Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV

Friedrich Fraundorfer, Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys
Computer Vision and Geometry Lab, ETH Zürich, Switzerland

***Abstract*— In this paper, we describe our autonomous vision-based quadrotor MAV system which maps and explores unknown environments. All algorithms necessary for autonomous mapping and exploration run on-board the MAV. Using a front-looking stereo camera as the main exteroceptive sensor, our quadrotor achieves these capabilities with both the Vector Field Histogram+ (VFH+) algorithm for local navigation, and the frontier-based exploration algorithm. In addition, we implement the Bug algorithm for autonomous wall-following which could optionally be selected as the substitute exploration algorithm in sparse environments where the frontier-based exploration under-performs. We incrementally build a 3D global occupancy map on-board the MAV. The map is used by the VFH+ and frontier-based exploration in dense environments, and the Bug algorithm for wall-following in sparse environments. During the exploration phase, images from the front-looking camera are transmitted over Wi-Fi to the ground station. These images are input to a large-scale visual SLAM process running off-board on the ground station. SLAM is carried out with pose-graph optimization and loop closure detection using a vocabulary tree. We improve the robustness of the pose estimation by fusing optical flow and visual odometry. Optical flow data is provided by a customized downward-looking camera integrated with a microcontroller while visual odometry measurements are derived from the front-looking stereo camera. We verify our approaches with experimental results.**

I. Introduction

Quadrotor MAVs are an ideal choice for autonomous reconnaissance and surveillance because of their small size, high maneuverability, and ability to fly in very challenging environments. To perform these tasks effectively, the quadrotor MAV must be able to do precise pose estimation, navigate from one point to another, map the environment, and plan exploration strategies. Ideally, all these processes have to run on-board the quadrotor especially in GPS-denied environments.

Many researchers demonstrated some of these abilities by using laser range finders [2], [8], [22], artificial marker aided vision [9], [19] and/or a Vicon motion capture system [15]. A camera is sometimes used to detect loop closure opportunities in laser-based SLAM. In many of these systems, the choice of the sensors becomes an impedance factor when designing the quadrotor to achieve full autonomy in mapping and exploration. For example, the considerable weight and power consumption of the laser range finder pose a problem for quadrotor MAVs with stringent payload and power limitations. Other sensors such as artificial marker aided vision and the Vicon system require modifications of the environment, heavily limiting autonomy of the quadrotor.

In comparison with the popular choice of laser range finder, artificial marker aided vision and Vicon motion capture system, the use of a camera as the main sensor for quadrotor MAVs has largely been undermined. This is mainly due to the need for significant computational resources for most computer vision algorithms and the lack of robustness in pose estimation from cameras. However, a camera offers several advantages over the other sensors for quadrotor MAVs. It is much more light-weight and consumes less power, and the camera does not require any modifications to the environment.

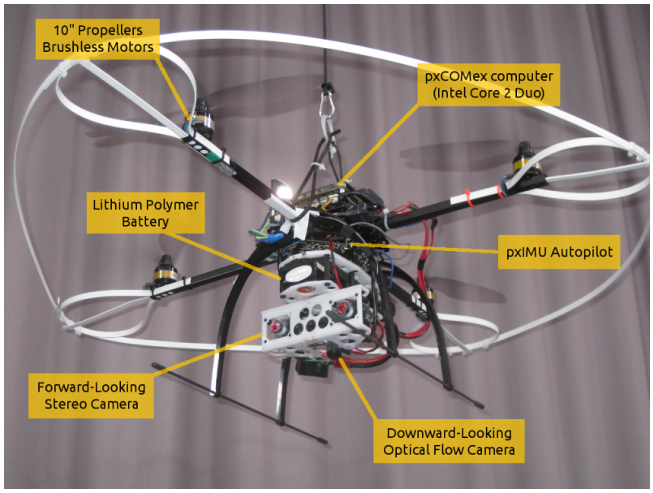


Fig. 1. Our PIXHAWK quadrotor system.

The main objective of this paper is to demonstrate the feasibility of autonomous mapping and exploration for a quadrotor with cameras as its main sensor. Our quadrotor system as shown in Figure 1 is fitted with a customized computer based on a Intel®Core™2 Duo processor that meets our computationally-demanding computer vision processes. We improve the robustness of the pose estimation by fusing visual odometry readings from a front-looking stereo camera with partial state estimates from a customized downward-looking camera. The downward-looking camera has a built-in ARM®Cortex-M4F microcontroller with a Digital Signal Processing (DSP) extension which runs an optical flow algorithm. The quadrotor moves autonomously from one point to another using the VFH+ algorithm [23]. It plans exploration strategies based on the frontier-based exploration algorithm [25] where frontiers are selected from

a global occupancy map [14] which is constructed on-board. The frontier-based exploration algorithm works very well in dense environments where distinctive frontiers could be identified. The algorithm, however, under-performs in sparse environments where clearly defined frontiers are absent. As a remedy, we implement the Bug algorithm [7] for autonomous wall-following which could be optionally selected as the substitute exploration algorithm in cases where the frontier-based exploration under-performs. Images from the front-looking camera are transmitted to the ground station where large scale SLAM is done off-board with pose-graph optimization. We use the g2o [16] implementation for SLAM. Each vertex in the graph represents the quadrotor pose, and each edge represents the constraints between each quadrotor pose obtained from the visual pose estimation. Additional edge constraints are also added from loop-closure detections in the camera images via the vocabulary tree [11], [20]. We discuss some of the related works in Section II. In Section III, we show the hardware and software design of our quadrotor system. The pose estimation for the quadrotor is described in Section IV. In Section V, we describe our implementation of the autonomous mapping and exploration algorithms in detail. The off-board SLAM is discussed in Section VI. Lastly, we show the experimental results in Section VII.

