Flying Fast and Low Among Obstacles

Sebastian Scherer, Sanjiv Singh, Lyle Chamberlain and Srikanth Saripalli

***Abstract*— Safe autonomous flight is essential for widespread acceptance of aircraft that must fly close to the ground. We have developed a method of collision avoidance that can be used in three dimensions in much the same way as autonomous ground vehicles that navigate over unexplored terrain. Safe navigation is accomplished by a combination of online environmental sensing, path planning and collision avoidance. Here we report results with an autonomous helicopter that operates at low elevations in uncharted environments some of which are densely populated with obstacles such as buildings, trees and wires. We have recently completed over 1000 successful runs in which the helicopter traveled between coarsely specified waypoints separated by hundreds of meters, at speeds up to 10 meters/sec at elevations of 5-10 meters above ground level. The helicopter safely avoids large objects like buildings and trees but also wires as thin as 6 mm. We believe this represents the first time an air vehicle has traveled this fast so close to obstacles. Here we focus on the collision avoidance method that learns to avoid obstacles by observing the performance of a human operator.**

I. INTRODUCTION

Today the threat of low-altitude obstacles constrain fielded unmanned aerial vehicles (UAVs) to operate at high altitude, or under close human supervision at low altitude. This is because even a low-speed collision with the environment can be fatal. Safe autonomous flight is essential for widespread acceptance of aircraft that must fly close to the ground and such capability is widely sought. For example, search and rescue operations in the setting of a natural disaster allow different vantage points at low altitude. Likewise, UAVs performing reconnaissance for the police, news or the military must fly low enough that the environment presents obstacles.

Operationally, we would like a system that can safely fly between coarsely specified waypoints without having to know beforehand that the flight path is clear or that a waypoint is even achievable. Flying close to and among obstacles is difficult because of the challenges in sensing small obstacles, and in controlling a complex system to avoid obstacles in three dimensions. Some aspects of collision avoidance are easier for air vehicles than ground vehicles. Any object close to the intended path of an air vehicle must be avoided as opposed to ground vehicles where deviations from the nominal ground plane indicate obstacles and are often not visible until they are close. The use of helicopters, rather than fixed wing aircraft also helps because in the worst case it is possible to come to a hover in front of an obstacle.

Sebastian Scherer, Sanjiv Singh and Lyle Chamberlain are with the Robotics Institute, Carnegie Mellon University. Srikanth Saripalli is with the CRES, University of Southern California.

email: [basti, ssingh, lylecham]@ri.cmu.edu, srik@robotics.usc.edu

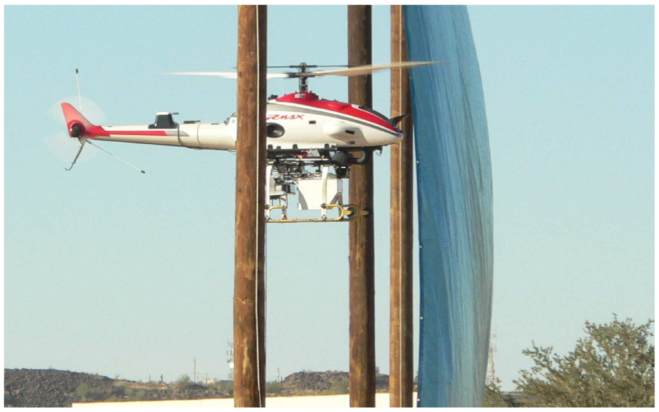

Fig. 1. The helicopter platform we used to test collision avoidance is flying autonomously in between two poles 10 m apart. The rotor span of this helicopter is 3.4 m

Still, the availability of appropriate sensors, logistical issues of mounting a vehicle with sufficient sensing, computing and communication gear, and the risk involved in such experimentation has kept researchers from significant progress in this area. While some methods of obstacle avoidance on aircraft have been implemented, none to our knowledge has achieved the speed and endurance that we present here.

In order to operate in real time, we have developed a layered approach, similar in a general way, to the approach used by autonomous ground vehicles that operate in uncharted terrain: *plan globally and react locally.* Our approach combines a slower path planning system with a high-frequency reactive obstacle avoidance algorithm. The planner updates a global path to the goal every few seconds, while the reactive algorithm uses intermediate points along the path as its short term goal to quickly avoid unforeseen obstacles. The planner is based on Laplace's equation, while the reactive algorithm is an adaptation of a model of obstacle avoidance by humans [1].

The system uses a novel scanning ladar that can detect small (cm sized) obstacles at ranges of 100 m. Even poorly reflective wires with sub-cm diameter can be detected from 75 m. All processing and control is done onboard. This system has been implemented on a Yamaha RMax helicopter shown in Fig. 1.

Below, we discuss related work in section II and sensor processing in section III. Section IV explains our architecture and algorithm. In section V we convey the setup of the flight hardware. Section VI presents results from field tests.

