Autonomous Obstacle Avoidance and Maneuvering on a Vision-Guided MAV Using On-Board Processing

Lionel Heng, Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys

***Abstract*— We present a novel stereo-based obstacle avoidance system on a vision-guided micro air vehicle (MAV) that is capable of fully autonomous maneuvers in unknown and dynamic environments. All algorithms run exclusively on the vehicle's on-board computer, and at high frequencies that allow the MAV to react quickly to obstacles appearing in its flight trajectory. Our MAV platform is a quadrotor aircraft equipped with an inertial measurement unit and two stereo rigs. An obstacle mapping algorithm processes stereo images, producing a 3D map representation of the environment; at the same time, a dynamic anytime path planner plans a collision-free path to a goal point.**

I. INTRODUCTION

This work presents efficient real-time obstacle mapping and path planning algorithms for micro air vehicles (MAVs). The size of unmanned aerial vehicles in this class typically ranges between 15cm and 60cm, and these vehicles can be operated in both indoor and outdoor environments. MAVs are a challenging platform to work with; their payloads are limited by weight, thus often limiting the choice of exteroceptive sensors to cameras which make autonomy an even more non-trival task. Furthermore, the flight dynamics of a MAV are hard to model.

In this paper, we describe in detail the on-board algorithms that enable autonomous navigation and give an overview of our MAV platform as shown in Figure 1. The obstacle mapper efficiently constructs 3D virtual scans from range data derived from stereo image pairs; a virtual scan projects rays at regular angular intervals outward from the MAV until the rays either extend a maximum distance or intersect an obstacle. In turn, the mapper uses the virtual scan data to incrementally build a 3D map of the MAV's environment. At the same time, a path planning algorithm rapidly computes a suboptimal path, and improves the suboptimal bound of the initial path based on available planning time while immediately repairing the path upon discovery of previously unobserved obstacles. Finally, we discuss the results of autonomous flight experiments, and explore future directions.

A. Related Work

Recent research on MAVs has focused on real-time localization, perception, and path planning. An AscTec Hummingbird quadrotor [1] collects sensor data from both a Hokuyo URG laser rangefinder and a stereo camera, and transmits the data to a computer cluster which runs a localization algorithm. An operator selects waypoints on the map for the quadrotor to follow. Similarly in [2], a laptop runs a SLAM algorithm based on laser data from a MAV, and a person remotely controls the MAV. Blöesch et al. [3] demonstrates visual localization using a camera on an USB tether cable and processing on a laptop. The outdoor MAV of [4] utilizes an analog camera for object tracking and a GPS/INS system for position control. It is worthwhile to note that all these systems depend on a close-by processing unit to run mission-critical algorithms such as localization. They lack one important element of autonomy: to both map out obstacles and avoid these obstacles especially at close range in cluttered settings while in flight. Hence, the MAV inevitably crashes if the connection to its processing system is interrupted, even for a brief moment. This issue is particularly challenging in typical indoor environments where concrete walls block wireless signals and it is common to experience heavy interference from 2.4 GHz sources common in households such as Wi-Fi networks, ZigBee house installations, DECT phones, and wireless surveillance cameras. On-board processing resolves these issues, but requires computationally efficient algorithms, especially in the case of stereo vision where significant computational resources have to be allocated to image processing. The challenge of working with cameras is further exacerbated by the high level of noise associated with range data from stereo vision. Celik et al. [5] first demonstrated on-board indoor SLAM on a MAV using a monocular camera; however the MAV did not perform obstacle mapping and path planning.

Reactive behaviors for collision avoidance [6] are suscep-

Fig. 1. The PIXHAWK quadrotor platform.

The authors are with the Computer Vision and Geometry Lab, ETH Zurich, 8092 Zurich, Switzerland.
{hengli,lm,tpetri}@student.ethz.ch
{friedrich.fraundorfer, marc.pollefeys}@inf.ethz.ch

tible to the well-known local minima problem. Furthermore, most on-board planning algorithms [7], [8] for MAVs are not able to efficiently and quickly repair paths in unknown environments, especially when a large number of obstacles are observed over time. These algorithms can fail to react timely to moving obstacles.