II. Related work

Our work in this paper is closely related to the works of [2], [22]. In [2], Bachrach et al. use laser scan matching to estimate the MAV's pose. SLAM is performed with incremental pose-graph optimization, and loop detection is done via laser scan matching. The SLAM and planning processes run on a ground station while pose estimation runs on-board the MAV. The frontier-based exploration approach [25] is used for autonomous mapping and exploration of the environment. Results of large-scale exploration and SLAM are shown. In [22], Shen et al. show multi-floor navigation based on the exclusive use of on-board processing. The MAV pose is estimated by laser scan matching, and optimized using pose-graph optimization SLAM with an iterative Kalman filter approach. Loops in the pose-graph are detected using a camera and vocabulary tree matching. In their approach, camera images are only used for loop detection. Pose estimation and SLAM run on-board the MAV. The pose-graph includes full 6-DOF poses so that it is possible to distinguish poses at different altitudes, thus allowing multi-floor mapping. They demonstrate autonomous operation and mapping of the MAV where the flight path is defined by manual setting of waypoints in the currently estimated map. They also show results from large-scale SLAM; however, instead of actual flight, the MAV was carried around by an operator. Despite the close similarities to our work, we highlight one important difference; we use cameras whereas both described works use a laser range finder as the main sensor. This sensor difference requires a vastly different set of components in our system. Laser range finders are also used in [8], [12] which focus on pose estimation and SLAM without autonomous flight behaviors.

Artificial marker aided vision-based flights are demonstrated in [9], [19] In [9], Eberli et al. show autonomous takeoff, landing, and hovering using one circular artificial marker. This marker is detected in images from a monocular camera, and used to estimate the pose of the MAV. The pose data is fed back to a set-point controller in order to achieve autonomous takeoff, landing, and hovering. Meier et al. [19] pushed the technology further by demonstrating autonomous takeoff, landing, hovering, and waypoint following over a much larger space using numerous ARToolKitPlus [24] markers laid on the ground. Their MAV is equipped with a single downward-looking camera which detects these markers, and computes the MAV pose. However, the autonomy in both works is largely restricted to the area covered by the artificial markers. In [15], Heng et al. showed autonomous occupancy grid mapping, global path planning, and obstacle avoidance of the MAV. However, the poses of the MAV are obtained from the Vicon system, thus restricting the autonomy of the MAV to the Vicon space.

Markerless computer vision control based on a single downward-looking camera with wide-angle lens has been demonstrated for landing, takeoff, and hovering and small-scale mapping in [1], [5]. However, they cannot demonstrate other autonomous behaviors such as path planning and exploration because the downward-looking camera is not able to see the front environment of the quadrotor. In contrast, our main sensor is a forward-looking stereo camera complimented with a downward-looking camera for optical flow. Consequently, this allows our quadrotor to have better perception of the environment, and thus, the ability to do autonomous path planning and exploration.

Bills et al. [4] implemented an approach for higher level navigation using a Parrot AR.Drone. Utilizing image classification, the MAV was able to follow corridors, and even make turns, but no notion of a metric map was involved.

III. Our Quadrotor Platform

In this section, we give a brief description of the hardware and software designs of our quadrotor platform which is shown in Figure 1. The quadrotor is designed for autonomous flights with the capability of processing computer vision algorithms on-board. Figure 2 shows a schematic of the hardware and software components on our quadrotor platform. The quadrotor platform weighs 1.4 kg, spans a diameter of 0.6 m and has a payload capacity of up to 600 g. We arrange two Matrix Vision mvBlueFOX cameras with 752x480 resolution in a front-looking stereo rig with a 80 mm baseline. Images from the cameras are inputs to the stereo processes which compute disparity and depth maps. The disparity and depth maps are used as inputs to the visual odometry and Artificial Intelligence (AI) Planner that give the quadrotor all the autonomous behaviors such as local navigation and exploration. Our visual odometry also requires the images directly from the stereo camera (see Section IV-B for more details). All stereo processes, visual odometry, and AI Planner run on our customized flight computer with an Intel®Core™2 Duo 1.86 GHz processor

and running Ubuntu. The AI Planner selects new waypoints and passes these waypoints to the position controller which moves the quadrotor from its current location to the new waypoint. Our position controller is a closed-loop PID controller with the full state $(x, y, z, \varphi, \theta, \phi)$ (see Section IV for state estimation) of the quadrotor as feedback. Details of the implementation of the position controller can be found in one of our previous works [17].