II. RELATED WORK

Current UAV obstacle avoidance techniques can be categorized into two broad methods: Planning and Reactive. Planning paradigms use a world map and plan a trajectory for the vehicle to follow. Such approaches must find trajectories that not only geometrically avoid obstacles but also ensure that the dynamics of the vehicle are capable of following the paths. This can be prohibitively expensive to compute if the trajectory has to be continuously revised. In contrast, reactive methods overcome the real-time problem by using a simple formula to *react* to obstacles as they appear, but they cannot guarantee an appropriate solution to every possible situation.

Vision-based reactive methods have been popular because payload weight is a serious limitation for UAVs. For example, Merrel et. al. [2] proposed to use optical flow in order to compute how to avoid obstacles. The micro flyer developed by Zufferey and Floreano in [3] is expected to reactively avoid obstacles using very small 1-D cameras. Zapata and Lepinay have proposed a reactive algorithm[4] similar to ours in motivation, but we are aware only of simulated results [5]. Hrabar and Sukhatme use vision in order to navigate in canyon-like environments using optical flow[6]. This method implicitly avoids the walls of the canyons by centering the vehicle such that optical flow from both sides of the canyon is equal. Byrne et al have demonstrated obstacle detection from a wide-baseline stereo system that uses color segmentation to group parts of the scene (even if lacking in optical texture) into spatially contiguous regions for which it is possible to assign range [7].

Larger helicopters such as our Yamaha RMax helicopter on the other hand can afford more resources to explore alternative plans. One approach by Vandapel et. al. implements a planning algorithm in [8] that uses point clouds in order to determine a path tunnel network. While this work used real data taken from a helicopter, the path produced was only visualized and not executed. Kitamura et. al. [9] use an octree representation of the world to plan paths using a 3-D A* implementation[10]. Frazolli et. al. [11] on the other hand use a concatenation of motion primitives in order to construct valid paths that avoid obstacles and achieve a desired goal pose. On our helicopter we use a Laplacian path planning algorithm [12] that is similar to the approach used in [13] and [14].

Shim and Sastry have proposed an obstacle avoidance scheme that uses nonlinear model predictive control. Their system builds controllers for arbitrarily generated trajectories. They have demonstrated results on a RMax platform operating in a plane using a single-axis laser rangefinder at speeds of 2m/s [15]. Our collision avoidance system combines reactive and planning algorithms and therefore enables travel at speeds limited only by the furthest distance at which the smallest obstacle can be avoided. At the same time, we know that our system can plan complicated paths to arrive at a difficult goal without risking a collision.

III. PERCEPTION

Our experience with passive vision outdoors is that natural lighting far exceeds the dynamic range of commercially available imagers. Deep shadows can provide sharper edges than occlusion boundaries of real objects and small objects can be visually obliterated under very low or very bright

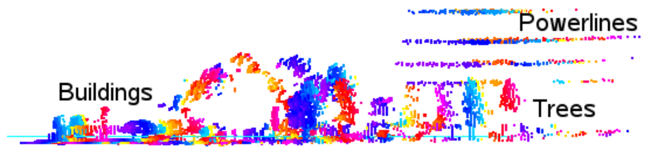

Fig. 2. A scene generated from a series of static scans of the environment with buildings, trees and powerlines. The powerlines are visible at 170m.

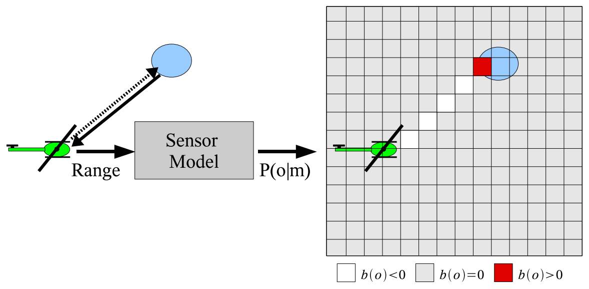

Fig. 3. A laser rangefinder returns a signal indicating a hit. The signal is mapped to a probability $P(o|m)$. This probability is used to update the relevant cells in the evidence grid.

lighting. Since we want to be able to detect even very thin and poorly reflective objects such as powerlines and telephone wires, the only modality with sufficient sensitivity and small footprint is ladar. The downside of using ladar is that it requires mechanical scanning and hence a significantly larger package.

A. Sensor

We used a custom 3-D laser scanner from Fibertek Inc. with a 30x40 degree field of view to sense the environment. The scanner sweeps an oval pattern created by two continuously spinning lenses. Originally designed to operate as an operator aid for human pilots, this pattern is particularly suited to the detection of wires in many orientations as shown in Fig. 2. This sensor facilitates collision avoidance because it is sensitive enough to detect obstacles at distances much greater than the stopping distance of the helicopter at the maximum speed. For instance it can detect typical powerlines at 100-150 m while the stopping distance is 40 m at a speed of 10 m/s.

B. Evidence Grids

A three dimensional *evidence grid*[16] represents the world because it is able to express the belief of a volume of space being occupied. The grid is regular with each cube storing the log-likelihood ratio of the voxel being non-empty. If the log-likelihood ratio is larger than zero a cell is considered occupied.

This data structure provides a way to accumulate more information about obstacles from a sequence of instantaneous ladar scans. For example a false positive range measurement is erased if enough rays penetrate the cell containing a false positive hit. Dust particles and rain drops can cause a false obstacle to appear. Although the robot might react to such false positives the false evidence will be erased if enough evidence of empty volume is accumulated. However the same is true for small real obstacles.

