2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Congress Center Hamburg
Sept 28 - Oct 2, 2015. Hamburg, Germany

Real-Time Visual-Inertial Mapping, Re-localization and Planning Onboard MAVs in Unknown Environments

Michael Burri, Helen Oleynikova, Markus W. Achtelik and Roland Siegwart
Autonomous Systems Lab, ETH Zürich

***Abstract*— In this work, we present an MAV system that is able to relocalize itself, create consistent maps and plan paths in full 3 D in previously unknown environments. This is solely based on vision and IMU measurements with all components running onboard and in real-time. We use visual-inertial odometry to keep the MAV airborne safely locally, as well as for exploration of the environment based on high-level input by an operator. A globally consistent map is constructed in the background, which is then used to correct for drift of the visual odometry algorithm. This map serves as an input to our proposed global planner, which finds dynamic 3 D paths to any previously visited place in the map, without the use of teach and repeat algorithms. In contrast to previous work, all components are executed onboard and in real-time without any prior knowledge of the environment.**

I. Introduction

Small multi-rotor helicopters, to which we refer as Micro Aerial Vehicles (MAVs), have drawn the attention of both research groups and industry, because of their size, light weight and great versatility. However, the agility of these platforms also creates great research challenges in the fields of localization, mapping, control and planning. Thanks to recent advances in mobile and light-weight computers, and developments in on-board state estimation [1]–[3], MAVs are more and more capable of navigating without the aid of external motion tracking systems. This makes them very compelling platforms for industrial inspection or search and rescue missions, especially in cluttered environments.

Such scenarios present problems even for human operators, where fast reaction times are necessary to safely operate the MAV in close proximity to obstacles. This becomes even more difficult when the MAV is far away, or when line of sight cannot be maintained. While low-level stabilization tasks can be taken over by the computer, humans are still vastly better at judging and prioritizing, for instance during exploration or determining points of interest for inspection.

We suggest using local visual-inertial odometry in the control loop, enabling an operator to explore previously unseen environments based on first person view and high-level velocity commands. During this exploration phase, our mapping module creates a dense globally consistent map of the environment. When the MAV needs to return to the starting location, for instance to exchange the battery, we employ this global map for two purposes: First, we use a global planner to find a short and feasible dynamic 3 D path, even if the explored path contained branches. This distinguishes our approach from teach and repeat approaches [4], where the travelled path is followed back exactly even if it contained detours. Second, once a feasible path is found, we relocalize the MAV against the global map and correct for drift of the visual odometry system while the path is followed. The generated map is compact and can be preloaded to plan a path to any previously visited location, for instance to continue an inspection task after changing batteries. An example image of the final system in action is shown in Fig. 1. With all computation running onboard, our system is independent of brittle data links and off-board computation. This is crucial as a safety feature for a robust system, where the robot must be able to return on its own in case of communication loss.

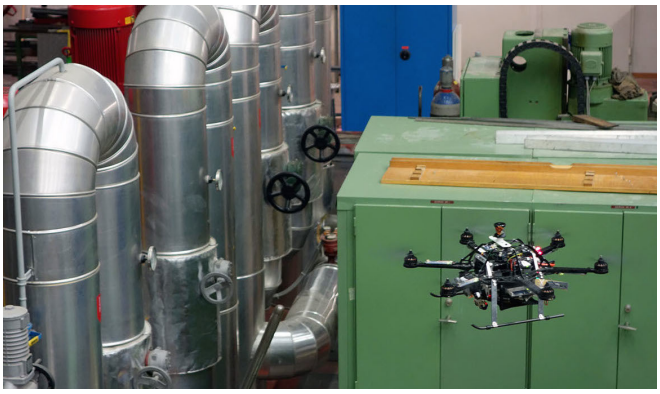

Fig. 1: Instant of the flight experiment, following the trajectory back to the take-off position.

The contributions of the paper are as follows: We show how to incorporate local maps from our visual-inertial odometry estimator into a globally consistent map. Online re-localization is performed against this map to compensate for drift in the visual-inertial odometry system. We present how to build a dense 3 D occupancy grid, represented as an octomap [5], from this sparse posegraph. Lastly, we introduce an improved version of the 3 D path-planning algorithm proposed by Richter *et al.* [6]. Our version is able to handle state constraints (e.g. maximum velocity, acceleration), has improved numerical stability and reduced computation time.

All components are running onboard a real MAV in flight while operating in a cluttered environment – to our knowledge, this is the first MAV running vision based localization, mapping, and planning without prior knowledge of the environment entirely onboard and in real-time.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7) under grant-agreement n.608849 (EuRoC). (email:{michael.burri, markus.achtelik}@mavt.ethz.ch, oelena@student.ethz.ch, r.siegwart@ieee.org).

II. Related Work

Heng *et al.* [7] show a system on-board a helicopter that uses 3 D occupancy grids built from stereo images to get a 3 D representation of the environment. But the 3 D occupancy grid is directly built from the visual odometry estimates which leads to significant drift. They also perform bundle adjustment and loop closure off-board, but discard the improved external pose estimate, since it leads to jumps in their planner and relies on constant network connectivity.

