Vision Based MAV Navigation in Unknown and Unstructured Environments

Michael Blösch and Davide Scaramuzza, Member, IEEE and Stephan Weiss, Member, IEEE and Roland Siegwart, Fellow, IEEE

Abstract—Within the research on micro aerial vehicles, the field on flight control and autonomous mission execution is one of the most active. A crucial point is the localization of the vehicle, which is especially difficult in unknown, GPS-denied environments. This paper presents a novel vision based approach, where the vehicle is localized via a downward looking monocular camera. A visual SLAM algorithm tracks the pose of the camera, while, simultaneously, building an incremental map of the surrounding region. Based on this pose estimation a LQG/LTR based controller stabilizes the vehicle at a desired setpoint, rendering simple maneuvers possible like take off, hovering, setpoint following or landing. Experimental data shows that this approach efficiently controls a helicopter while navigating through an unknown and unstructured environment. To the best of our knowledge this represents the first MAV platform able to navigate through an unexplored environment without any time-drift and independent of any external aid like GPS or artificial beacons.

Index Terms—MAV Navigation, Visual SLAM, LQG/LTR Controller, VTOL.

I. INTRODUCTION AND RELATED WORK

In the past years, micro aerial vehicles (MAVs) strongly gained in autonomy. This was strongly motivated through the very wide field of applications for these little engines. Imaginable missions can vary from the inspection of static structure for eventual critical spots over to the exploration of unknown and dangerous environments or the searching and rescuing of persons in emergency situations. Commonly associated keywords are: search and rescue, exploration, surveillance, agriculture and inspection.

Because such MAVs are in general highly unstable and nonlinear systems, a clever combination of sensor equipment and controller must be designed. Most of the approaches model the MAV as two connected ideal subsystems and use a cascaded control structure: one controller for the attitude (3D orientation of the helicopter) of the MAV and one superposed controller for its 3D position. In general the attitude controller must be faster than the position controller. Often a simple PD-controller is enough for that purpose, but also other techniques can be applied for its design [1] [2] [3]. E.g. Bouabdallah et al. [4] analyzed the application of two different control techniques "Sliding-Mode" and "Backstepping" and especially showed that the later has very good stabilizing qualities. In our case, we use the on-board attitude controller provided by Ascending Technologies [5], which is basically a PD-controller.

The use of accelerometers, gyros and compasses can lead to strong attitude controller and enables the stabilization of the orientation of the MAV. Although it is possible to hold in a hovering state, there is no possibility to perceive any drift caused by accumulated error. To tackle this issue, exteroceptive sensors (e.g. laser rangers, cameras, GPS, pressure sensors or ultrasonic sensors) able to measure the position must be used. The most common approach is to mount a DGPS receiver on the MAV. By using a so called INS/GPS approach, where IMU data and GPS data are fused together, the position can be estimated up to a precision of some decimeters. Thus the MAV can be fully stabilized and controlled [6] [7]. Two drawbacks of this approach are the necessity to receive any GPS signal and the lack of precision of the position estimate.

An alternative approach is to use cameras for the localization task. There are several advantages that can be mentioned: cameras are lightweight bearing sensors with low power consumption and are relatively cheap to buy. Also, they provide very rich information on the environment. However, it is exactly this property that is also one of the greatest challenges when applying cameras: in order to obtain a precise and consistent pose estimation the vast amount of data has to be interpreted in a correct manner. The most simple way is to install a number of external cameras with known location and to have them track the MAV [8] [9] [10]. This method is very efficient for testing purposes and can be used to evaluate other approaches as ground truth reference. However it is not suitable for missions where the installation of an appropriate infrastructure is not feasible.

This approach can also be implemented the other way round: the camera is mounted on the helicopter and tracks a known pattern on the ground [11]. The team of Hamel [12] implemented a visual servo trajectory tracking to control an UAV with a mounted camera observing $n$ fixed points. Nice methods have also been developed by fusing the visual data with IMU data [13].