We add another downward-looking camera that runs an optical flow algorithm on a built-in ARM®Cortex-M4F microcontroller with DSP extension. Data from a customized inertial measurement and autopilot unit (pxIMU), and ultrasonic distance sensor is also included for partial state estimation with optical flow (see Section IV-A for more details). The partial state estimates are then fused with the visual odometry readings to obtain more robust pose estimates. The pxIMU is also used for attitude control which stabilizes the quadrotor flight. All the estimation and control algorithms are executed on an ARM7 microcontroller in hard real-time. Note that two separate microcontrollers are used for the optical flow and control. The pxIMU also synchronizes and timestamps all on-board sensors including images from the cameras. This hardware synchronization allows us to fuse outputs from different sensors and correctly estimate sensor delays. The outputs from the stereo processes, visual odometry, and pose estimator are sent to the ground station where SLAM and loop-closure detection are computed off-board.

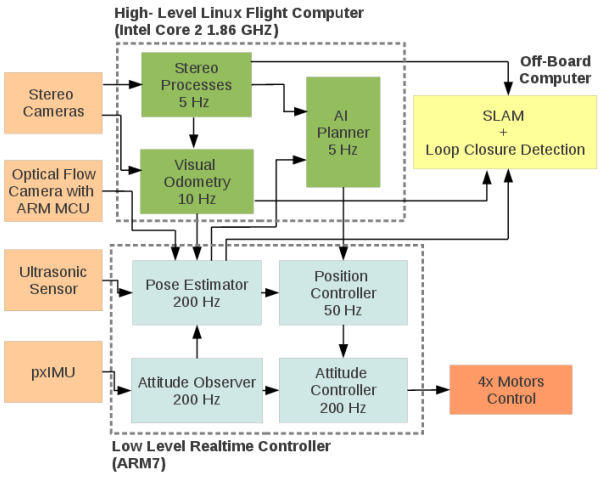


Fig. 2. System hardware and software design.

IV. Pose Estimation

A. Partial State Estimation with Optical Flow

We estimate the x and y velocities of the quadrotor in the low-level controller using the optical flow output from the microcontroller integrated with the downward-looking camera. We assume that there is no coupling effect between the velocities in the two axes, and that each of these quantities can be estimated with an independent 1-dimensional Kalman filter.

The dynamic model of the quadrotor is obtained from system identification. We assume that the quadrotor is a point mass, and that the velocities in the x and y axes (v_x, v_y) are decoupled and can be computed with a linear state equation given in Equation (1) that takes the pitch θ^I and roll φ^I angles in the inertial frame as the input in the respective axis. In this work, θ^I and φ^I are directly read from the IMU on the quadrotor.

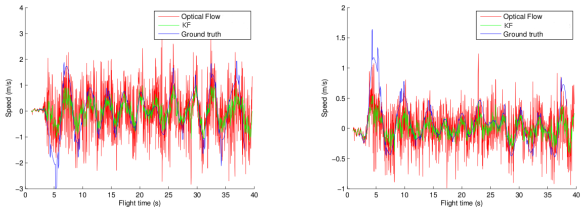
$$\begin{aligned} v_x(k) &= a_x v_x(k-1) + b_x \theta^I(k) \\ v_y(k) &= a_y v_y(k-1) + b_y \varphi^I(k) \end{aligned} \tag{1}$$

The parameters of the state equations (a_x, b_x, a_y, b_y) are determined by fitting the ground truth velocities of the quadrotor collected from a Vicon motion capture system with the pitch and roll values collected from the IMU on the quadrotor using the ARX model found in the Matlab System Identification toolbox [1].

The measurements of the x and y velocities for the Kalman filter come from the optical flow. The x and y velocities (v_x^o, v_y^o) measured by the optical flow camera are in pixel space, and these quantities are converted into metric space (v_x^m, v_y^m) based on the altitude h of the quadrotor which is measured by the ultrasonic sensor. Equation (2) shows this conversion where λ_x and λ_y are scaling constants. $\Delta\theta^B$ and $\Delta\varphi^B$ are instantaneous pitch and roll angles of the quadrotor in the body frame obtained by integrating the angular rates from the IMU. Note that $\Delta\theta^B$ and $\Delta\varphi^B$ are assumed to be small because small angle approximation is used in the derivation of Equation 2. $\Delta\theta^B$ and $\Delta\varphi^B$ compensate for the extra optical flow introduced by tilting the optical flow camera off the horizontal plane. v_x^m and v_y^m are in the body frame and have to be transformed into the inertial frame before it can be fused with the full state estimate from visual odometry. This is easily achieved by a pre-multiplication with the rotation matrix obtained from the IMU readings.

$$\begin{aligned} v_x^m &= h\left(\lambda_x v_x^o - \Delta\theta^B\right) \\ v_y^m &= h\left(\lambda_y v_y^o - \Delta\varphi^B\right) \end{aligned} \tag{2}$$

Figure 3 shows the comparison of the velocity estimates from the Kalman filter with the Vicon ground truth and optical flow. Our velocity estimates are smoother and sufficiently close to the ground truth. The partial state of x and y in the inertial frame are obtained by integrating the velocities over time.