Similar work was also done by Schmid *et al.* [8], where a local 3 D occupancy grid was built using the onboard visual odometry. The operator was able to guide the MAV using waypoints in the 3 D occupancy grid representation and a local planner connected the current position with the desired position using a down projected 2 D obstacle map.

Michael *et al.* [9] demonstrate the utility of vision-based 3 D maps for inspection tasks such as mapping earthquake damaged buildings. Their system is operated entirely in teleoperation, but records vision-based maps from both ground and flying robot. The maps are voxel-grid based and aligned and fused together using iterative closest point and a strong prior on the take-off position of the helicopter relative to the ground robot.

A commonly used strategy for homing and returning to a previous position is teach and repeat, based on either laser or visual data. Furgale *et al.* [4] were one of the first to show successful homing with visual teach and repeat on a ground robot. However, teach and repeat is always limited to following the same trajectory and finding the same viewpoints as the original demonstration, which can lead to needing to point the camera backwards or suboptimal paths. Having a global map to do the planning in allows us to overcome these limitations.

Richter *et al.* [6] was one of the first to show global planning for MAVs, and successfully demonstrated aggressive flight without external tracking. A laser scanner was used to relocalize against a previously recorded map and a global plan was calculated before flying. To reduce the computational complexity, a straight line solution is found using RRTStar [10]. Because straight lines are not well suited for MAV dynamics, we use high order polynomials to get a smooth trajectory. In this work the nonlinear optimization is extended, by including maximum velocity and accelerations as a soft constraints to account for limitations in the accuracy of the visual odometry estimate.

III. System Description

For the experiments we use an AscTec Firefly, equipped with a stereo camera and an IMU [11]. An overview over the software components is shown in Fig. 2, with the associated frame convention. The visual odometry module returns pose information in a local drifting frame, denoted as mission frame (M). This frame is mainly used to keep the UAV airborne safely and avoid big disturbances on the controller. In addition a local map around the MAV is built and could be used for local collision avoidance, which is not part of this work.

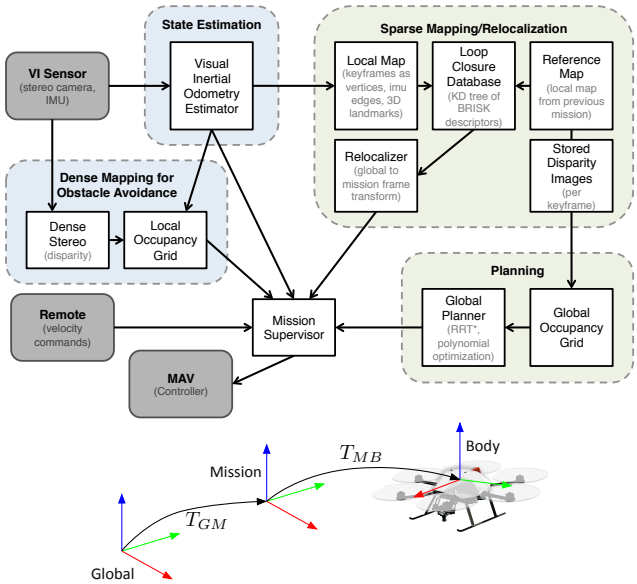

Fig. 2: Overview of the software components running on-board the helicopter. All components highlighted in blue are safety critical and running in the local mission frame (M). The global mapping and planning are running in the global frame (G) highlighted in green.

A sparse mapping and re-localization module builds a globally consistent map in parallel, which is referred to as global map (G). Having a global map allows our planner to find full 3 D paths to any previously seen position. The information of the global map for localization can be stored very efficiently [12] and thus shared between agents or reused in future missions.

A mission controller is used as a bridge between the MAV and the planner. In manual exploration mode, velocity commands from the operator are forwarded to the MAV. As soon as the trajectory following mode is activated, the reference from the global planner needs to be expressed in the mission frame before sending to the MAV.

IV. Localization and Mapping

In order to estimate the state of the MAV in the world, we use a keyframe-based visual inertial odometry estimator [13]. However, all odometry suffers from slow drift over time. This makes it impossible to perform meaningful global planning, as a drift in the position estimate of the helicopter relative to the planned global path may result in collisions with the environment. To overcome this problem, we build a local map based on the output of the visual-inertial odometry system in the background. We are then able to relocalize against this local map to correct for estimator drift. Furthermore, the framework allows for running bundle adjustment (BA) in the background to further reduce errors and drift in the odometry.

By storing stereo disparity images for keyframes and associating them with the vertices in the local map, we are additionally able to benefit from bundle-adjusted vertex poses when building a global 3 D dense model for planning.