Our work is novel in the aspect that we perform vision-based incremental mapping and path planning on a MAV with 100% on-board processing. We particularly adapt techniques successfully applied to ground robots which plan in high-dimensional spaces and in large-scale environments.

II. PIXHAWK PLATFORM

The PIXHAWK quadrotor platform is a soft- and hardware system optimally designed to support real-time on-board computer vision algorithms. In contrast to recent work [1], [2], [3], [4], our system design is computationally powerful enough to process in parallel up to four cameras on-board. Two stereo rigs on our vehicle point ahead and downwards, and allow the vehicle to localize itself and detect obstacles in its vicinity. Tight, hardware-guaranteed synchronization of images from all four cameras with the IMU paves the way for cutting-edge multi-view-IMU fusion algorithms. In the context of this paper, the synchronization is crucial in correctly matching the 3D stereo reconstruction results from the two front cameras to the global localization data based on the bottom cameras. Without this synchronization method, the global obstacle map will be signficantly distorted due to inaccurate 3D world coordinates of the point cloud data.

A. PIXHAWK Toolkit

The MAV middleware supports three important main features: reliable vision-IMU synchronization (and synchronization of multiple cameras) on the hardware level, an efficient central image hub which allows multiple algorithms to simultaneously use image output from the same cameras in parallel, and a homogenous, efficient communication architecture for on- and off-board communication that scales from serial ports to UDP sockets.

1) Camera - IMU Synchronization: Synchronization of all on-board cameras is extremely crucial for multi-view geometry applications such as stereo vision. For approaches using vision-IMU fusion, it is important to have IMU measurements available at the time the cameras capture each image. The toolkit triggers all on-board cameras using the inertial measurement unit hardware, and attaches IMU metadata to the images.

2) Image Transfer: To enable localization, pattern recognition, and depth triangulation from stereo images, several computer vision algorithms work in parallel on the platform. This is facilitated by the central image hub, which distributes the images of all cameras with IMU metadata to all subscribing processes. With an efficient implementation based on shared memory, the distribution is resource-effective.

3) Communication: Several robotics packages for large unmanned ground, surface, and air vehicles exist; examples include CARMEN, ROS and CLARAty. However, these packages implicitly assume that all processes using their frameworks support either IEEE 802.3 Ethernet or IEEE 802.11a/b/g/n Wi-Fi. Many system components in the MAV domain are connected via UART, and inter-system communications, in particular, between the MAV and the ground station, are not based on Wi-Fi, but rather on radio modems and wireless UART emulation. For outdoor MAVs, this mode of communications is necessary for long-range data transmission; for indoor MAVs operating in buildings, radio modems are much more robust to interference sources in the 2.4 GHz band. Therefore, our communication backend in the form of the MAVLink protocol scales from UART to UDP interfaces and is used by both on- and off-board components operating the unmanned air vehicle. Messages are transparently routed without the need for bridge and adapter processes, effectively reducing the system latency to a minimum. The MAVLink protocol messages are serialized and deserialized using C89-compliant C-code which can execute on the ARM7 microcontroller of the inertial measurement unit. Manual coding of the serialization routines such as in [9] is error-prone; all code is automatically generated from a XML protocol definition, enabling quick addition of new message types. The automated code generation ensures that all messages are well-formed.

B. Quadrotor Design

The high payload capacity of 400g with a maximum dimension of 55cm is made possible with an optimized vehicle design and carbon composite materials. The entire mechanical structure is fabricated from 0.5cm thick carbon fiber sandwich sheets reinforced with an internal layer of honeycomb material.