The beam of the ladar has a beam diversion at the maximum range that is smaller than the evidence grid cell

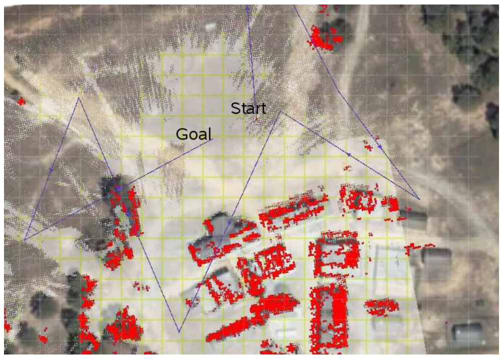

Fig. 4. A 2D horizontal slice of an evidence grid from the McKenna MOUT site at Ft. Benning, GA (USA). This model was created by flying part of the shown path. Red(Dark) represents occupancy and white is evidence of free space. An aerial image is overlayed on top of the evidence grid in order to facilitate the interpretation and show the accuracy of the constructed model. The grid spacing is 10 m.

size.Therefore processing of a new ladar hit is linear $O(n)$ in the number of cells that are affected by the ray from the current location to the hit location. Since the work per cell is only n, processing is fast.

The raw range sensor data is transformed to an occupancy probability and the result is mapped into an evidence grid using position and orientation information. Fig. 3 summarizes the flow of data. The belief of a grid cell being occupied is $b(o)$ which represents the log-likelihood ratio of the cell being occupied vs. free. We assume that the prior probability of a cell being occupied is unknown $P(o) = 0.5 \rightarrow b_{initial}(o) = 0$.

The sensor model maps the signal returned from the laser rangefinder to a probability of occupancy (o) given the measurement (m): $P(o|m)$. Each cell in the grid on the line between a hit and the location of the robot is updated using a update rule. The belief of the cells $b(o)$ of the evidence grid is then updated by

$$\overline{b}(o) = \overline{b}(o) + \ln P(o|m) - \ln(1 - P(o|m)) \tag{1}$$

Since the accuracy of the ladar is independent of range we have a range-invariant sensor model. Furthermore a reported hit is extremely likely to be from an obstacle. Accordingly, for valid returns, we map the probability to a simple model in which

$$\ln P(o|m) - \ln(1 - P(o|m)) = 127 \tag{2}$$

and

$$\ln P(o|\overline{m}) - \ln(1 - P(o|\overline{m})) = -1 \tag{3}$$

We determined these values based on experiments with the sensor and chose the values with the qualitatively best evidence grid.

The evidence grid algorithm assumes a static scene. It will however incorporate changes in the environment if it sees enough evidence of empty space or occupancy. Accordingly our algorithms will avoid moving obstacles suboptimally since moving obstacles will create a smeared representation in the evidence grid and no explicit tracking is performed.

A 2D slice of an evidence grid from McKenna MOUT at Ft. Benning, GA (USA) is shown in Fig. 4. This model was created from data collected during the displayed flight path and is overlayed with an aerial view of the site.

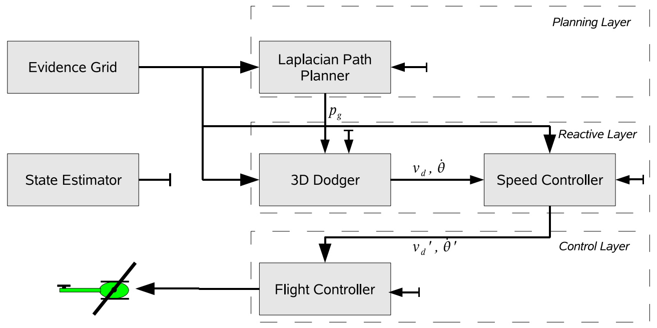

Fig. 5. Overall architecture of the algorithms. The higher the layer the lower the frequency of execution. A path p_g is produced in the planning layer and transmitted to the reactive layer. The reactive layer produces commands $(v_d', \dot{\theta}')$ that are then executed by the flight controller.

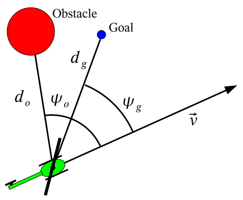

Fig. 6. A diagram illustrating the terms used in the control law. The coordinates used are expressed in a robot centric reference frame.

IV. OBSTACLE AVOIDANCE

Our layered obstacle avoidance architecture (Fig. 5) uses a combination of a deliberative path planning algorithm with a reactive collision avoidance method to compute a command. The algorithms are layered and executed at different frequencies on one processor in order to be able to react quickly to obstacles while still finding a global path to the goal. At the lowest layer the speed controller slows and accelerates the vehicle based on the distance to the closest reachable obstacle and on the minimum turning radius. At the next level, 3D Dodger, a reactive steering-space algorithm produces steering commands in both horizontal and vertical axes to actively avoid obstacles. At the highest layer, a path planning algorithm generates a smooth path around obstacles to the next goal. Here we present the details of the reactive method only.

