*, vol. 26, no. 2, 2009.

[15] B. MacAllister, J. Butzke, A. Kushleyev, H. Pandey, and M. Likhachev, "Path planning for non-circular micro aerial vehicles in constrained environments," in *ICRA*, 2013, pp. 3918 – 3925.

[16] M. Pivtoraiko, D. Mellinger, and V. Kumar, "Incremental micro-UAV motion replanning for exploring unknown environments," in *ICRA*, 2013.

[17] G. R. Martin, "What is binocular vision for? A bird's eye view," *Journal of Vision*, vol. 9, no. 11, pp. 1 – 19, 2009.

[18] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. How, "Real-Time Motion Planning With Applications to Autonomous Urban Driving," *Trans. on Contr. Sys. Tech*, vol. 17, no. 5, pp. 1105–1118, 2009.

[19] J. How, B. Bethke, A. Frank, D. Dale, and J. Vian, "Real-Time Indoor Autonomous Vehicle Test Environment," *IEEE Control Systems Magazine*, vol. 28, no. 2, pp. 51–64, April 2008.

[20] T. Schouwenaars, B. D. Moor, E. Feron, and J. How, "Mixed Integer Programming for Multi-Vehicle Path Planning," in *Proc. Europ. Contr. Conf.*, Porto, Portugal, September 2001.

[21] B. Luders, S. Karaman, E. Frazzoli, and J. How, "Bounds on Tracking Error using Closed-Loop Rapidly-Exploring Random Trees," in *American Control Conference*, Baltimore, MD, June 2010, pp. 5406–5412.

[22] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart, "Monocular vision for long-term micro aerial vehicle state estimation: a compendium," *Field Robotics*, vol. 30, no. 5, pp. 803 – 831, 2013.

[23] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ser. ISMAR '07. IEEE Computer Society, 2007, p. 110.

[24] S. Goldberg and L. Matthies, "Stereo and IMU assisted visual odometry on an OMAP3530 for small robots," in *2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2011, pp. 169 –176.