1) Electronics: The on-board electronics includes the px-IMU inertial measurement unit (IMU) for camera triggering, and on-board position estimation and control. The pxCOMex processing board can carry any processing module that adheres to the microETXexpress form factor. It is currently used with a Kontron module with an 1.86 GHz Intel Core 2 Duo processor and 2 GB DDR3 RAM.

2) Frame and Propulsion: The four motors, motor controllers, the on-board computer, and the IMU are mounted on a cross-shaped carbon fiber composite frame. Two pairs of motors rotate in the opposite direction, with the motors along the x-axis rotating clockwise and the motors along the y-axis rotating counter-clockwise. Each motor with its attached 8" propeller contributes a maximum thrust of 452g, yielding a thrust-to-weight ratio of 2:1. The camera mount holds up to four Point Grey Firefly MV cameras with remodeled S-mount lens holders; we use two stereo camera setups with a 5cm baseline each.

3) Attitude and Position Estimation and Control: The IMU estimates the current attitude with a state observer filter which predicts the gravity vector using the accelerometers, gyroscopes, and either the magnetometer or estimated yaw from the computer vision pipeline. The position is estimated using a standard Kalman filter, implemented as four independent Kalman filters following a constant speed model. These filters each estimate the linear and angular displacements and velocities. As the dynamics of a quadrotor are loosely coupled in the x, y, and z directions, the quadrotor dynamics can be modelled in these three independent dimensions together with the yaw. Hence, the overall vehicle control system is decomposed into four independent PID controllers which control the x, y, and z positions and yaw angle. The output of these controllers is the set of roll and pitch angles along with the rotational difference δw of the two rotor pairs. This attitude setpoint is used as the input to three PID attitude controllers which regulate the roll, pitch and yaw angles. The output of these attitude controllers is the orientation of the collective thrust normal vector spanned by the four motor thrust vectors. The additional degree of freedom (scaling of the x vs y axis vectors) is used to control the rotational speed around the yaw axis.

4) Operator Control Unit: We use an open-source ground control station software called QGroundControl that we earlier developed as part of the PIXHAWK project to monitor and control the MAV. Figure 2 shows QGroundControl in the data plotting mode.

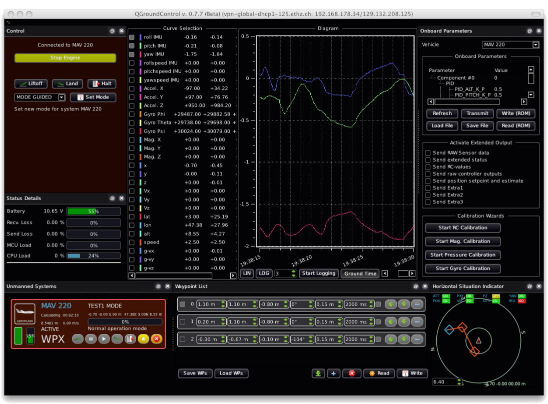


Fig. 2. QGroundControl operator control unit.

III. INCREMENTAL OBSTACLE MAPPING

We compute disparity data from stereo image pairs, and subsequently, compute a point cloud. We build a 3D virtual scan from the point cloud data, and use this scan to update a 3D global occupancy map. Obstacles are then inferred by applying a cost threshold to each cell in the occupancy map.

A. Range Data from Stereo

With each stereo image pair, we use the OpenCV implementation of the sum-of-differences block matching stereo correspondence algorithm to build a dense 320 x 240 disparity map. Subsequently, we compute the depth to points in the scene using the disparity-to-depth matrix:

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{1}{b} & 0 \end{bmatrix} \begin{bmatrix} i \\ j \\ d \\ 1 \end{bmatrix} \quad (1)$$

where (i, j) are the image coordinates of the pixel, d is the disparity, (c_x, c_y) are the coordinates of the principal point of the camera, f is the focal length, b is the baseline, and $[x\ y\ z]^T = \frac{1}{W}[X\ Y\ Z]^T$ are the world coordinates of the pixel relative to the camera coordinate system. The values of c_x, c_y, f, and b are obtained from an one-time calibration.