A. Local Map Building

We use an implementation of the keyframe based visual inertial odometry described in [13] to get a local pose information. This approach tracks keypoints across keyframes in a camera image and estimates the poses using a joint optimization between keypoint tracks and IMU measurements. However, due to computational constraints, only a small number of the past keyframes is kept in the optimization. Therefore, in order to take full advantage of map-based optimization methods, we build up a local map on the keypoint tracks and keyframes from the estimator. This results in a sparse posegraph with keyframes as vertices and IMU measurements as edges. The vertices contain image keypoints, keypoint descriptors, and 3 D triangulated landmarks from keypoint tracks over several vertices and across the stereo camera pairs.

An important feature of our posegraph formulation is the use of missions (sub-maps) with independent baseframes that align the mission to the global frame. All vertex poses and landmark positions are expressed in the local mission frame. This allows us to update the alignment of several sub-maps relative to each other by changing the baseframe transformation between them, without having to update every vertex or landmark in the map. For this application, we can hold the reference map fixed and constantly update the tranformation between the local map and reference map as new information becomes available.

B. Mission Handling

In order to keep the local odometry frame consistent for controllers, we *only* modify the baseframe transforms. Before we can perform relocalization against the map we have built, we must break off the local map and add it to the reference map. Additionally, we use a framework that allows us to access the map from several threads at once and do transaction-based changes [14]. This allows us to continue building our incremental local map, while running a BA on the new reference mission.

However, as a result of BA, the positions of the vertices often change significantly. Therefore, to keep the map consistent, we need to move the baseframe of the new local mission based on how much the end of the reference mission moved during BA.

$$T_{V_b V_a} = T_{V_b}^{-1} T_{V_a} \quad (1)$$

$$T_{GM_l} = T_{V_b V_a} T_{GM_r} \quad (2)$$

Where T_{V_a} is the transformation representing the 6 DoF pose of the last vertex in the reference mission before the BA, T_{V_b} is the pose of the same vertex after BA, T_{GM_r} is the baseframe transformation for the reference mission, and T_{GM_l} is the update to the baseframe of the local mission. This adjustment to baseframes also allows us to have more than one reference mission - that is, the latest information about the alignment to the last reference mission is used when creating a new local mission.

C. Relocalization

The benefit of having a bundle-adjusted, fixed reference mission is to be able to localize against it to correct drift in the visual-inertial state estimate. This is done by detecting nearest neighbor matches in projected BRISK keypoint descriptor space [15], followed by outlier rejection using geometric verification in a RANSAC scheme. Any inliers from this procedure are added as constraints between the current vertex and triangulated 3 D keypoints in the reference map.

Since we assume estimator drift is slow relative to the motion of the MAV, we do not need to relocalize at the same rate as the pose estimator. Instead, we query for loop closures against the global map at every new keyframe. We then update the localization estimate by running a non-linear least squares optimization on all the constraints between keyframes and landmarks in the past sliding window of keyframes (typically 20). Both the position of the vertices and the 3 D position of the triangulated landmarks from the reference are held fixed, with the only non-fixed term being the baseframe transformation of the local mission. Reprojection errors between 2 D keypoints and corresponding 3 D landmarks in the reference missions are the residuals. More discussion on how relocalization mitigates the effects of estimator drift can be found in [16].

D. Global Dense Model

While building the local map, we also record stereo disparity images, store them to disk onboard the MAV, and associate them with our sparse relocalization map. We only store disparity maps for keyframes of the visual odometry (vertices in our posegraph), which minimizes access to disk. Since keyframes in visual-inertial odometry are already selected with sufficient motion in-between, this leads to even and efficient coverage of the covered area.

After breaking off a reference mission and running BA, we iterate over all past vertices in the reference mission and re-project the disparity images into the global map using the updated vertex poses. This allows us to build a 3 D occupancy grid of the helicopter's environment. Using the poses after BA corrects errors and leads to metrically-accurate maps.

V. Path Planning

In this section we describe how we plan paths through the maps that we generated in Section IV-D. We are interested in feasible and fast-to-compute solutions that take into account vehicle dynamics and do not need to stop at every intermediate point. Based on the analysis on the flat outputs of multirotor MAVs by Mellinger *et al.* [17], we plan paths in the flat output space of position and yaw and their derivatives.

Our solution is based on the approach proposed by Richter *et al.* [6], who suggest to sacrifice *theoretical* optimality in favor of computation time: instead of sampling state vertices in very high dimension (i.e. snap) and using a polynomial steering function, they suggest to sample in position only and use straight-line steering in the first place. Then, the resulting position vertices are used as support points to compute a

smooth piece-wise polynomial trajectory, while iteratively handling collisions that were not present in the straight path. Richter *et al.* show that for *practical* applications, i.e. a reasonable number of samples, their solution finds shorter paths than the theoretically optimal solution, which would use a polynomial steer function in the sampling phase.