The availability of an on-board camera can offer new possibilities. Templeton et al. [14] used a vision-based terrain mapping algorithm to estimate the 3D structure of the environment in order to find adequate landing sites (the flight control system still uses GPS data). The problem of autonomously landing a MAV on a known landing platform using vision has been solved already quite early by Saripalli et al. [15]. A vision-based forced landing algorithm has been implemented where a MAV has to localize a good landing area and reach it as fast and safely as possible [16]. Another possibility is to have a MAV tracking a leading MAV with a fixed relative position and orientation. This has been implemented by Chen et al. by constructing an Euclidean homography based on some feature points on the leading vehicle [17].
Alternatively, stabilizing controllers can be built by means of optical flow considerations. Herisse et al. [18] use an optical flow based PI-controller to stabilize a hovering MAV, they also implemented an automatic landing routine by contemplating the divergent optical flow. Hrabar et al. [19] developed a platform able to navigate through urban canyons. It was based on the analysis of the optical flow on both sides of the vehicle. Also, by having a forward looking stereo camera, they were able to avoid oncoming obstacles.

Based on optical flow some biologically inspired control algorithms have been developed for MAV stabilization [20] [21]. However, the optical flow based pose estimation is also affected by slow drift as it does observe the relative velocity of features only. This counts also for visual odometry based implementations, where the drift is estimated by considering the feature displacements between two successive images [22].

A very similar approach to ours was implemented by Ahrens et al. [23]. Based on the SLAM algorithm of Davison et al. [24], they build a localization and mapping framework that is able to provide an almost drift-free pose estimation. With that they implemented a very efficient position controller and obstacle avoidance framework. However, due to the simplification they used in their feature tracking algorithm a non-negligible drift persists. Also, to our knowledge, they kept using an external Vicon localization system to control the aerial vehicle (a system of external cameras that tracks the 3D pose of the vehicle). The output of the SLAM-based localization system was not used for the vehicle control so far.

In this paper, we present an approach based on the visual SLAM algorithm of Klein et al. [25]. It enables the MAV to autonomously determine its location and consequently stabilize itself. In contrast to other approaches we do not require any a priori information on the environment or any known pattern in order to obtain a time-drift-free MAV control. The controller is based on a cascaded structure and is designed by means of the discrete LQG/LTR procedure applied on a simplified MAV model. This enables us to handle the considerable time delay that comes from the image processing and from the SLAM algorithm.

For the experimental tests a downward looking camera is mounted on the Hummingbird quadrotor from Ascending Technologies [5]. The images are fed to the SLAM algorithm which is currently running on a fixed ground station. Based on the position estimate the control input are computed and then sent back to the quadrotor. To the best of our knowledge, this is the first implementation of a vision-based MAV controller that can be used in an unknown environment without the aid of any infrastructure based localization system, any beacons, artificial features, or any prior knowledge on the environment. In other words, our platform does not need any external assistance in order to navigate through an unexplored region without any time-drifting position control.

The outline of the paper is as follows: after introducing some notations in section II we will shortly summarize the SLAM algorithm that has been used here and explain why we chose it for our approach (section III). In section IV we will take a look at the modeling of the system and the parameter identification. After that we will discuss the controller design (section V) and analyze the entire structure of our approach (section VI). To the end we will have a look at the achieved results and discuss them (section VII).

II. NOTATIONS

To facilitate the following considerations we will introduce some notations. We will always use boldface for vectors.

Common notations:

- \( \mathbf{v} \): Vector \( v \) expressed in the \( A \) coordinate system.
- \( R_{AB} \): Rotation matrix from coordinate system \( B \) to coordinate system \( A \).

Coordinate systems:

- \( I \): Inertial coordinate system, is chosen so that the gravity lies along the \( z \)-axis.
- \( M \): Coordinate system of the map of the SLAM algorithm.
- \( C \): Coordinate system of the camera frame.
- \( H \): Coordinate system of the Helicopter.