A. Reactive collision avoidance algorithm: 3D Dodger

Our formulation of the 3-D reactive obstacle avoidance algorithm is based on the original formulation by Fajen and Warren [1] used in their study of obstacle avoidance in human subjects.

The algorithm uses a single goal point which attracts the vehicle's heading. The command is calculated based on only angles and distances to obstacles and the goal point. The vehicle relative angles and distances used in the formulation are shown in Fig. 6.

A helicopter typically has four control inputs $\{v_{xd}, v_{yd}, v_{zd}, \dot{\theta}_d\}$. However we impose an artificial non-holonomic constraint for lateral velocities $v_{yd} = 0$, chiefly because we want the laser scanner to point in the direction of travel. Since the magnitude of the velocity

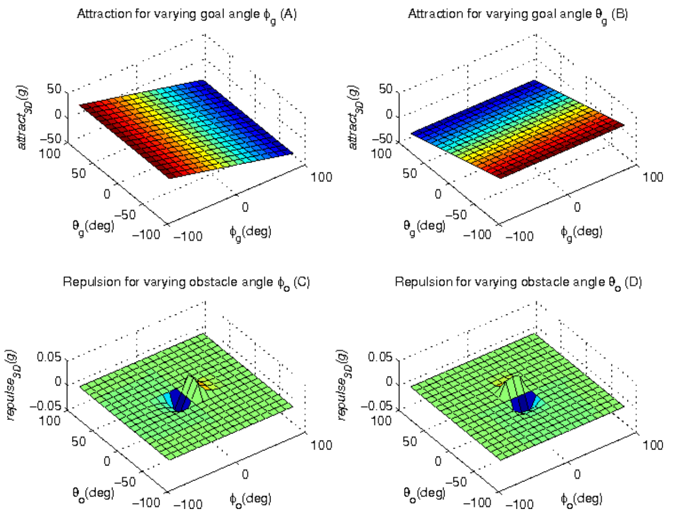

Fig. 7. The attraction and repulsion surface for our modified control law. Plot A and B show the independence of the goal attraction on each axis. Since a sigmoid shapes the repulsion for the repulsion function in plot C and D is not uniform.

vector is determined by the speed controller we are left with 2 degrees of freedom: A heading rate $\dot{\theta}$ and a vertical velocity component v_{zd}. The goal and obstacles are represented in spherical coordinates $[r, \theta, \phi]$. Accordingly we also determine a vertical and horizontal turning rate $\left[\dot{\theta}, \dot{\phi}\right]$ to command the vehicle.

The goal has two angles θ_g, ϕ_g and a distance to the goal d_g. The attraction to the goal increases proportionally with angle as shown in Fig. 7A-B and decreases exponentially with distance. The vector of the two steering rates for goal attraction is therefore defined as

$$\overrightarrow{attract}(g) = \overrightarrow{k_g} \begin{bmatrix} \theta_g \\ \phi_g \end{bmatrix} (e^{-c_1 d_g} + c_2) \tag{4}$$

Obstacles are also expressed in spherical coordinates. The repulsion increases exponentially with decreasing angles and decreases exponentially with increasing distance as shown in Fig. 7C-D. Also larger angles from the other axis like ϕ for θ decrease the repulsion from obstacles. The repulsion function is

$$\overrightarrow{repulse}(o) = -\overrightarrow{k_o} \cdot$$
$$\begin{bmatrix} sign(\theta_o) \cdot sigmoid(s_1(1 - \frac{|\phi_o|}{s_2})) \\ sign(\phi_o) \cdot sigmoid(t_1(1 - \frac{|\theta_o|}{t_2})) \end{bmatrix} (e^{-c_3 d_o}) \begin{bmatrix} e^{-c_4|\theta_o|} \\ e^{-c_4|\phi_o|} \end{bmatrix} \tag{5}$$

where

$$sign(x) = \begin{cases} 1 \; if \; x > 0 \\ 0 \; if \; x = 0 \\ -1 \; if \; x < 0 \end{cases} \tag{6}$$

and

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{7}$$

The resulting steering rate command sent is

$$\overrightarrow{\dot{\phi}} = \overrightarrow{attract}(g) + \sum_{o \in O} \overrightarrow{repulse}(o) \tag{8}$$

because we assume a superposition principle holds.

Since the off-axis angles have less weight than the angles closer to the direction of travel the algorithm commits to going around or over obstacles after it has reacted sufficiently to an obstacle. For example, imagine the vehicle approaching a telephone pole head on. Initially both the horizontal and vertical Dodger controllers will respond to the pole, turning up and say, to the left. But as the pole moves more to the right, it falls out of the attention region of vertical Dodger defined by the sigmoid function and remains in the attention region of horizontal Dodger. The result is that the vehicle stops its vertical avoidance maneuver and commits to the horizontal avoidance. In another scenario, say the vehicle approaches a wide building. Again, both Dodgers initially react, but this time the building moves out of the attention region of horizontal Dodger first. The result is that the vehicle commits to the vertical maneuver and climbs over the building. Allowing both Dodgers to initially react and *compete* for control results in intrinsically choosing the reaction that most quickly avoids the obstacle, while keeping both options open initially.