A. Unconstrained Linear Initial Solution

We compute a linear initial solution following the method suggested by Richter *et al.* [6]. We briefly summarize the essentials in order to understand the non-linear solution and point out optimizations for both numerical stability and to save computation time. The value of a polynomial of order $N-1$ with N coefficients at time t can be expressed as:

$$p(t) = \boldsymbol{t} \cdot \boldsymbol{c};\ \boldsymbol{t} = \begin{bmatrix} 1 & t & t^2 \dots t^{N-1} \end{bmatrix}; \boldsymbol{c} = \begin{bmatrix} c_0 \dots c_{N-1} \end{bmatrix}^T \tag{3}$$

Where $\boldsymbol{t}$ is a vector containing the powers of t according to N, and $\boldsymbol{c}$ is a vector containing the polynomial coefficients. A trajectory consists of M continuous polynomial segments, where each polynomial is valid from $t=0$ to the segment duration $t = T_{s,i}, i = 1 \dots M$. In case of multiple dimensions, each segment consists of D polynomials. During the optimization process quadratic cost of the polynomials is considered, such that the cost over the whole trajectory is:

$$J_{\text{polynomials}} = \sum_{i=1}^{M} \sum_{d=1}^{D} \overbrace{\int_0^{T_{s,i}} \underbrace{\sum_{j=0}^{N-1} \left\| \frac{d^j p_{i,d}(t)}{dt^j} \right\| \cdot w_j}_{\text{cost per derivative term}}}^{J_{i,d}\text{,cost per polynomial}} \tag{4}$$

The cost $J_{i,d}$ for a polynomial in a segment i in dimension d with its derivatives weighted by w_j can be written as:

$$J_{i,d} = \boldsymbol{c}_{i,d}^T \cdot \boldsymbol{Q}(T_{s,i}) \cdot \boldsymbol{c}_{i,d} \tag{5}$$

We only consider cost of the snap here, thus $w_4 = 1$ and all other w_j are zero. This optimization is subject to equality constraints in terms of derivatives at the start and end[1] of each segment, and have the form:

$$\underbrace{\begin{bmatrix} \boldsymbol{d}_{i,d,\text{start}} \\ \boldsymbol{d}_{i,d,\text{end}} \end{bmatrix}}_{\boldsymbol{d}_{i,d}} = \underbrace{\begin{bmatrix} \boldsymbol{A}(t=0) \\ \boldsymbol{A}(t=T_{s,i}) \end{bmatrix}}_{\boldsymbol{A}} \cdot \boldsymbol{c}_{i,d} \tag{6}$$

Where $\boldsymbol{A}$ is a mapping matrix between $\boldsymbol{c}$ and $\boldsymbol{d}_{i,d}$ that consists of row vectors $\boldsymbol{t}$ and $\frac{d}{dt}\boldsymbol{t}$ according to the specified end point derivative. Note that both the cost-matrix $\boldsymbol{Q}$ and mapping matrix $\boldsymbol{A}$ only depend on the segment time $T_{s,i}$ and thus are constant over all dimensions for the segment, which allows for computation-time savings in the case of multiple dimensions. $\boldsymbol{d}_{i,d}$, $\boldsymbol{Q}$ and $\boldsymbol{A}$ can now be stacked to form a joint optimization problem over the whole trajectory.

For solving this problem, we refer to Richter *et al.* [6] who showed how to solve this as an unconstrained QP, and its superior numerical stability compared to a constrained QP. In their method, the mapping matrices $\boldsymbol{A}$ for each segment need to be inverted, and involve high powers of t. For improved numerical robustness and lower computation time, we further exploit the structure of $\boldsymbol{A}$:

$$\boldsymbol{A}(t=0) = \begin{bmatrix} \frac{d^0}{dt^0}\boldsymbol{t}(0)^T \dots \frac{d^{N/2-1}}{dt^{N/2-1}}\boldsymbol{t}(0)^T \end{bmatrix}^T \tag{7}$$

$$\boldsymbol{A}(t=T_{s,i}) = \begin{bmatrix} \frac{d^0}{dt^0}\boldsymbol{t}(T_{s,i})^T \dots \frac{d^{N/2-1}}{dt^{N/2-1}}\boldsymbol{t}(T_{s,i})^T \end{bmatrix}^T \tag{8}$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}(t=0) \\ \boldsymbol{A}(t=T_{s,i}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{0} \\ \boldsymbol{\Gamma} & \boldsymbol{\Delta} \end{bmatrix} \tag{9}$$

As a result of the segment time being zero at the beginning, only the constant parts of $\boldsymbol{t}$ and its derivatives[2] are left in the upper part, why $\boldsymbol{\Sigma}$ is a diagonal matrix. Similarly, $\boldsymbol{\Gamma}$ is an upper diagonal matrix, and only $\boldsymbol{\Delta}$ is fully dense. We use the Schur-Complement to invert this matrix as follows:

$$\boldsymbol{A}^{-1} = \begin{bmatrix} \boldsymbol{\Sigma}^{-1} & \boldsymbol{0} \\ -\boldsymbol{\Delta}^{-1}\boldsymbol{\Gamma}\boldsymbol{\Sigma}^{-1} & \boldsymbol{\Delta}^{-1} \end{bmatrix} \tag{10}$$

$\boldsymbol{\Sigma}$ is simple to invert and does not contain high powers of t. $\boldsymbol{\Delta}$ however does contain high powers of t, but the dimension and the condition number became much lower than for inverting the whole matrix $\boldsymbol{A}$ at once.