Vectors and scalars:

- \( \mathbf{r} \): Position vector of the helicopter.
- \( \mathbf{T} \): Thrust vector of the helicopter (always lies on the \( z \)-axis of the \( H \) coordinate frame).
- \( |T| \): The absolute value of the thrust vector \( T \).
- \( \varphi \): Roll angle of the helicopter, rotation around the \( x \)-axis of the \( I \) coordinate system.
- \( \theta \): Pitch angle of the helicopter, rotation around the \( y \)-axis of the \( I \) coordinate system.
- \( \psi \): Yaw angle of the helicopter, rotation around the \( z \)-axis of the \( I \) coordinate system.
- \( \omega \): Rotational speed around the \( z \)-axis of the \( I \) coordinate system.

Constant parameters:

- \( \mathbf{F}_G \): Gravitational force.
- \( g \): Gravitational acceleration.
- \( m \): Mass of the helicopter.

Please note that we use the Tait-Bryan convention for the Euler decomposition of the rotation matrix \( R_{HI} \) into the 3 angles \( \varphi \), \( \theta \) and \( \psi \). If the angles represent rotations between two other coordinates frames than \( I \) and \( H \), we specify them in the index, e.g. \( \psi_{CM} \) represents the rotation around the \( z \)-axis from the map coordinate frame to the camera coordinate frame. All coordinate frames have the same invariant origin.

Estimated values are denoted by an additional tilde (e.g. \( \tilde{\mathbf{r}} \)). Reference values are denoted with a star (e.g. \( T^* \)).

III. VISUAL SLAM BASED LOCALIZATION

A. Description of the Visual SLAM algorithm

The presented approach uses the visual SLAM algorithm of Klein et al. [25] in order to localize the MAV from a single camera (see Fig. 1). In summary, they split the simultaneous localization and mapping task into two separately-scheduled threads: the tracking thread and the mapping thread.
that both threads can run at different frequencies. Thus the analysis of the SLAM algorithm

smaller loops. In order to recognize larger loops an additional

is able to efficiently track the point features and to close

of Georg Klein especially accurate while expanding the map.

of features that are being tracked and the parallel application

error between the observed point features and the projection

of the current camera image, i.e., it compares the extracted point features with the stored map and thereby attempts to determine the position of the camera. This is done with the following steps: first a simple motion model is applied to predict the new pose of the camera. Then the stored map points are projected into the camera frame and corresponding features (FAST corners) are searched, this is also called the data association procedure. When this is done, the algorithm refines the orientation and position of the camera such that the total error between the observed point features and the projection of the map points into the actual frame is minimized.

The Mapping thread uses a subset of all camera images (also called keyframes) to build a 3D point map of the surroundings. The keyframes are selected using some heuristic criteria. After that a batch optimization is applied on the joint state of map points and keyframe poses. This attempts to minimize the total error between projected map points and the corresponding observations in the keyframes. In the computer vision community this procedure is also called bundle adjustment. To optimize this thread the bundle adjustment is alternately applied on the global and local set of map points and keyframes.

There are several important differences that can be mentioned in comparison to the standard SLAM algorithm of Davison et al. [24]. First of all it does not use any EKF-based state estimation and does not consider any uncertainties, be it for the pose of the camera or for the location of the features. This does spare a lot of computational effort that would occur with the processing of the corresponding data. Considering the uncertainty of the state can ease the data association process and enable larger loop closing. However, the very large number of features that are being tracked and the parallel application of bundle adjustment procedure render the SLAM algorithm of Georg Klein especially accurate while expanding the map. Therefore, using a fixed area for feature matches, the algorithm is able to efficiently track the point features and to close smaller loops. In order to recognize larger loops an additional loop closing could be implemented.

B. Analysis of the SLAM Algorithm

The main advantage of the thread splitting lies therein that both threads can run at different frequencies. Thus the mapping thread is able to apply much more powerful and time-consuming algorithm to build up its map. Simultaneously, the tracker can estimate the camera position at a higher frequency. This does strongly improve the performance. Compared to an incremental map making, the algorithm of Klein et al. does spare a lot of computational effort in that it does not process every single image, which often contains a lot of redundant information. For example, when the camera is moving very slowly or if it stays at the same position, the mapping thread requires only very little power. This is one of the main reasons why we choose this SLAM algorithm.