B. 3D Virtual Scan

Since a typical point cloud built from stereo range data is very dense, it is not computationally feasible to process each point. In addition, the point cloud has a significant number of outliers. Hence, we downsample the data and remove outliers by building a 3D spherical grid centered on the MAV and taking the median of the range readings located in each spherical cell; a virtual ray is defined as extending from the MAV to that median point. We then define a virtual scan as a set of virtual rays; each ray $r(\theta, \varphi)$ records the distance to the obstacle it first hits where θ is the elevation angle and φ is the azimuth angle. An example of a virtual scan is shown in Figure 3. Since we treat a stereo camera as a multi-beam sensor, the virtual scan structure is a natural data representation for efficient downsampling with minimal loss of information.

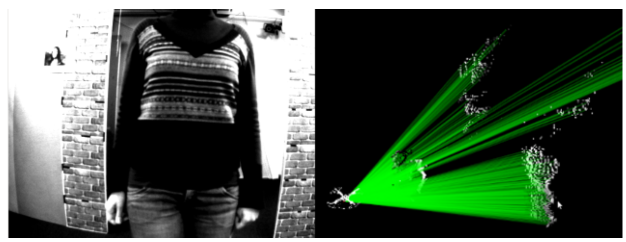


Fig. 3. An image of the scene with a corresponding visualization of the point cloud data and virtual scan.

C. 3D Occupancy Map

We use a modified version of the octree-based 3D map structure [10] to represent the MAV's environment; the 3D map is aligned to the world coordinate system. Each cell in the map represents the log odds of the probability of occupancy given the measurements $z_1, ..., z_t$:

$$L(m_{x,y,z}|z_{1:t}) = \log \frac{p(m_{x,y,z}|z_{1:t})}{1 - p(m_{x,y,z}|z_{1:t})} \quad (2)$$

where $m_{x,y,z}$ is a binary variable indicating whether the cell with coordinates (x, y, z) is occupied. We update the cells which the virtual rays traverse across using the update rule:

$$L(m_{x,y,z}|z_{1:t}) = L(m_{x,y,z}|z_{1:t-1}) + L(m_{x,y,z}|z_t) \quad (3)$$

where $L(m_{x,y,z}|z_t)$ is the inverse sensor model that accounts for the uncertainty inherent in stereo range data at varying distances. We determine whether a cell is an obstacle by recovering its occupancy probability from the log odds value and performing a simple threshold on the probability.

The multi-resolution sampling and compact size make the map represention a natural fit for MAVs with limited computational resources. A lossless compression method converts the map structure to a binary stream which is then transmitted through the middleware to the path planner.

IV. Incremental Path Planning

Our path planner searches for a path from the current position of the MAV to an user-defined goal in 3D state space (x, y, z). In this paper, we assume the MAV to be a holonomic vehicle, and hence, we do not include the roll-pitch-yaw in the state vector.

A. State Lattices

We use the state lattice concept [11] to decouple the path planning problem into two distinct subproblems: vehicle mobility constraints and graph search. Within our state lattice of resolution 0.25m, a trajectory is discretized into a set of motion primitives which can be computed offline: one factor that improves the efficiency of our path planner. Figure 4 shows an example of a control set with 3D motion primitives.

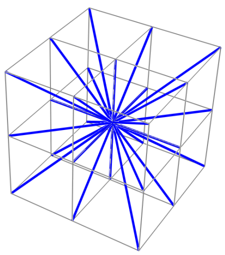


Fig. 4. A control set with 26 motion primitives.

To determine the cost of a motion, we add the costs of the cells in the motion's swath. A swath is the set of map cells occupied by the MAV's volume during the motion, and is precomputed for each motion primitive. This method is more efficient than the widely-used method of simulating the motion to check which cells the MAV can safely traverse.