(a) (b)

Fig. 3. Results of x and y velocities from the Kalman filter.

[1] www.mathworks.com/products/sysid/

B. Full State Estimation with Visual Odometry

We use the front looking stereo camera to compute the visual odometry that estimates the full state $(x, y, z, \varphi, \theta, \phi)$ of the quadrotor in the inertial frame. As mentioned earlier, the visual odometry is computed on the high-level flight computer. The visual odometry outputs are then fused with the integrated partial state estimates x and y from the optical flow to get better full state estimates. The fusion is done with a low-pass filter on the low-level controller.

In contrast to many stereo camera visual odometry algorithms that do consecutive frame-to-frame matching, we maintain a reference frame and compute the poses of the subsequent frames by localization with respect to the reference frame. First, we extract and match FAST corners [21] and BRIEF descriptors [6] from the left cameras of both the reference and current frames. Here, the reference frame includes the image from the stereo left camera, and disparity and depth maps from the stereo processes. The current frame consists of only the image from the stereo left camera. The 3D points are computed from the depth map of the reference frame. Next, we get the 2D-3D correspondences of the current frame to the reference frame from their 2D-2D matches of the image features. Lastly, we compute the current pose with a RANSAC PnP [10] followed by a non-linear refinement with robust cost function [13].

We select a new reference frame based on the following conditions:

- The number of geometric inliers of the feature correspondences has to be above a threshold value.
- Either the Euclidean or angular distance measured between the existing reference frame and the current frame exceeds a threshold value.

In cases where there are no sufficient matches, we replace the pose estimation with the optical flow and IMU readings. If this persists over a certain period of time, we replace the reference frame with the current frame. The advantage of using reference frames as compared to consecutive frame-to-frame matching is that it is less susceptible to drift. This is because we compute the current pose by localizing the current frame directly on the reference frame instead of concatenating frame-to-frame relative pose estimates which increases the chance for errors to accumulate. In particular, our reference frame visual odometry helps to stabilize the quadrotor during hovering by eliminating any possible drifts.

V. Autonomous Mapping and Exploration

Figure 4(a) shows the sub-components of the AI planner for dense environments where frontiers are clearly defined. Autonomous selection of the next waypoint is done by the frontier-based exploration which makes the selection based on a 2D slice of the existing occupancy grid mapping chosen at a fixed altitude. The quadrotor moves to the next waypoint via the VFH+ algorithm. Figure 4(b) shows the sub-components of the AI planner for sparse environments where frontiers are absent. In this case, the Bug algorithm for wall-following replaces the frontier-based exploration and VFH+ algorithm, and generates the next waypoint at a fixed altitude. The selection of the exploration algorithm is done manually in our current implementation.

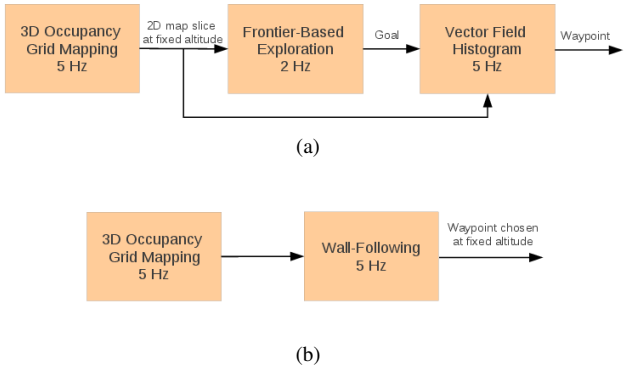


Fig. 4. AI planner for autonomous mapping and exploration in (a) dense and (b) sparse environments.

A. Mapping

The same mapping algorithm is used in both dense and sparse environments to generate a 3D occupancy grid map. Since the point cloud from a stereo frame potentially contains up to ∼350000 points, it is inefficient to directly update a 3D occupancy map with the point cloud. At the same time, the point cloud contains a considerable number of outliers. Instead, we downsample the point cloud to a virtual scan as described in [14], [15], removing many of the outliers in the process, before updating the 3D occupancy map via traversal of each ray in the virtual scan.

B. Exploration

1) Frontier-Based Exploration: We build on top of our visual odometry (Section IV-B) and mapping (Section V-A) algorithms the capability to autonomously explore and map an unknown environment. This essentially turns our mapping algorithm into active mapping [18] where waypoints are selected autonomously based on the exploration strategy. We use the frontier-based exploration [25] algorithm as our exploration strategy, and we set a fixed altitude for the MAV to fly at during the exploration process.

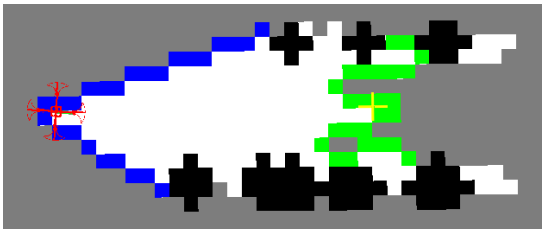


Fig. 5. 2D slice of initial occupancy map. Unknown cells are colored as grey, occupied cells are black, and free cells are white. The blue frontier behind the MAV is designated as the home frontier.