When moving the helicopter through a region, our camera is facing more or less straight downwards all the time. This increases the overlapping image portion of neighboring keyframes, so that we can even further decrease the speed at which keyframes are added to the map. Also, once the MAV has explored a certain region, no more new keyframes will be added within the region boundaries and the computation time remains constant. On the other hand, when exploring new areas the global bundle adjustment can be very expensive, limiting the number of keyframes to a few hundred on our platform (around 50-100 m²).

Another strength of the SLAM algorithm is its robustness against partial camera occlusion. If a sufficient part (around 50%) of the point features can still be tracked the pose estimate is accurate enough to sustain stable MAV control. Also, the algorithm will avoid to add any keyframes in such situation so as not to corrupt the map.

The algorithm does not estimate any uncertainties, neither for the camera poses nor for the map point locations. This reduces the ability of detecting loop closures which is an important quality of a SLAM algorithm. Loop closure events, where the SLAM algorithm successfully re-observes a previously visited area, strongly improve the quality and consistency of the map. However, the algorithm of Klein et al. increments the map very accurately and accumulates only little error during expansion. Consequently, it is naturally able to handle simple loop closures by just re-observing known map points in new keyframes.

An intricate hurdle when using a monocular camera is the lack of any depth information. Because of that the algorithm must initialize new points based on the observations from
more than one keyframe. This could motivate the use of a stereo camera. However, for a stereo camera to bring any further advantage, the observed scene must be within some range of the stereo camera, otherwise a single camera will yield the same utility. Closely linked to this problem is the unobservability of the map scale, to tackle this we are forced to estimate the map scale by hand and pass it to the controller. Using an IMU, we are currently implementing an online scale estimation to tackle this problem [26].

C. Adaptations to the SLAM Algorithm

We adapt some parameters of the SLAM algorithm to increase its performance within our framework. First we use a more conservative keyframe selecting heuristic in order to decrease the number of keyframes added during map expansion. Thereby we are able to map much larger areas before reaching the computational limit. Additionally, we reduce the number of points being tracked by the tracking thread from 1000 to 200. This again increases the maximal map size and the frame rate, while keeping the accurate tracking quality and the robustness against occlusion. Last, in order to increase the tracking quality when hovering over very uneven environments, we increase the depth range at which new map points are added.

A point to note for the later controller design is the speed of the SLAM algorithm. On our infrastructure we can observe varying time steps between the outputs of the pose estimates. Depending on the actual state of the mapping thread, 10 to 40 pose estimations can be obtained per seconds. This yields us a Nyquist frequency of around 5 Hz (worst case).

To increase the autonomy of the system we write a routine that can store and load maps. With this we are able to overcome the initialization of the map during which no position estimate is available. It provides the possibility to start the helicopter from a small known patch and to skip the initialization process. Later the loaded map of the patch can be expanded by exploring the environment.

IV. MODELING AND PARAMETER IDENTIFICATION

As our MAV platform we choose the Hummingbird quadrotor from AscTec [5]. A wide arsenal of sensors is mounted on it. This includes 3 gyros, a 3D compass and an accelerometer. At this point it is important to mention that an efficient attitude controller is implemented on the onboard microcontroller of the helicopter, what permits us to focus on the design of a controller for the stabilization of the x,y,z positions coordinates and the yaw angle.

We produce a model of the system and use the reference values of the attitude controller as the control inputs. The output is the pose of the camera, i.e., the 3 dimensional position and orientation of the camera in the coordinate frame of the stored map. Thus the dynamics of the internal attitude controller must be included in the model.

The attitude controller is designed in order to control the two tilt angles $\varphi$, $\theta$, the angular velocity around the vertical axis $\omega$ and the total thrust $T$ of the helicopter. Therefore the corresponding reference values of the attitude controller (denoted by $\varphi^*$, $\theta^*$, $T^*$ and $\omega^*$) are the inputs to the model, while the outputs are the estimation of the helicopter position $M_r$ and the yaw angle $\psi_{CM}$ computed by decomposing the rotation matrix $R_{CM}$. $M_r$ and $R_{CM}$ are obtained through the SLAM algorithm and can also be transformed to the $I$ coordinate frame.