B. Virtual range sensor

It is desirable to consider a minimal and relevant set of obstacles to avoid because we assume that a superposition principle holds and consequently sum all the obstacles. Obstacles not visible from the current location of the robot have no influence because it is not possible to collide with these obstacles. Therefore it is sufficient to consider obstacles that are in line-of-sight. Furthermore obstacles behind the current direction of travel do not matter for obstacle avoidance and can be ignored.

The set of obstacles $o_i \in O$ where

$$o_i = \begin{bmatrix} \theta_i \\ \phi_i \\ d_i \end{bmatrix} \tag{9}$$

determines the behavior in the presence of obstacles. The virtual range sensor gives a range for a grid-discretized latitude and longitude from a reprojected z-buffer, that is created by rendering the evidence grid. The advantage of this representation is that it considers the relative size of an obstacle to be more important than the absolute size. Large obstacles at a large distance have less influence than small obstacles close to the robot. Furthermore the number of obstacles is also limited by the discretization. The grid of ranges to obstacles has a field of view of 140 degrees in both axis and each grid cell represents the closest distance in the volume of a 2x2 degree pyramid.

The obstacles considered by our reactive algorithm are additionally limited by a box-shaped constraint. The yaw axis of the box is defined by the location of the robot and the current goal point. Since the box only extends 5 m below the lowest point it allows the robot to fly at a low altitude without being influenced by the ground as an obstacle.

Furthermore the box also reduces the amount of processing because only obstacles inside the box need to be considered for obstacle avoidance. The grid restricted by the box typically contains on the order of 30000 cells approximately 0.7% of the total number of cells in the evidence grid. The evidence grid without the box contains 256x256x64 cells and has a resolution of 1 m.

C. Determining the parameters

Our control law has a large number of parameters that need to be set in order to generate a desired behavior. Overall there are 12 constants

$$\bar{u} = (k_g, c_1, c_2, s_1, s_2, t_1, t_2, k_{o,1}, k_{o,2}, c_{3,1}, c_{3,2}, c_{4,2}) \quad (10)$$

Some of the parameters have an intuitive meaning and are defined by the problem domain but some of the parameters are tedious and difficult to set. The goal following parameters k_g, c_1 and c_2 were determined by hand since we had a desired performance for pure goal following and tuning this subset is intuitive. The values of s_1, s_2, t_1, t_2 are used to shape the number of obstacles considered and were therefore also fixed.

In order to learn the remaining parameters our pilot flies the helicopter and tries to follow a straight line between a start and goal point while avoiding a pole obstacle in one axis as shown in Fig. 8. Data about the goal point, the obstacles, and the flown path are recorded and used to determine the unknowns of the described control model. The input to the control model and human subject at any point in time is a goal point p_g and a set of obstacles O. The pilot flies the helicopter pretending he is seeing the obstacles only when the algorithm actually uses the the obstacle information.

Our training example is chosen to not require any sink or climb maneuver. This reduces the number of parameters that need to be learned, because only the horizontal commands determine the behavior:

$$u_e = (k_{o,1}, c_{3,1}, c_{4,1}) \quad (11)$$

Given a set u of parameters, we generate a path $P_t = \{q_i = (k_i, l_i) | i = 1..n\}$ with the same n number of points as the training path segment P_s, which contains regularly sampled points in the plane. $P_s = \{p_i = (x_i, y_i) | i = 1..n\}$.

The error between the two paths is defined as the Euclidean distance between each point pair : $d(\bar{u})_i = \sqrt{(k_i - x_i)^2 - (l_i - y_i)^2}$. Consequently, the total error minimized between two path segments is $D(\bar{u}) = \sum_{i=1}^{n} d(\bar{u})_i$. The optimization procedure minimizes the error term $min_{\bar{u}} D(\bar{u})$.

The path P_t is generated from a forward simulation of the commands sent to the robot. Since the length of the path and velocities are not controlled in this model, we use the recorded speeds to ensure P_t has the same length as the training path P_s. Since the space of parameters has many local minima we randomly choose sets of parameters uniformly distributed between 0 and 10.

The model of the helicopter used for training is not perfectly accurate and therefore it was necessary to fine tune the parameters on the actual helicopter to improve the behavior of the real system. We varied the value of

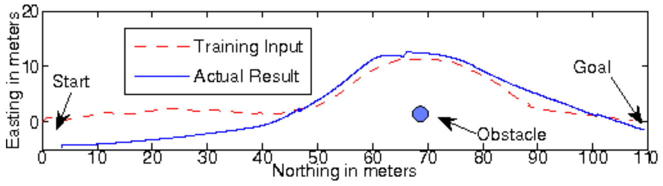

Fig. 8. This figure shows the path used for training as a dashed line and the actual path taken by the helicopter as a solid line. The path with the fine-tuned learned parameters still is a adequate match for our training path.

k_o systematically between $100\% - 150\%$ to fine tune the behavior on a set of test cases.

Figure 8 shows the actual path of the helicopter overlaid with the training input. The parameters used after fine-tuning still adequately matched the prediction.