Exploration begins at an arbitrary location in an unknown environment. The frontier behind the MAV is designated as

the home frontier, and the MAV goes to as many frontiers as possible until it reaches the home frontier and lands at that frontier. Figure 5 shows the 2D slice of the initial occupancy map at the fixed altitude, and the home frontier colored in blue. Due to the use of a forward-looking stereo camera, the home frontier marks the left and right sides of the camera's field of view.

Initially, the MAV extracts a 2D slice of the 3D occupancy grid map based on its fixed altitude. Each grid cell in the map is classified as either occupied, free, or unknown based on its occupancy probability. A grid cell is labeled as a frontier cell if the cell is classified as free and has at least 1 neighbor classified as unknown. These frontier cells are then clustered into frontiers via connected-component labeling. Both frontiers whose cell count is below a threshold, and inaccessible frontiers are removed from the frontier set. A flood-fill algorithm is initialized at the MAV's current position, and frontiers which contain cells not labeled by the flood-fill algorithm are determined as inaccessible. The remaining frontiers are the regions where the MAV should fly to in order to maximize additional information about the environment. The centroid (C_x^j, C_y^j) of the j^{th} frontier in the map is computed as:

$$\begin{aligned} C_x^j &= \frac{1}{n}\sum_{i=0}^{n} x_i^j \\ C_y^j &= \frac{1}{n}\sum_{i=0}^{n} y_i^j \end{aligned} \tag{3}$$

The MAV selects the centroid that is closest to its current location as its next desired waypoint. The desired heading κ is set to be the mean of the heading of the frontier cells with respect to the centroid. The MAV then flies to the desired waypoint using VFH+, and along the way, the occupancy map expands based on successive stereo images. Once the MAV either reaches the desired frontier or cannot reach that frontier within a certain period of time, the MAV selects another frontier to go to. Exploration continues until the home frontier is the only frontier left in the map, and the MAV lands at the home frontier.

2) Wall-Following Exploration: We adopt a wall-following exploration strategy using the Bug algorithm [7] in sparse environments where the frontier-based exploration under-performs. The quadrotor starts at an arbitrary location near a wall. A 3D occupancy grid map described in Section V-A is built incrementally as the quadrotor circumvents the wall. An example of the 3D occupancy map is shown in Figure 6. We extract the wall plane from the 3D points from the stereo camera by doing iterative plane fittings with RANSAC. The plane fitting process is initialized with a randomly chosen set of points. A plane is taken as the wall plane if it is approximately parallel to the gravity vector from the IMU. The 3D points belonging to a plane which does not fulfill this criteria are discarded, and the selection process continues iteratively with the remaining 3D points until we find a suitable wall plane. Figure 7 shows an example of a wall detected (highlighted in blue) by the plane detection algorithm. The MAV has to select a desired waypoint once the wall is detected. This is done by first computing the point p_1 on the wall which is closest to the current position of the MAV. We then compute the point p_2 which is at a distance d away from p_1 along the wall plane at the same altitude as the MAV, and the next waypoint p_3 is chosen as the point in the perpendicular direction from the plane with a distance equal to the distance between p_1 and the MAV's current position. The choice of d depends on the admissible distance from one waypoint to another. The chosen waypoint is directly sent to the position controller of the quadrotor. Exploration with wall-following terminates at the command of the operator.

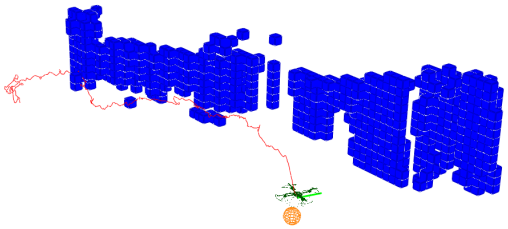


Fig. 6. 3D occupancy map built incrementally as the MAV was following a wall. The map resolution is 0.1 m.

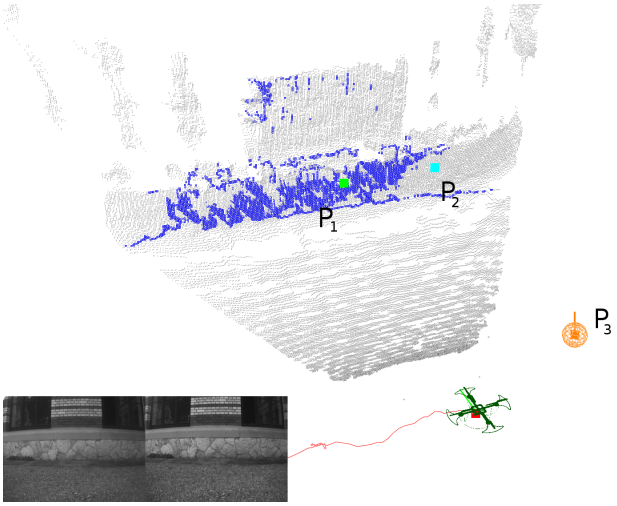


Fig. 7. Bug algorithm detecting and following a wall in a sparse environment.