B. Nonlinear Trajectory Refinement

Up to this stage we assumed that the times $T_{s,i}$, needed traverse each segment, are known, which is not the case in practice. We are interested in finding solutions with minimal segment times, while respecting state constraints, such as maximum velocity and acceleration.

1) Nonlinear optimization problem: We add the segment times $T_{s,i}$ as optimization variables, which turn the problem into a nonlinear optimization problem due to the powers of t. The cost function needs to be modified by adding the time, since otherwise the optimizer would drive $T_{s,i}$ to large values in order to minimize the cost for snap:

$$J = J_{\text{polynomials}} + k_T \cdot (\sum_{i=1}^{M} T_{s,i})^2 \tag{11}$$

k_T is an "aggressiveness" constant that lets us trade snap cost against time. The total set of optimization variables consists now of the free end-point derivatives from the linear solution [6] and the segment times $\begin{bmatrix} T_{s,1} \dots T_{s,M} \end{bmatrix}$. We compute an initial solution with the linear method described above. For an initial guess of the segment times, we use $\frac{1}{2} v_{\max}$ over the straight line distance between two vertices.

2) Finding maxima: In order to incorporate state limitations such as maximum velocity or acceleration, one option is to sample the trajectory at a certain interval and add an inequality constraint to the optimization framework for each of these samples. This leads to many inequality constraints that have to be evaluated, and will thus slow down the optimization procedure significantly for longer trajectories. In addition, the question raises how to handle these discrete

[1] This does not necessarily have to be at the beginning or end of a segment. As long as there are enough free parameters, there could also be constraints at arbitrary times in the segment.

[2] Due to the derivatives, the constant part in $\boldsymbol{t}$ shifts right.

sampling points, if the segment times of the trajectory get adjusted by the optimizer.

The following analysis shows how to compute maxima of the given 3 D polynomial trajectory analytically for a single polynomial segment, and is repeated for all segments of the trajectory. Especially for velocity and acceleration, we are interested in the maximum of the norm and not in each single axis. The norm of the velocity can be written in terms of the polynomials of position as:

$$v_{\text{norm}}(t) = \sqrt{(\dot{p}(t)_x)^2 + (\dot{p}(t)_y)^2 + (\dot{p}(t)_z)^2} \tag{12}$$

To find the candidates for the maximum, we need to compute the derivative with respect to time:

$$\frac{dv_{\text{norm}}(t)}{dt} = \frac{2\,(\dot{p}(t)_x \cdot \ddot{p}(t)_x + \dot{p}(t)_y \cdot \ddot{p}(t)_y + \dot{p}(t)_z \cdot \ddot{p}(t)_z)}{-\sqrt{(\dot{p}(t)_x)^2 + (\dot{p}(t)_y)^2 + (\dot{p}(t)_z)^2}} \tag{13}$$

To find an analytical solution for which the numerator becomes zero, we do the following: Recall that when taking time derivatives of a polynomial, there are additional coefficients from taking derivatives of the powers of t. For convenience, we denote the coefficients of $\dot{p}(t)$ as $\dot{\boldsymbol{c}}$, and so forth. A multiplication of two polynomials can in general be expressed as a convolution of their coefficients, thus the problem becomes:

$$\boldsymbol{t} \cdot (\dot{\boldsymbol{c}}_x * \ddot{\boldsymbol{c}}_x) + \boldsymbol{t} \cdot (\dot{\boldsymbol{c}}_y * \ddot{\boldsymbol{c}}_y) + \boldsymbol{t} \cdot (\dot{\boldsymbol{c}}_z * \ddot{\boldsymbol{c}}_z) \stackrel{!}{=} 0 \tag{14}$$

$$\boldsymbol{t} \cdot (\dot{\boldsymbol{c}}_x * \ddot{\boldsymbol{c}}_x + \dot{\boldsymbol{c}}_y * \ddot{\boldsymbol{c}}_y + \dot{\boldsymbol{c}}_z * \ddot{\boldsymbol{c}}_z) \stackrel{!}{=} 0 \tag{15}$$

The expression in (15) is a polynomial, for which we compute the real roots with the numerically stable Jenkins-Traub algorithm [18]. The real roots within $[0, T_s]$ in are then candidates for the maximum, which we find by numerical evaluation of $v_{\text{norm}}(t)$ at $t_{\text{candidate}}$. The same methodology can be applied for higher order derivatives such as acceleration.