D. Integrating the Reactive and Planning Layer

Since a helicopter can come to a complete hover, it can get around obstacles by moving horizontally or vertically. However, we prefer to smoothly change the direction of travel like a fixed wing aircraft, partly for energy reasons but also because such motion keeps the sensor looking forward. Hence, in the ideal case, the helicopter maintains a constant speed while it maneuvers around obstacles.

This behavior is achieved by integrating the reactive layer (3D Dodger and Speed Control) with the path planning layer (Laplacian). Since the Laplacian planner looks far ahead to the next waypoint (possibly 100s of meters ahead) it will produce a nominal path that will not require the reactive layer to escape by flying straight up or by turning around. In the unlikely case that a large obstacle appears suddenly before it is incorporated by the planning layer will however cause an escape maneuver. Furthermore it is is necessary to incorporate a planning layer because the reactive layer alone can get stuck in some cluttered configurations of obstacles.

The planning layer runs at a lower frequency than the reactive layer and produces a smooth path from the current position to the next waypoint. However, the path is not a prescription for the helicopter to follow in the short run. Instead, a point on the path a fixed distance ahead of the current position provides a near term goal for the reactive algorithm. In case no path can be found by the path planning algorithm the default desired path is used by the reactive layer.

V. TESTBED

We fitted a Yamaha RMax helicopter (Fig. 9), originally designed for crop-dusting operations, with an H_∞ flight control system made by weControl. The resulting system provides a reliable platform capable of carrying a comparatively large payload (29 kg @ 1200 meters elevation). The flight control system provides a robust velocity loop even in the presence of strong gusty winds.

All processing of sensor data, path planning and collision avoidance is performed on a Linux computer with a Pentium M processor at 1.6 Ghz, 2 GB of RAM and 4 GB compact flash. The onboard accelerated graphics processor reduces computation of the virtual range sensor. The collision avoidance system relies on an accurate position estimate

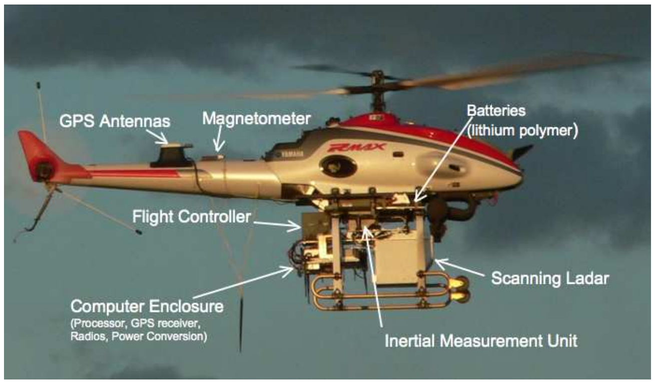
GPS Antennas
Magnetometer
Batteries (lithium polymer)
Flight Controller
Scanning Ladar
Computer Enclosure
(Processor, GPS receiver, Radios, Power Conversion)
Inertial Measurement Unit

Fig. 9. The helicopter testbed, a Yamaha RMax, used for our experiments has a payload of 29kg + fuel.

for registration of the ladar data into a coherent world map. We use the Novatel SPAN system with a HG-1700 inertial measurement unit to provide accurate estimates of six degree-of-freedom in position and attitude at 100Hz.

VI. RESULTS

We performed over 1000 successful obstacle-avoidance legs on actual hardware. Speeds varied from 3 m/s to 10 m/s, and the helicopter also avoided obstacles in up to 24 knot winds. Our layered architecture allowed the reactive system to quickly respond to obstacles, while the path planner found a good general path. On several instances, this ability to react quickly saved the helicopter from collisions with wires that the sensor could only register at short ranges. On the other hand, the path planner was invaluable for finding routes around very large objects (such as tree lines or buildings), where the reactive system would have a tendency to oscillate or get stuck if left to itself. Finally, in cases where both collision avoidance measures failed (during early development due to software bugs), the speed control system brought the helicopter to a stop before a collision occurred.

The development phase required 600 autonomous runs to develop satisfactory performance. Once we reached a final configuration, we ran more than 700 successful obstacle avoidance runs, many of which ran in long continuous sequences that maintained autonomy for up to 35 minutes (restricted by fuel limitations). Only one error (discussed below) required aborting a run. Other aborts were caused when the safety pilot couldn't keep up with the helicopter.

In Fig. 10 the helicopter follows a sequence of waypoints through trees, a wire and over a town. At waypoint *E* the helicopter is required to fly to a fairly tight hiding spot between trees and buildings. There are approximately 10 m between the obstacles and the location of the helicopter. The altitude(8 m above ground) is so low that the helicopter is not visible nor audible from the start of the path.