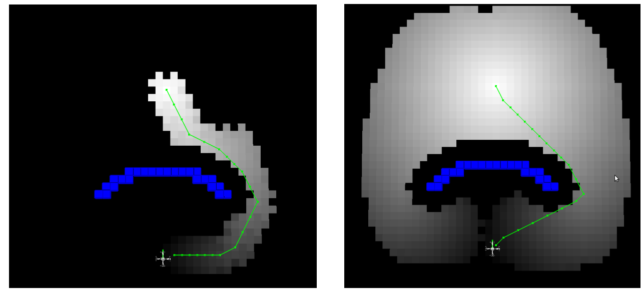
B. Graph Search

For graph search, we use an anytime replanning search algorithm called Anytime Dynamic A* (ADA*) [12]. At the beginning, ADA* quickly computes a suboptimal path to the goal, and continually improves the bound on the initial solution based on available planning time. The heuristic inflation factor ϵ is the key; as shown in Figure 5, by decreasing the inflation factor, the solution improves in terms of optimality but more states are explored.

Throughout the planning process, ADA* uses the same search tree. Furthermore, ADA* incrementally repairs the current path in response to new obstacles that have just been observed. We use an admissible and consistent heuristic required by ADA* to guide the search: the cost of a path from the MAV to the goal given an obstacle map. We calculate the path cost via Dial's implementation of Dijkstra's search algorithm [13].

V. Implementation

We focus on autonomous flight at a fixed height in a lab environment and discuss the details of the implementation used to achieve this autonomy.



(a) $\epsilon = 2.5$ (b) $\epsilon = 1.0$

Fig. 5. Any computed path can be decomposed into motion primitives. ADA* gradually improves the computed path over time by decreasing the heuristic inflation factor.

A. Pose Estimation

We estimate the MAV's pose using either our artificial feature-based localization software or a Vicon motion capture system. The feature-based localization [14] tracks unique ARTK markers [15] with known positions on the ground using downward-pointing cameras and estimates the pose of the camera; we apply a coordinate transform to find the MAV's pose. In our Vicon motion capture system, strobe units reflect infrared light off small retro-reflective markers attached to the MAV. Camera units use the reflectance information to compute the image positions of the reflected markers, and subsequently, the MAV's pose.

B. Obstacle Mapping and Path Planning

We use a virtual scan resolution of $0.5°$ in our obstacle mapping. For path planning in a state lattice, we use a control set with 16 motion primitives, all of which lie on the same z-plane. Every planned path maintains a minimum safety clearance of 1m from obstacles.

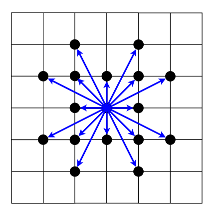


Fig. 6. Our control set implementation contains 16 motion primitives.

On the MAV platform, both the obstacle mapping and path planning software run at approximately 10 Hz.

VI. Experiments and Results

In one set of experiments, we placed an obstacle directly between the MAV and a designated goal. A snapshot of the obstacle mapping process is shown in Figure 7. The MAV was able to plan a path around the obstacle as shown in Figure 8; blue cells denote obstacles, red cells denote locations in close proximity to obstacles and which are not considered safe for the MAV, and yellow cells represent states that the MAV has yet to explore. For other cells, the grayscale color indicates the cell cost; the more white the

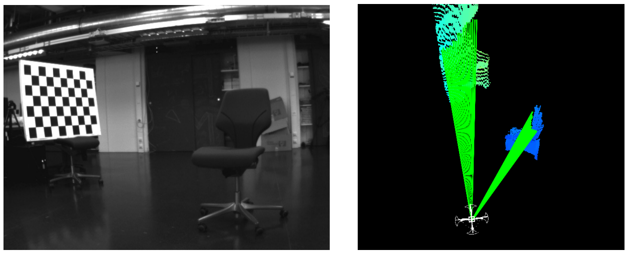


Fig. 7. The chessboard and the chair as seen on the left are marked as candidate obstacles in the virtual scan shown on the right.

cell, the closer the cell to the goal. The planned path is marked in green.