3) Incorporating state constraints in the nonlinear optimization problem: A straight-forward solution is to use an optimization method that is able to incorporate nonlinear inequality constraints. This limits however the choice of optimization methods. Also, it turned out in practical experiments that adding inequality constraints increases the number of necessary iterations significantly and the optimizer does not always respect the constraints. Since state constraints are more guidelines than hard constraints in our case, we implemented state constraints as soft constraints by adding an additional cost term:

$$J_{\text{soft-constraint}} = \exp\left(\frac{x_{\text{max, actual}} - x_{\text{max}}}{x_{\text{max}} \cdot \epsilon} \cdot k_s\right) \tag{16}$$

ϵ defines how much the deviation from the maximum is tolerated and k_s is a constant that allows to set how much the violation of a constraint is weighted. This is no critical tuning parameter in practice, just the order of magnitude has to be right.

4) Handling collisions in the optimized path: A problem of the suggested method is that the optimized trajectory computed from the straight-line solution from the RRT* planner is not guaranteed to be collision free anymore, since polynomials tend to overshoot at the vertices of the straight line path. First, we highlight that this overshoot is reduced significantly by the nonlinear optimization over all parameters of the trajectory (see Section V-C). Similarly to [6], we handle the remaining collisions by adding additional vertices on the straight line, which is guaranteed to be collision free, as shown in Fig. 3.

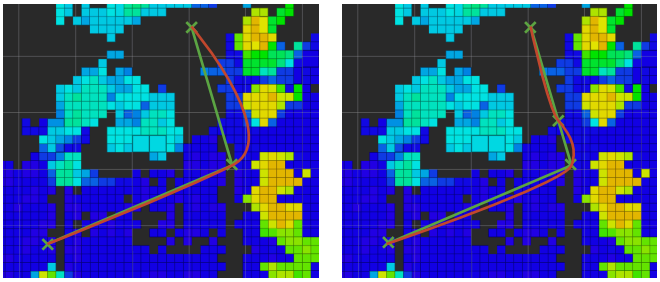

Fig. 3: Handling of collisions on the polynomial path: we project the collision onto the straight-line path and add a vertex. This pulls the polynomial towards the straight line path and slows down the trajectory.

C. Path Optimization Results

We give a brief analysis of the path optimization framework on given vertices, as they would result from a RRT* planner. The RRT* planner is excluded in this analysis, since its solutions are highly dependent on the environment. All polynomials in the experiments are of order 9, i.e. 10 coefficients, continuous up to snap, and we optimize over the integral of squared snap.

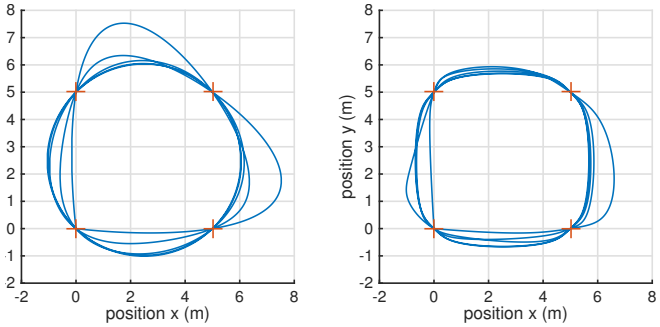

Fig. 4: Comparison of the initial linear solution (left) with the optimized solution (right), through vertices aligned in multiple squares of $5 \times 5\,\text{m}$. The optimized solution stays closer to straight-line connections, while still staying withing the maximum speed limits. The time required to fly along this trajectory is 200 s for the initial, and 112 s for the optimized solution.

We compare the initial solution from the linear method with the optimized version through vertices aligned in multiple $5 \times 5\,\text{m}$ squares, as shown in Fig. 4. The optimized version stays closer to the straight-line connections, which is important in order to avoid too many re-iterations for inserting additional vertices in case of collisions of the optimized path with the environment (see Section V-B.4).

We randomly generated sets of vertices and show timings, for the initial linear solution t_{init}, the nonlinear optimization t_{opt}, and whether generating the trajectory was successful. A trajectory is considered successful, if it does not exceed the state limits ($v_{\text{max}} = 2\,\text{m/s}$, $a_{\text{max}} = 2\,\text{m/s}^2$,) by 10 %. For each

number of segments, ran the optimization over 100 different paths, where the average distance of the vertices was 5 m.

segments	t_{init} (ms)	t_{opt} (ms)	std dev t_{opt} (ms)	success (%)
3	0.117	48.0	12.1	96
5	0.171	143	41.4	95
10	0.297	584	169	91
20	0.565	2157	632	88
50	1.58	10110	1290	47

TABLE I: Timings for the path optimization for selected numbers of segments. t_{init} is the time required to compute the initial linear solution, while t_{opt} is the time for the non-linear optimization.

The results are shown in Table I. From 3-20 segments, we have a high success-rate, while the optimized solution is found within reasonable time.

VI. Experiments

To demonstrate the utility of our system for assisting in inspection and mapping tasks, we designed a series of experiments closely mimicking real industrial applications. We used an AscTec Firefly hex-rotor helicopter equipped with a computer and our visual inertial sensor [11]. Images are processed at 20 Hz and fused with the IMU to get higher update rates for the controller. From the keyframes we store the position and a disparity image to recreate the 3 D occupancy grid before the planning phase. The two experiments were conducted in the machine hall at ETH and show our algorithm working in an unstructured 3 D environment. All processing is done online on the MAV with no external computation.