C. Local Planning

The VFH+ plans the path to the desired frontier if the frontier-based exploration algorithm was selected. The VFH+ planner reads in pose and occupancy map updates; at each iteration, the planner constructs a polar histogram centered at the MAV's current position, and expands each cell classified as occupied in the occupancy map by the sum of the MAV's width and safety clearance distance. The score of each histogram sector is based on the distance between the MAV's position and the obstacle cells falling in the histogram sector. The scores of all histogram sectors are thresholded to yield candidate vector directions. Each vector direction

has a score equal to the weighted sum of three angular differences: between that vector direction and the MAV's current yaw, between that vector direction and the direction towards the desired goal, and between that vector direction and the previously chosen direction. This scoring method facilitates a smooth trajectory. The vector direction with the lowest score is chosen as the direction for the MAV to travel in. A set-point at a preset distance from the MAV along the vector direction is sent to the position controller.

In figure 8, the exploration module has selected the desired waypoint to be at the upper right of the image along the corridor. The figure shows the cells in the occupancy map classified as occupied and colored from blue to red in order of increasing height. The thresholded polar histogram is visualized as a circle around the MAV. The green sectors represent the set of feasible vector directions, while the red sectors represent the set of infeasible vector directions due to the directions' proximity to nearby obstacles. The lines radiating out from the MAV represent the candidate vector directions. The lines are colored according to their score: blue indicates a low score while red indicates a high score. The set-point represented as an orange sphere indicates the set-point along the desired vector direction.

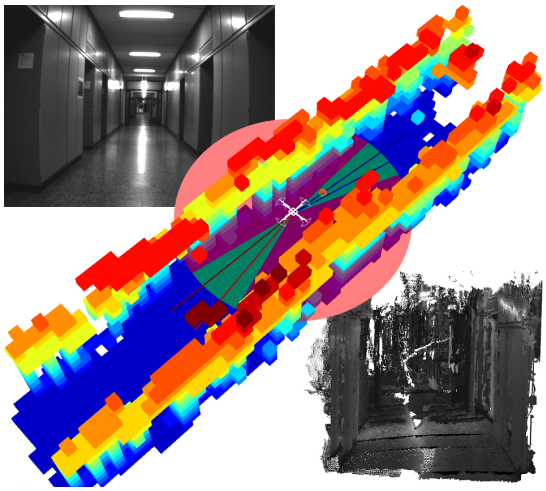


Fig. 8. Visualization of obstacle map and VFH+ planning along a corridor. The left stereo image on the top left shows the corridor and the bottom right shows the textured map in 3D. The map resolution is 0.25 m.

VI. Off-Board Visual SLAM

A. Visual SLAM

We use the g2o [16] framework for our off-board visual pose-graph SLAM. Figure 9 shows an example of the hypergraph representation of our visual pose-graph SLAM. The node of the graph represents the position X_t of the MAV in the inertial frame at time t. The edges of the graph represents the constraints between the poses. These constraints are measurements from the visual odometry $z^{vo}_{t,t+1}$, optical flow $z^{of}_{t,t+1}$ and loop-closure constraint $z^{loop}_{t,t+5}$. The visual odometry constraint $z^{vo}_{t,t+1}$ is the transformation between two successive poses X_t and X_{t+1} estimated by visual odometry. The optical flow constraint $z^{of}_{t,t+1}$ is the transformation between two successive poses X_t and X_{t+1} estimated by the optical flow sensor. The loop-closure constraint $z^{loop}_{t,t+5}$ is the transformation between matching views X_t and X_{t+5} that have been identified by the loop-closure detection step. Additionally, the measurement from the ultrasonic sensor z^{alt}_t is used as prior estimate for the pose-graph SLAM. The maximum-likelihood estimates of the MAV positions are computed by minimizing the Euclidean distances between the transformations. The non-linear optimization is done by sparse Cholesky decomposition using the g2o framework.

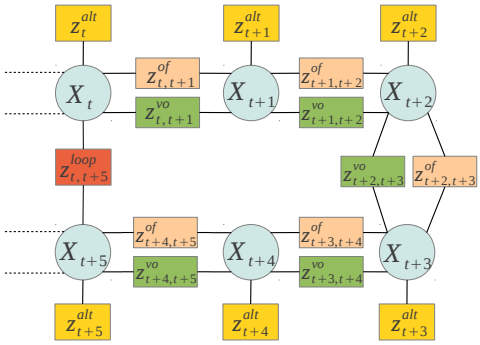


Fig. 9. g2o hypergraph representation of our pose-graph SLAM problem.

B. Loop-Detection and Relocalization

We remove drifts in the visual odometry and enforce global consistency on the pose estimations with loop-detections and loop-closures. In our system, loop-detection is done with the vocabulary tree [11], [20] approach. We extract the SURF features and descriptors [3] for every image and quantize them into visual words using a vocabulary tree which was pre-trained on a general image data set. The visual words and image IDs are stored in a database which is organized as an inverted file for efficient insertion and retrieval. The global poses of the MAV are also added as meta-data into the database. We rank all existing images in the database according to visual similarity with every new incoming image. The n best candidates go through a geometric verification where we match the SURF features, compute the relative pose between the image frames with RANSAC PnP and check for the feature inliers. A candidate image passes the geometric verification if the inlier count is above a preset threshold. The transformation between the current image and the candidate image that passes the geometric verification test is computed and added to the pose-graph as constraint. Node X_t from Figure 9 is an example of a candidate image that passes the geometric verification with the current image $X_{t,t+5}$. $z^{loop}_{t,t+5}$ is the loop constraint between the two nodes. We compute the transformation for loop constraint with the RANSAC PnP that was mentioned earlier in Section IV-B.