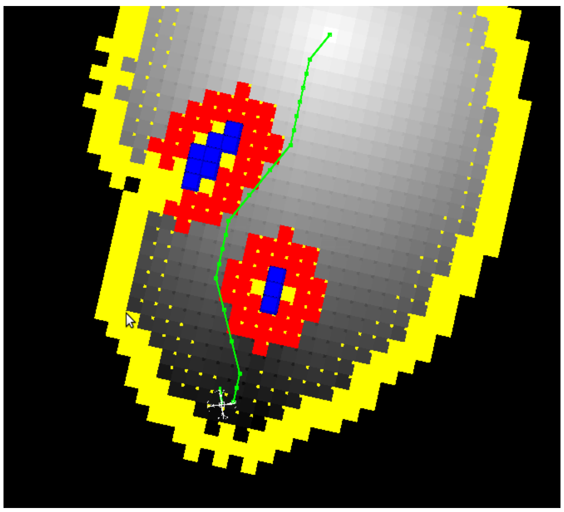


Fig. 8. A visualization of the path planning in progress.

Another set of experiments demonstrates the path planning algorithm adjusting the MAV's path when unseen obstacles are detected. In Figure 9, the top image shows the obstacle configuration, the middle image illustrates the initial plan generated at the beginning of the test run, and the bottom image illustrates the repaired path due to newly observed obstacles. At the start, the MAV detects the row of four chairs but does not detect the other two chairs as they are hidden from the MAV's view by the row of chairs. When the MAV moves into a position where it detects the two previously unobserved chairs, the new obstacles are added to the global map, and the path planner repairs its path to avoid these new obstacles.

Using the results of another experiment, we compare the planned path to the actual trajectory of the MAV which is shown in red in Figure 10. We observe that the trajectory significantly diverges from the planned path at two points; at the beginning and mid-way. Initial large PID errors occur as a result of the MAV lifting off from the ground towards the first waypoint; the position controller attempts to stabilize the MAV which explains the beginning part of the trajectory. Near the midpoint of the trajectory, the MAV's close proximity to an obstacle affects the air flow around the MAV; the air turbulence pushes the MAV away from the obstacle.