Notice also how the system begins to avoid obstacles in both the vertical and horizontal axis before committing to one or the other. This is a property of the 3-D Dodger algorithm as shown in equation 5, which begins to evade with both degrees of freedom until one direction becomes clear of obstacles. This behavior allows the system to implicitly decide which path provides the closest safe path as a function

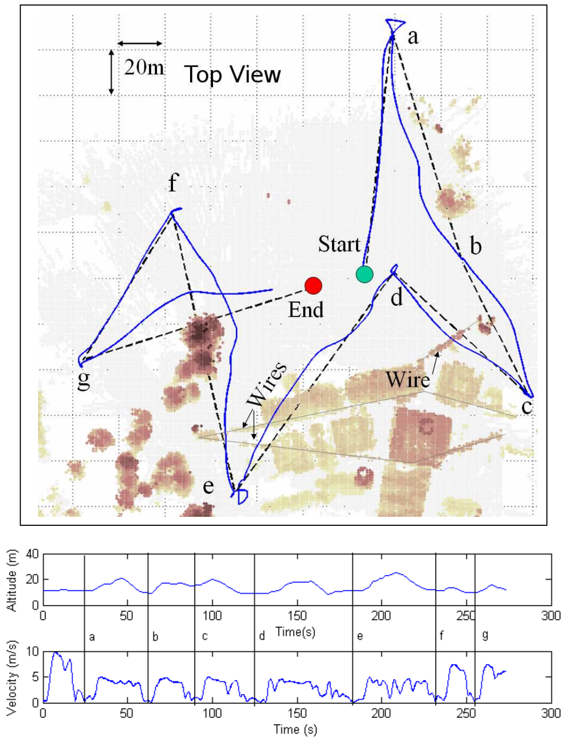

Fig. 10. A flight through the McKenna MOUT site at Ft. Benning at 4 m/s. 8 m/s was commanded at the first and 6 m/s at the last two segments of the path.

of vehicle dynamics. For example, in Fig. 10, the path between points *d* and *e* is blocked by a long building. The system begins to turn to go around it, and at the same begins to climb over. As the vehicle climbs, it encounters a free path on the vertical axis. The horizontal avoidance component quickly drops off, and the vehicle follows a vertical avoidance trajectory. The converse happens on the leg between *g* and *End*. The system chooses to fly around the tall tree rather than climb over it.

A ceiling or upper-limit of flight is helpful in cases where the helicopter has to stay low for stealth or low-level observation; however, forcing a robot to observe the ceiling can result in an impasse (such as coming to a long tree line which extends above the ceiling).We therefore force only the path planning algorithm to respect the ceiling constraint, while the reactive algorithm has no such constraint. The system will obey the constraint if the planner can see a way around, but otherwise will do what is necessary to avoid obstacles and continue the mission. An example of this situation during actual flight is shown in Fig. 11.

During the final testing phase of 700+ runs, we encountered only one dangerous bug in the behavior, twice in the same location. In a cluttered environment sensor occlusions are quite common and leave holes in the evidence grid. The path planner will consequently plan a path through unseen obstacles. This is not a problem as long as the sensor covers this unknown area before the vehicle traverses it, as Dodger will react immediately while the planner finds a new route. In the error case, the planned path went through a large patch of ground. The geometry of the scene and sensor

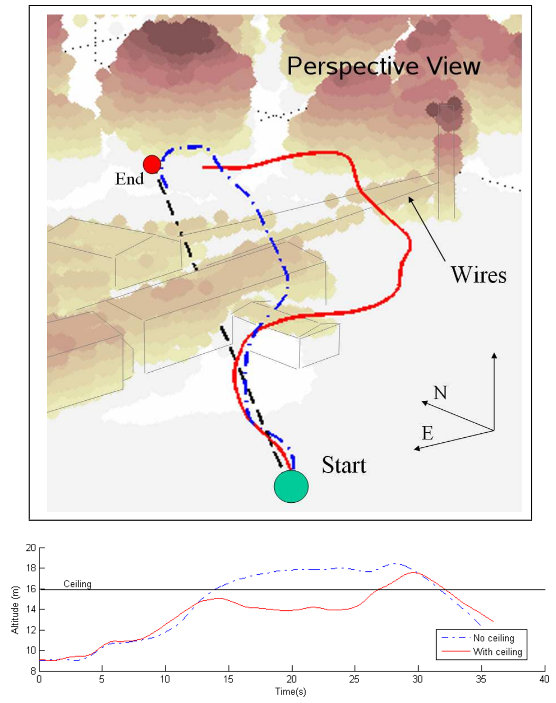

Fig. 11. Flying with and without ceiling when specified path is directly through a building. If a ceiling has been specified, the vehicle tries to stay below a specified altitude. On the run with ceiling, the system begins by flying around buildings, but then ignores altitude constraint and climbs to safely clear power lines. Key: Blue(dash-dot)$\longrightarrow$No ceiling. Red(solid)$\longrightarrow$With Ceiling.

FOV prevented the sensor from seeing the patch before the helicopter started descending in the direction of the ground. This behavior was rare, as any holes in the evidence grid are too small to fly through without having Dodger cause an evasion. While the simple addition of a ground plane in the evidence grid would have eliminated this behavior, we believe it is essential that a UAV is able to point the range sensor in the direction of travel.