The helicopter is modeled as a simple point mass on which we apply the principle of linear momentum (Newton’s second law). The forces that are acting on the helicopter are reduced to the thrust force $T$ aligned with the z-axis of the helicopter and the gravitational force $F_G$ pointing towards the positive z-axis of the inertial coordinate system (see Fig. 2). We can now project $F_G$ onto the x,y,z axes and apply the principle of linear momentum onto the three directions of the inertial coordinate frame. This yields the following matrix equation:

\[
\dot{\mathbf{r}} = R_{1M} \cdot M_r = \frac{1}{m} R_{H1}^{-1}(\varphi, \theta, \psi) \begin{pmatrix} 0 \\ 0 \\ -T \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix}
\]

Now the state $M_r$ can be computed given the three angles $\varphi$, $\theta$, $\psi$ and the thrust value $T$. For the yaw angle $\psi$ and the thrust value $T$ we just assume the following simple relations:

\[
\dot{\psi} = \omega^*, \quad T = T^*
\]

Note that it is important that the z-axis of the inertial frame points toward the center of gravity. Otherwise the gravity vector is added in the wrong direction. Before flying this has to be done correctly. At the moment this is done manually.

As the attitude controller dynamics from the inputs $\varphi^*$ and $\theta^*$ to the angles $\varphi$ and $\theta$ is quite fast, we model them as two separated second-order systems with the following transfer function:

\[
T(s) = \frac{\omega^2}{s^2 + 2 \cdot d \cdot \omega \cdot s + \omega^2}
\]
We then identify the parameters $d$ and $\omega$ on the plant by analyzing the step response (see Fig. 3). This yields a value of 15.92 rad/s for $\omega$ and 1.22 for $d$.

For the subsequent controller design we need to estimate the time delay $T_d$ in the control loop, this was done by observing the same step response as before. We assume that the time delay is mainly caused by the data transmission and SLAM algorithm, so that its value is the same for all outputs. Depending on the data transmission method the delay varies between 80.6 ms (USB cable) to 250 ms (Wi-Fi n-standard).

All in all, the system can be represented like in Fig. 4. Please note that for the sake of simplicity we do not model any disturbances or noise.

V. CONTROLLER DESIGN

As already mentioned, the design of the position controller is based on a plant model where the dynamics of the attitude controller are included. Unfortunately, the control inputs introduce strong non-linearities into the system as can be seen in equation (1). Using the Tait-Bryan convention the following equation for the force acting on the helicopter are derived:

$$
\begin{pmatrix}
F_x \\
F_y \\
F_z
\end{pmatrix} = -T
\begin{pmatrix}
\cos \psi \sin \theta \cos \varphi + \sin \psi \sin \varphi \\
\sin \psi \sin \theta \cos \varphi - \cos \psi \sin \varphi \\
\cos \theta \cos \varphi
\end{pmatrix}
+ \begin{pmatrix}
0 \\
0 \\
mg
\end{pmatrix}
$$

By solving this equation for $\varphi$, $\theta$ and $T$ we can write the following transformation of the control inputs:

$$
\theta^* = \arctan \left( \frac{\cos \psi \cdot 1 \cdot F^*_x + \sin \tilde{\psi} \cdot 1 \cdot F^*_y}{1 \cdot F^*_z - mg} \right) 
$$

$$
\varphi^* = \arctan \left( \frac{\sin \tilde{\psi} \cdot 1 \cdot F^*_x - \cos \tilde{\psi} \cdot 1 \cdot F^*_y}{1 \cdot F^*_z - mg} \cos \varphi^* \right) 
$$

$$
T^* = \frac{mg - 1 \cdot F^*_z}{\cos \theta^* \cos \varphi^*}
$$

Assuming that the attitude control’s dynamic is fast and smooth enough, the second-order system block (attitude controller dynamics) and the control input transformations can be exchanged in order to obtain a new plant with the input $1 \cdot F^*_x$, $1 \cdot F^*_y$, $1 \cdot F^*_z$ and $\omega^*$. This yields four decoupled linear systems that can be controlled separately (see Fig. 5).