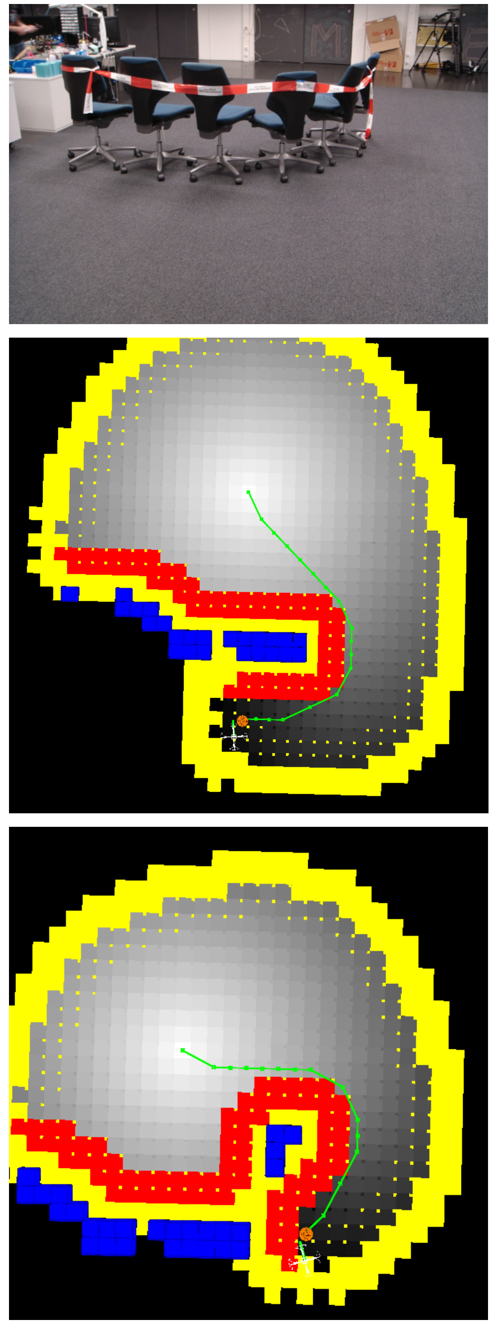


Fig. 9. Various stages of path planning in an unknown environment.

In turn, the position controller attempts to compensate for the resulting disturbance; the MAV slowly returns to the trajectory before continuing to track its planned path.

The results of one experiment in which the MAV flies in a S-curve can be viewed in the accompanying video or at `http://www.inf.ethz.ch/~hengli/mav.mp4`.

VII. CONCLUSIONS AND FUTURE WORK

A. *Conclusions*

Limited on-board computing resources and the helicopter's 3D mobility greatly motivate our work on fast obstacle mapping and path planning methods that are able to scale

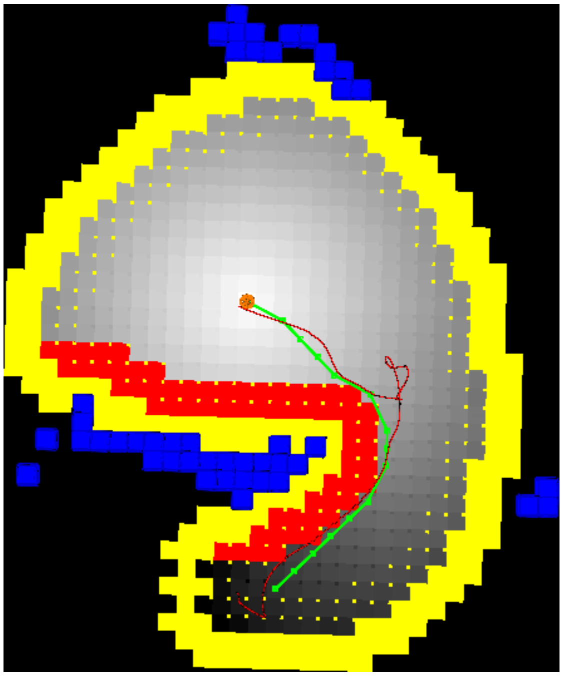


Fig. 10. Comparison of the actual trajectory with the planned path.

to large environments. We have demonstrated a standalone system that is capable of maneuvering around obstacles in unknown environments. Our path planning framework leaves a small computational footprint. In addition, our presented system provides a novel on- and off-board architecture for autonomous and computer vision-based flight with tightly-coupled IMU and GPS integration. This architecture facilitates the development of onboard computer vision algorithms.

B. Future Work

As the system presented in this paper is a suitable platform for on-board computer vision applications, future work will focus on natural feature-based localization and mapping for autonomous indoor and outdoor flight. The current system load for artificial feature-based localization is 10% of the maximum CPU capacity; sufficient capacity is available on-board for mapping and planning. As the system design provides a precise common time base, fusion of IMU, GPS, and vision data will be a future extension for outdoor navigation. On a multi-system level, the lightweight MAVLink protocol provides an ideal basis for future swarm extensions. As all processing is done on-board and allows the vehicle to be fully autonomous, the communications bandwidth does not limit the number of vehicles.

Furthermore, we are working on fully extending the MAV's autonomous navigation capabilities to 3D space. Path planning will be done in higher-dimensional state spaces to obtain smooth paths that incorporate the kinematic motion constraints of the helicopter.