Additionally, we also implement a relocalization algorithm that runs on our on-board computer at a framerate of 0.5 Hz. The relocalization algorithm is particularly useful when a pre-built map is available. In this case, the MAV can perform

relocalization on-board with the pre-built map instead of the computationally expensive SLAM process. Pre-built maps of the environment could be provided by a different MAV/robot or the same MAV that has previously done SLAM on the environment. Relocalization is built on top of the visual loop-detection. The current camera image is checked against the database and the most similar image in the database is used to compute the global position of the current camera image with similar methods described in the previous paragraph.

VII. Experimental Results

A. Frontier-Based Exploration

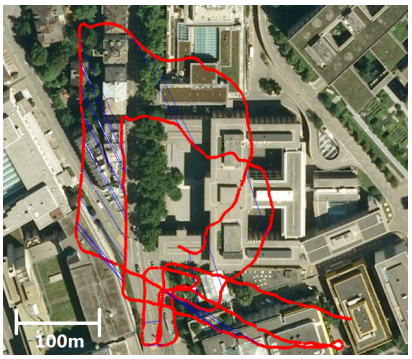
We test the automonous frontier-based exploration of our quadrotor in both indoor and outdoor environments. However, we only show the results of the indoor exploration in the paper due to space limitation. For outdoor results, please refer to the video at `http://www.inf.ethz.ch/~hengli/exploration.avi`. Our quadrotor automonously explores a stretch of corridor which is approximately 30 m long for the indoor test. Figure 10 shows the occupancy maps generated during the exploration. Figure 10(a) shows the quadrotor at the starting position. It builds a local occupancy map in which two frontiers (represented by green and blue cells) are detected. The quadrotor selects the nearest frontier (represented by a red dot) and reaches for it using the VFH+ algorithm. Once the quadrotor reaches its destination, it retrieves the latest occupancy map, and chooses the next frontier to go to as shown in Figure 10(b). The frontier selection and mapping processes are repeated until it is manually terminated in Figure 10(c).

B. Visual SLAM

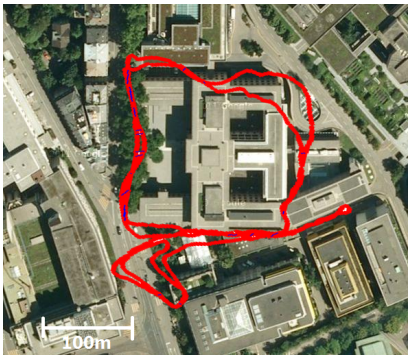
We test the scalability of our pose-graph SLAM with a large outdoor dataset that consists of more than 5000 stereo image pairs which spans across a total trajectory of approximately 1 km. The dataset contains two large and two small nested loops, and it also contains passing cars, bicyclists and pedestrians. As the dataset was collected along urban streets, we manually carry the quadrotor around instead of flying it to prevent any accidental injuries to the pedestrians. We took a log of the dataset with the MAV and processed it off-board. Figure 11(a) shows the raw pose-graph of the trajectory, which is estimated from visual odometry and optical flow, overlaid on the satellite image. The blue lines are loop-closure opportunities detected by the vocabulary tree. It is clear from the figure that the raw pose-graph is not correct. Figure 11(b) shows the pose-graph after optimization overlaid on the satellite image. Since the loop-closure constraints are taken into account, the result is much accurate.

C. Relocalization

We test our relocalization algorithm, which was mentioned earlier in Section VI-B, in a laboratory setting where a Vicon motion capture system is available for ground truth. We first collect a set of images of the laboratory from manual flight of the quadrotor. This set of images are then used to create a map using our visual SLAM algorithm. We specify several waypoints within the created map for the quadrotor to follow. Figure 12 shows the x and y positions from a part of the flight compared to the Vicon ground truth and desired control positions. We can see from the plots that our relocalization algorithm is sufficiently close to the Vicon ground truth.

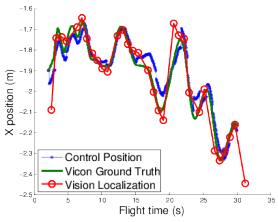


(a)

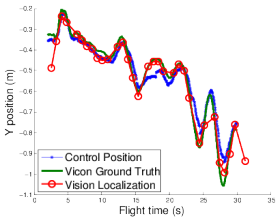


(b)

Fig. 11. Large scale pose-graph before and after optimization.