Because the subsystem for the yaw angle is simple and can be easily stabilized a simple design will be sufficient: it outputs a constant angular velocity $\omega^*$ when the angular error exceeds a certain value. Thereby large angular velocities that could destabilize the position of the helicopter can be avoided.

The position controllers are designed by means of the discrete LQG/LTR approach. The procedure is identical for all three position values, except that for the $z$-coordinate the second-order system is left out. Because of the limited computational power the constant controller frequency is kept at roughly 20 Hz. This approximately matches the frequency of the SLAM pose estimates (around 15-30Hz). For the Nyquist frequency we take 7.5 Hz (half of the measurement frequency). It is possible that the measurement frequency falls awhile beneath 15 Hz, but this generally does not last longer than some 100 ms.

The discrete system model is derived via the zero-order hold transformation of the continuous time model including the second-order system of the internal controller dynamics and the momentum law (double integration). After that, the time delay, approximated by a multiple of the sampling time, is added at the output of the model. Supplementary, due to varying battery power and tilt angles calibrations some integrating action has to be introduced in all position coordinates.
by expanding the system with an output error integrating part. Now the corresponding system matrices \( F, G, C, D \) required for the LQG/LTR can be computed. Applying the LQG/LTR procedure with feed-forward action [27] yields the structure in Fig. 6.

The Nyquist frequency (around 7.5 Hz) is strongly limiting the tuning process of the LQG/LTR. The resulting closed loop system has its poles like in Fig. 7. Except for the four poles induced by the time delay, all poles lie within 1-32 rad/s. No pole has a damping lower than 0.5. A zero situated at -7.2 is not displayed on the plot.

VI. FINAL SYSTEM STRUCTURE, FINAL IMPLEMENTATION

We use the Hummingbird quadrotor platform from Ascending Technologies [5]. Equipped with numerous sensors, an onboard 8-bit microcontroller fuses all data in order to obtain an estimate for both tilt angles and the angular velocity around the vertical axis. Based on these estimates a high performance onboard controller is able to stabilize the tilt angles and the yaw rate at desired reference values. These can be sent by an external operator via an XBee digital radio.

Beneath the quadrotor a 12g USB uEye UI-122xLE is installed which gathers 752x480 images with global shutter. At the moment the images are transmitted through an USB cable linked to the ground station. The computations on the ground stations are done on a Intel Core 2 CPU 2x2GHz processor. All code is implemented in C++. 
In the flow diagram (see Fig. 10) the entire closed-loop system is represented. The SLAM algorithm and the controller are both implemented on the ground station. Based on different flight phases the quadrotor is able to land or take off from the ground. As the vision based localization does not work when the helicopter is landed (the camera is too near to the floor), the take off is only feasible if the initial land-patch beneath it is already stored in the map. Giving increasing thrust the helicopter can then fly blindly until it re-finds the map and stabilizes itself (the position can be tracked from a height of ca 15 cm).

The scale problem, arising from the bearing-only information of the monocular camera, is solved by adjusting it manually at the begin of the flight. However, due to possible scale-drift in the map its value could change and destabilize the MAV. Therefore, we are currently working on a framework where an online scale estimation is included [26].

It is also possible to observe some rotational drift in the map, so that the inertial coordinate frame is not more correctly aligned. Rotations around the z-axis can be adjusted by the yaw controller. More problematic are the rotations around the x-axis and y-axis. Because the error between the reference value and the actual position is no more correctly decomposed into its x,y,z-values, this can lead to instability if not considered. At the moment this problem is solved by re-aligning the inertial coordinate frame every 40 cm. For that the helicopter has to be stabilized until its pose is approximately horizontal. This is done by observing the RMS value of the last 30 position errors (around 1.5 s). When this value is beneath a certain threshold (0.06 m) we can assume that the pose is horizontal (±0.02 rad in the tilt angles). This also leaves the mapping thread of the SLAM algorithm some time to expand the map.