A. Map Generation and Homing

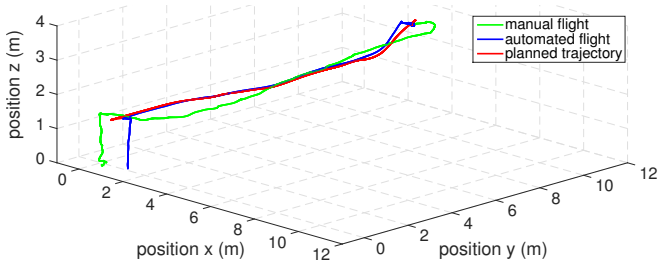

Fig. 5: Flight back to the take-off position. The manual flight in green was used to create the map. At the end of the green trajectory the homing command was given, to trigger the planner and fly back to the original position.

In the first experiment the MAV starts without any prior information about the environment. The pilot gives high-level position commands in order to explore the environment. This phase is shown in green in Fig. 5. During the flight, the MAV builds a local 3 D occupancy grid from the stereo images that contains some drift from the visual odometry. It also builds the sparse posegraph described in Section IV-C. After some minutes of flight the MAV receives the signal to fly back to the starting position. This event could also be triggered by a low battery threshold or loss of connectivity, to safely return.

After a few iterations of BA the global 3 D occupancy grid is generated by projecting the disparity maps from the optimized keyframe positions into 3 D space. This is needed to reduce drifts in the 3 D map and make the global planning consistent. The planned path together with the resulting 3 D occupancy grid is shown in Fig. 7. Using snap optimized polynomials leads to smooth paths, that are easy to follow for MAVs. The heading is always set into the direction of flight, which would allow for dynamic obstacle avoidance from forward-facing camera data.

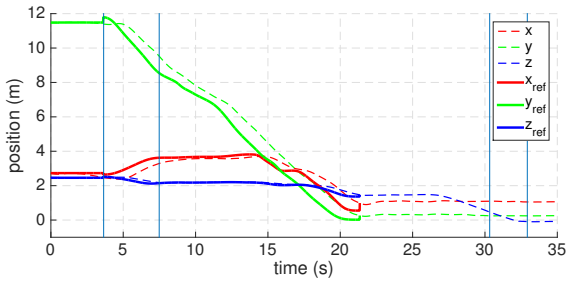

Fig. 6: Flight back to the take-off position including the landing phase. The planned trajectory is shown in bold and the successful relocalizations are shown with horizontal lines. After executing the planned trajectory, the MAV returns into teleoperation mode. The noticable lag in the controller is due to the internal reference model.

Fig. 6 shows the planned path in bold and the flown trajectory expressed in the drifting odometry frame. To overcome drifts, the re-localization module attempts to localize the local map against the bundle-adjusted reference map at every keyframe (approximately 4 Hz) and corrects the reference trajectory if necessary. This is shown with the blue vertical lines. As can be seen there are only loop closures at the beginning and the end of the trajectory because of the large deviations in the viewpoint from the previously seen trajectory. The results also show that the controller can handle small jumps in the reference, which shows the advantage of having the re-localization run separately. In future work the planner could be triggered for a re-planning of the current segment in case of new loop closures.

B. Reusing the Previous Map

In the second experiment we used the map generated in the first mission to plan a path back to the position where we started the previous trajectory, to "resume the mission". Although this is not necessarily required and the planner can find feasible paths to any known point in the map. This is a very useful feature in any real world scenario, where the limited battery time makes it necessary to return home quickly, change the battery and continue the original task. Since the starting position is very similar to the starting position of the previous mission, good loop-closures are found and the MAV can relocalize to reuse the previous 3 D occupancy grid for planning. The planned path and the resulting trajectory are shown in Fig. 8. Due to the random nature of RRT* and the limited time for planning a feasible trajectory, the path is different from the first experiment. Loop-closures are marked with vertical lines and similar conclusions as before can be drawn. Again the different viewpoint leads to successful loop-closures mainly at the beginning and end of the trajectory.