(a)

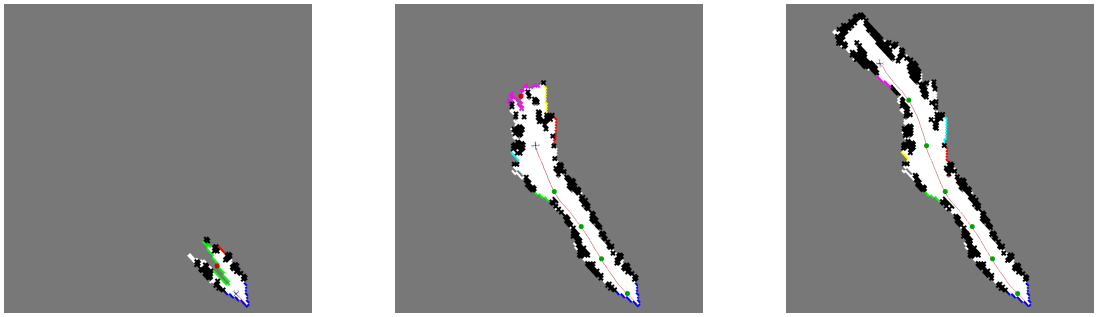


(b)

Fig. 12. Comparison of the results from our relocalization algorithm with Vicon ground truth and desired control positions.

VIII. Conclusion

The advantages of using cameras as the main sensor for MAVs are the primary motivation for this work. We have showed with our quadrotor MAV system equipped with a stereo camera as its main sensor, the feasibility of doing pose estimation, autonomous mapping and exploration on-board. In comparison with other existing approaches that used vision, we are able to do pose estimation, autonomous



(a) Exploration at t_1 (b) Exploration at t_2 (c) Exploration at t_3

Fig. 10. Various stages of indoor exploration.

mapping and exploration on-board because of our exclusive combination of a front-looking stereo camera and a downward-looking camera running the optical flow algorithm. In addition, we showed large-scale pose-graph SLAM that was done off-board. Experimental results that verified our approaches are shown. It should be emphasized that despite the nice results that we have shown for vision-based autonomous mapping and exploration of the quadrotor, many other challenges, such as detecting sufficient image features and lighting etc, associated with vision-based autonomous quadrotor still remain. All these challenges remain as future works.

IX. Acknowledgement

This work is supported in part by the European Communitys Seventh Framework Programme (FP7/2007-2013) and by the Swiss National Science Foundation (SNF) under grant #200020-135050. Lionel Heng is funded by the DSO National Laboratories Postgraduate Scholarship.

References

[1] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart. Onboard imu and monocular vision based control for mavs in unknown in- and outdoor environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[2] A. Bachrach, A. de Winter, Ruijie He, G. Hemann, S. Prentice, and N. Roy. Range - robust autonomous navigation in gps-denied environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1096 –1097, may 2010.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). In *Computer Vision and Image Understanding*, Jan 2008.

[4] C. Bills, J. Chen, and A. Saxena. Autonomous mav flight in indoor environments using single image perspective cues. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 5776–5783, 2011.

[5] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart. Vision based mav navigation in unknown and unstructured environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2010.

[6] M. Calonder, V. Lepetit, and C. Strecha.... Brief: binary robust independent elementary features. In *Proceedings of ECCV 2010*, Jan 2010.

[7] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.

[8] I. Dryanovski, W. Morris, and J. Xiao. An open-source pose estimation system for micro-air vehicles. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 4449–4454, 2011.

[9] D. Eberli, D. Scaramuzza, S. Weiss, and R. Siegwart. Vision based position control for mavs using one single circular landmark. In *Journal of Intelligent and Robotic Systems*, volume 61, pages 495–512, 2011.

[10] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Communications of the ACM*, Jan 1981.

[11] F. Fraundorfer, C. Engels, and D. Nistér. Topological mapping, localization and navigation using image collections. In *IEEE/RSJ Conference on Intelligent Robots and Systems*, volume 1. IEEE, 2007.

[12] S. Grzonka, G. Grisetti, and W. Burgard. Towards a navigation system for autonomous indoor flying. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Kobe, Japan, 2009.

[13] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge, 2000.

[14] L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys. Real-time photo-realistic 3d mapping for micro aerial vehicles. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[15] L. Heng, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. Autonomous obstacle avoidance and maneuvering on a vision-guided mav using on-board processing. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2472–2477, 2011.

[16] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[17] M. Lorenz, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys. Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Auton. Robots*, 33(1-2):21–39, 2012.

[18] A. A. Makarenko, S. B. Williams, F. Bourgault, and H. F. Durrant-Whyte. An experiment in integrated exploration. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 534–539, 2002.

[19] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2992–2997, 2011.

[20] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, New York City, New York*, pages 2161–2168, 2006.

[21] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume 32, pages 105–119, 2010.

[22] S. Shen, N. Michael, and V. Kumar. Autonomous multi-floor indoor navigation with a computationally constrained mav. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 20–25, 2011.

[23] I. Ulrich and J. Borenstein. Vfh+: Reliable obstacle avoidance for fast mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1998.

[24] D. Wagner and D. Schmalstieg. Artoolkitplus for pose tracking on mobile devices. In *Proc. of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2007.

[25] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. of the IEEE International Symposium on Computational Intelligence, Robotics and Automation*, pages 146–151, 1997.