Another issue is the required initialization of the map. This has to be done before the vehicle can be controlled. At the moment we can either initialize the map by holding the helicopter in the hand, or load a stored one. We are currently including an algorithm that is able to take off from ground over a known pattern [28]. With that we could be able to initialize the map autonomously while the vehicle takes off.

VII. RESULTS AND DISCUSSION

In order to test the platform a setup was constructed, where the helicopter was secured by 2 nylon-wires within a large indoor laborotary environment.

When hovering the quadrotor is very stable. In Fig. 12 the flight path of 60 seconds hovering can be seen. As the mapping thread goes almost into sleep mode once a region has been explored, the computational power can be fully used for the tracking and controlling of the quadrotor. Also, the map will never get corrupted in this state and the localization quality is guaranteed. All in all the position error has an RMS value of 2.89 cm in x, 3.02 cm in y and 1.86 cm in z, what yields an absolute error value RMS of 4.61 (see Fig. 11). If the battery power was sufficient, the MAV could continue flying with this quality during months.

The platform is also able to fly to desired setpoints. For that the path is split into waypoints. The distance between them is chosen so that the helicopter can re-align the orientation of the inertial coordinate frame at each waypoint. Here, a little higher RMS value is obtained as in the hovering mode (see Fig. 13). The map consisted of 7 keyframes at the start. While expanding the map the SLAM algorithm has processed more than 40 additional keyframes. At each waypoint the helicopter stabilizes itself and waits until the RMS value is small enough (10 cm in absolute error value) to re-align the map. At the same time this does leave some time to the SLAM algorithm to process new keyframes and expand the map.

At some points, when the helicopter flies from waypoint to waypoint, a notable overshoot can be observed. This is due to the integrating action of the controller, which is necessary to correct steady state errors. It does introduce additional overshoot and reduces the robustness of the system. However, a satisfying trade-off was found.
Fig. 11. Position error in the x, y and z positions. The value remains between ± 10 cm. The z position is more accurate than the x and y positions.

Fig. 12. Position error while hovering during 60 seconds. The RMS value of the position error is 2.89 cm in x, 3.02 cm in y and 1.86 cm in z. The maximum runaway has an absolute error value of 11.15 cm (marked with a red o).

In Fig. 14 the map, that was built during the flight, can be seen. It is composed of 52 keyframes and 4635 map points. It represents approximately a surface of 15 m$^2$. We did not perform any test with ground truth comparison. This could present the amount of map drift that is present. However, the quadrotor will always localized itself correctly with respect to the nearer local environment, so that if we got some map drift, it does not influence the control quality.

Some failure modes have also to be mentioned. Due to the lack of features or to varying illumination it is possible that the tracker cannot find enough feature. This would disable any localization of the vehicle. Also, if the map gets too big, the mapping thread may use to much computational power for the global bundle adjustment, leading to a greater time delay and slower rate of the pose estimates until the controlled system gets unstable.

The achieved results show that our platform can autonomously fly through a larger unknown indoor environment with a high accuracy. The system is robust against external disturbances and can handle modeling errors. Some outdoor tests confirm the controller’s robustness, which was able to handle quit strong changing winds. Several live video can be found under the following links [1].

VIII. CONCLUSION

This paper presents a vision based MAV control approach. The pose is estimated by means of the SLAM algorithm of Klein et al. with a precision of some centimeters. It is then used to stabilize the position of the vehicle. Based on a control input transformation and on the linear LQG/LTR procedure a controller is designed. The resulting platform successfully manages to hover and follow desired setpoint within an indoor laboratory. For that it does not need any prior information on the environment. After the initialization a map is built of the surroundings, wherein the MAV is able to localize itself without any time-drift. Apart from some minor map drift the vehicle can control its position up to some centimeters of error (RMS around 2-4 cm). All in all we built an autonomous MAV platform which is able to navigate in an unknown and unstructured environment in a very robust and accurate way.
REFERENCES