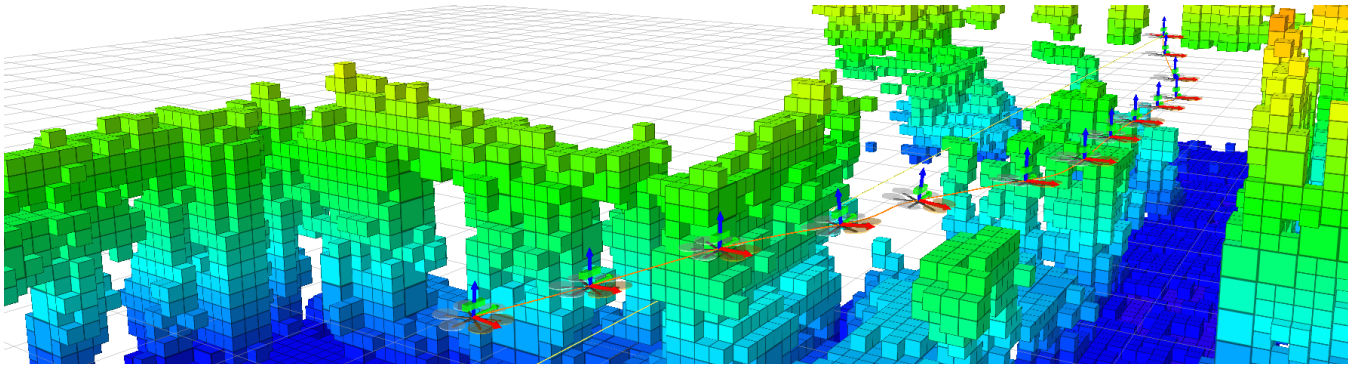

Fig. 7: After the manual flight phase the bundle adjustment is triggered and the global 3 D occupancy grid is generated. Additionally, the operator can verify the planned path back to the starting position before the MAV starts following it.

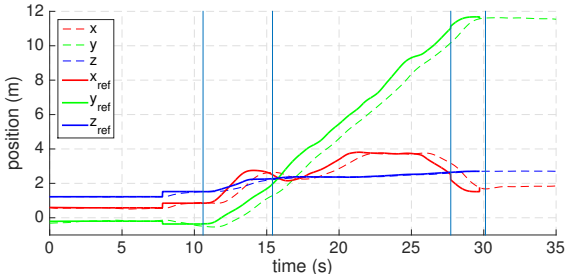

Fig. 8: Once the map is generated, it can be preloaded to plan trajectories to any previously covered position. The target position was set to a similar position from where the MAV returned before. The resulting trajectory is shown in bold and the MAV was able to follow it with a small delay. After executing the planned trajectory, the MAV returns into teleoperation mode. Successfuly relocalizations are shown as blue vertical lines.

VII. Conclusion

In this work we showed successful vision-based homing entirely on-board an MAV, without any prior information about the environment. Using BA of a sparse pose graph and re-localization to landmarks in this pose-graph, we are able to correct the drifting odometry estimates. This allows us to create a metrically-accurate 3 D occupancy grid by projecting disparity maps from the bundle-adjusted keyframe poses. Once the map is built, it can be used to plan arbitrary trajectories or repeat a task multiple times thanks to the re-localization. Finally, we show a planner that finds feasible 3 D paths in short time, which is essential for use with relocalization: if a successful relocalization causes a large-enough change in the estimated alignment to the global map, we trigger replanning the path.

Acknowledgments

The authors thank Simon Lynen, Marcin Dymczyk and Titus Cieslewski for providing the mapping framework and Andreas Jäger for providing the dense stereo pipeline.

References

[1] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A Robust and Modular Multi-Sensor Fusion Approach Applied to MAV Navigation," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, Nov. 2013.

[2] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *Robotics & Automation Magazine, IEEE*, vol. 19, no. 3, pp. 46–56, 2012.

[3] S. Shen, N. Michael, and V. Kumar, "Autonomous indoor 3d exploration with a micro-aerial vehicle," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012.

[4] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, vol. 27, no. 5, 2010.

[5] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, 2013.

[6] C. Richter, A. Bry, and N. Roy, "Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2013.

[7] L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Autonomous visual mapping and exploration with a micro aerial vehicle," *Journal of Field Robotics*, vol. 31, no. 4, pp. 654–675, 2014.

[8] K. Schmid, T. Tomic, F. Ruess, H. Hirschmuller, and M. Suppa, "Stereo vision based indoor/outdoor navigation for flying robots," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3955–3962.

[9] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, *et al.*, "Collaborative mapping of an earthquake-damaged building via ground and aerial robots," *Journal of Field Robotics*, vol. 29, no. 5, 2012.

[10] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[11] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 431–437.

[12] M. Dymczyk, S. Lynen, T. Cieslewski, M. Bosse, R. Siegwart, and P. Furgale, "The gist of maps – summarizing experience for lifelong localization," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015.

[13] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, 2014.

[14] T. Cieslewski, S. Lynen, M. Dymczyk, S. Magnenat, and R. Siegwart, "Map api - scalable decentralized map building for robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[15] S. Lynen, M. Bosse, P. Furgale, and R. Siegwart, "Placeless place-recognition," in *3D Vision (3DV), 2nd International Conference on*, 2014.

[16] H. Oleynikova, M. Burri, S. Lynen, and R. Siegwart, "Real-time visual-inertial localization for aerial and ground robots," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015.

[17] D. Mellinger and V. Kumar, " Minimum Snap Trajectory Generation and Control for Quadrotors ," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[18] M. A. Jenkins, "Algorithm 493: Zeros of a real polynomial [c2]," *ACM Transactions on Mathematical Software (TOMS)*, vol. 1, no. 2, pp. 178–189, June 1975.