Another perception problem is that of dust. The laser that we use is very sensitive so that it can see wires from large distances. Unfortunately, dust and pollen can have the same signature as a small wire. Despite some fast spatial filtering, observed dust clouds would occasionally divert the helicopter's flight path. Eventually the evidence grid would clear these clouds using negative evidence. This false-positive error does not cause dangerous behavior, but can impede low altitude flight. On windy days in dusty areas the system chose to fly higher to avoid dust clouds.

VII. CONCLUSIONS AND FUTURE WORK

We have developed a first-of-a-kind capability suited for UAVs that avoids obstacles of various sizes and shapes while flying close to the ground at significant speeds. In our experiments, the uninhabited helicopter started with no prior knowledge of the environment, having been provided only a list of coarse waypoints separated by up to hundreds of meters. The straight line path between the waypoints often intersected obstacles. While we regularly ran collision avoidance close to buildings, trees and wires between 4-6 m/s, the system was tested at speeds above 10 m/s. To accomplish these results our system uses a fast avoidance method that stays away from obstacles intelligently coupled with an online planning method that suggests a direction of travel.

We intend to address a few issues in future work. Our current method is not built to scale with significant increases in speed and avoids the obstacle with the margin irrespective of the speed. Ideally the reaction should depend on speed to react earlier to obstacles if the speed is fast and later if the speed is slow. Another issue is the tradeoff between sensitivity to small obstacles and an excessive reaction to large objects close by (such as in an urban canyon) even if they are not in the path of the vehicle.

VIII. ACKNOWLEDGMENTS

The results demonstrated were the result of work done by a large group of people. The authors gratefully acknowledge the contribution Mike Elgersma and Samar Dajani-Brown for the Laplacian algorithm; Henele Adams for his help with the Ladar; Alan Touchberry, Tara Schesser and Robert Schley for help with experimentation; Mark Delouis for extraordinary piloting and maintenance skills; and Brad Looney for leadership in the collision avoidance program.

REFERENCES

[1] B. Fajen and W. Warren, “Behavioral dynamics of steering, obstacle avoidance, and route selection,” *Journal of Experimental Psychology: Human Perception and Performance*, vol. 29, no. 2, 2003.
[2] P. C. Merrell, D.-J. Lee, and R. W. Beard, “Obstacle avoidance for unmanned air vehicles using optical flow probability distributions,” *Mobile Robots XVII*, vol. 5609, no. 1, pp. 13–22, 2004.
[3] J. Zufferey and D. Floreano, “Toward 30-gram Autonomous Indoor Aircraft: Vision-based Obstacle Avoidance and Altitude Control,” in *Proc. IEEE International Conference on Robotics & Automation*, 2005.
[4] R. Zapata and P. Lepinay, “Flying among obstacles,” in *Workshop on Advanced Mobile Robots (Eurobot)*, Zurich, Switzerland, September 6-8 1999, pp. 81–88.
[5] R. Zapata and P. Liepinay, “Collision avoidance of a 3d simulated flying robot,” in *Proc. International Symposium on Robotics and Automation*, Saltillo, Coahuila, Mexico, December 12-14 1998, pp. 113–120.
[6] H. S. and S. G.S., “Omnidirectional vision for an autonomous helicopter,” in *Proc. IEEE International Conference on Robotics & Automation*, vol. 1, 2003, pp. 558 – 563.
[7] J. Byrne, M. Cosgrove, and R. Mehra, “Stereo based obstacle detection for an unmanned air vehicle,” in *Proc. IEEE International Conference on Robotics & Automation*, May 2006.
[8] N. Vandapel, J. Kuffner, and O. Amidi, “Planning 3-d path networks in unstructured environments,” in *Proc. of the IEEE International Conference on Robotics & Automation*, 2005.
[9] Y. Kitamura, T. Tanaka, F. Kishino, and M. Yachida, “3-d path planning in a dynamic environment using an octree and an artificial potential field,” in *Proc. IEEE/RSJ International Conference on Intelligent Robots & Systems*, 1995.
[10] Y. Kitamura, T. Tanaka, F. Kishino, and M.Yachida, “Real-time path planning in a dynamic 3-d environment,” in *Proc. IEEE/RSJ International Conference on Intelligent Robots & Systems*, 1996.
[11] E. Frazzoli, M. A. Dahleh, and E. Feron, “Maneuver-based motion planning for nonlinear systems with symmetries,” *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1077–1091, Dec. 2005.
[12] Jackson, Sharma, Haissig, and Elgersma, “Airborne technology for distributed air traffic management,” *European Journal of Control*, vol. 11, no. 4-5, December 2005.
[13] C. I. Connolly and R. A. Grupen, “The application of harmonic functions to robotics,” *Journal of Robotic Systems*, vol. 10, no. 7, pp. 931–946, 1993.
[14] Z. X. Li and T. D. Bui, “Robot path planning using fluid model,” *Journal of Intelligent and Robotic Systems*, vol. 21, pp. 29–50, 1998.
[15] D. Shim, H. Chung, H. J. Kim, and S. Sastry, “Autonomous exploration in unknown urban environments for unmanned aerial vehicles,” in *Proc. AIAA GN&C Conference*, August 2005.
[16] M. C. Martin and H. Moravec, “Robot evidence grids,” Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-96-06, March 1996.