VIII. ACKNOWLEDGMENTS

We would like to thank our contributing students (in alphabetical order) Bastian Bücheler, Andi Cortinovis, Fabian Landau, Laurens Mackay, Tobias Nägeli, Philippe Petit, Martin Rutschmann, Amirehsan Sarabadani, Christian Schluchter, and Oliver Scheuss for their hard work in the IMAV 2010 competition last semester, and the students of the previous semesters for the foundation they provided. We would also like to thank Raffaello d'Andrea and Sergei Lupashin (ETH IDSC) for the fruitful discussions.

The first author was funded by the DSO National Laboratories Postgraduate Scholarship. In addition, this work was supported in parts by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant #231855 (sFly) and by the Swiss National Science Foundation (SNF) under grant #200021-125017.

REFERENCES

[1] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, Stereo Vision and Laser Odometry for Autonomous Helicopters in GPS-denied Indoor Environments, *In Proc. SPIE Conference on Unmanned Systems Technology XI*, 2009.

[2] S. Grzonka, G. Grisetti, and W. Burgard, Towards a Navigation System for Autonomous Indoor Flying, *In Proc. International Conference on Robotics and Automation*, 2009.

[3] M. Blöesch, S. Weiss, D. Scaramuzza, and R. Siegwart, Vision Based MAV Navigation in Unknown and Unstructured Environments, *In Proc. International Conference on Robotics and Automation*, 2010.

[4] R. He, A. Bachrach, M. Achtelik, A. Geramifard, D. Gurdan, S. Prentice, J. Stumpf, and N. Roy, On the Design and Use of a Micro Air Vehicle to Track and Avoid Adversaries, *International Journal of Robotics Research*, 29(5):529-546, 2010.

[5] K. Celik, S. Chung, M. Clausman, and A. Somani, Monocular Vision SLAM for Indoor Aerial Vehicles, *In Proc. International Conference on Intelligent Robots and Systems*, 2009.

[6] S. Soundararaj, A. Sujeeth, and A. Saxena, Autonomous Indoor Helicopter Flight using a Single Onboard Camera, *In Proc. International Conference on Intelligent Robots and Systems*, 2008.

[7] M. Wzorek, J. Kvarnstrom, and P. Doherty, Choosing Path Replanning Strategies for Unmanned Aircraft, *In Proc. International Conference on Automated Planning and Scheduling*, 2010.

[8] H. Yu, R. Beard, and J. Byrne, Vision-based local multi-resolution mapping and path planning for Miniature Air Vehicles, *In Proc. American Control Conference*, 2009.

[9] R. Touchton, D. Kent, T. Galluzzo, C. Crane, D. Armstrong, N. Flann, J. Wit, and P. Adist, Planning and modeling extensions to the Joint Architecture for Unmanned Systems for applications to unmanned ground vehicles, *SPIE Defense and Security Symposium*, 2005.

[10] K. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, OctoMap: A Probabilistic, Flexible, and Compact3D Map Representation for Robotic Systems, *In Proc. ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.

[11] M. Pivtoraiko, R. Knepper, and A. Kelly, Differentially Constrained Mobile Robot Motion Planning in State Lattices, *Journal of Field Robotics*, 26(3):308-333, 2009.

[12] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, Anytime Dynamic A*: An Anytime, Replanning Algorithm, *In Proc. International Conference on Automated Planning and Scheduling*, 2005.

[13] R. Dial, Algorithm 360: Shortest Path Forest with Topological Ordering, *Communications of ACM, 12:632-633*, 1969.

[14] G. Lee, M. Achtelik, F. Fraundorfer, M. Pollefeys, and R. Siegwart, A Benchmarking Tool for MAV Visual Pose Estimation, it In Proc. International Conference on Control, Automation, Robotics, and Vision, 2010.

[15] D. Wagner, and D. Schmalstieg, ARToolKitPlus for Pose Tracking on Mobile Devices, *In Proc. 12th Computer Vision Winter Workshop*, 2007.