2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

Robust Map Generation for Fixed-Wing UAVs with Low-Cost Highly-Oblique Monocular Cameras

Timo Hinzmann, Thomas Schneider, Marcin Dymczyk, Amir Melzer,
Thomas Mantel, Roland Siegwart, and Igor Gilitschenski[1]

***Abstract*— Accurate and robust real-time map generation onboard of a fixed-wing UAV is essential for obstacle avoidance, path planning, and critical maneuvers such as autonomous take-off and landing. Due to the computational constraints, the required robustness and reliability, it remains a challenge to deploy a fixed-wing UAV with an online-capable, accurate and robust map generation framework. While photogrammetric approaches have underlying assumptions on the structure and the view of the camera, generic simultaneous localization and mapping (SLAM) approaches are computationally demanding. This paper presents a framework that uses the autopilot's state estimate as a prior for sliding window bundle adjustment and map generation. Our approach outputs an accurate geo-referenced dense point-cloud which was validated in simulation on a synthetic dataset and on two real-world scenarios based on ground control points.**

I. INTRODUCTION

Recent years have shown an increasing interest in using UAVs to enable applications such as industrial inspection, surveillance, and agricultural monitoring at much lower cost than conventional aircrafts. In contrast to rotary-wing UAVs, fixed-wing UAVs can cover large areas in a short amount of time. Even some solar-powered fixed-wing UAVs have recently demonstrated a flight endurance of several days [1]. These properties make fixed-wing UAVs the ideal platform for mapping missions but also requires a robust framework that efficiently processes the large amount of recorded data. For obstacle avoidance and path planning, for instance, the information needs to be available in the range of milliseconds and seconds respectively. But also for search and rescue or surveillance missions, the map needs to be available as soon as possible.

Photogrammetric approaches for UAV mapping missions have shown impressive results in the last decades [2], [3] leading to several commercial products such as Pix4D. However, these approaches are usually processed off-line and are formulated as a batch optimization problem. Furthermore, they often assume a nadir-looking camera, the landmarks to lie approximately on a common ground plane, and are unable to make use of all sensor measurements.

A major advantage of state-of-the-art SLAM approaches [4], [5] is the capability of efficiently generating a map and simultaneously localizing the robot within this map based on visual and inertial measurements. However, this advantage can also turn into a drawback when a low-cost camera is involved and the landmark-distance to inter-keyframe baseline becomes too high which results in a high uncertainty of the landmark locations. A degradation of the map leads to a degradation of the state estimation and vice versa. For fixed-wing UAVs that are loitering above the same scenery, altitude drift of the estimated robot position becomes visible in form of a map consisting of stacked landmark layers.

To avoid these effects of scale ambiguity at high altitudes, our method decouples state estimation and map generation and is based on the indirect Extended Kalman Filter implementation presented in [6]. In this regard, our approach is similar to the one presented by Irschara in [7] where weak position and orientation priors speed up the structure from motion calculation. However, the latter considers all information simultaneously whereas our approach is designed to run onboard of the UAV and to compute a dense point-cloud once the camera poses and images are available. An overview of our framework is given in Fig. 1: The state estimates

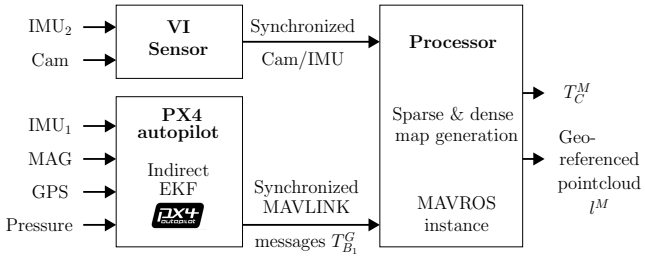

Fig. 1: Framework overview for state estimation and map generation.

of the PX4 autopilot are used as camera pose priors for feature tracking, triangulation, and are refined in a sliding window bundle adjustment scheme. Although the framework is presented for this specific setup it can be employed with any sensor or state estimation setup which provides camera pose priors. To the best of our knowledge, this paper represents the first approach to generate a dense map onboard of a fixed-wing UAV by decoupling state estimation and mapping in a fixed-lag smoothing formulation. In sum, we present a flight-tested real-time capable mapping framework with the following benefits:

- No assumptions on the structure, such as ground plane assumption.
- In contrast to most photogrammetric approaches we do

[1] All authors are with the ETH, the Swiss Federal Institute of Technology Zurich, Autonomous Systems Lab (www.asl.ethz.ch), Leonhardstrasse 21, LEE, CH-8092 Zurich, Switzerland. {firstname.lastname}@mavt.ethz.ch.

not assume that the camera is mounted with a nadir view. In our case the camera is mounted with a highly-oblique view such that it can be used for obstacle avoidance and map generation at the same time.
- Robust mapping with a low-cost monochrome camera by decoupling of state estimation and map generation.
- The computational cost is kept bounded due to the sliding window bundle adjustment scheme that keeps only a subset of the camera poses in the optimization problem. The cost for state estimation and map generation is distributed on different boards.

The remainder of this paper is structured as follows: Section II presents the methodology for generating a dense point-cloud based on camera pose priors. In Section III, the sliding window bundle adjustment is validated on a synthetic dataset and compared to the result of the batch bundle adjustment. The small unmanned research plane used for the real-world experiments is described in Section IV. Section V presents real-world experiments, which are based on two datasets recorded onboard of the fixed-wing UAV and includes the results of the sparse and dense mapping framework as well as landmark accuracy validation. The paper concludes with Section VI.

II. METHODOLOGY

The outline of the mapping algorithm is presented in algorithm 1. The most relevant parts include feature tracking, feature track triangulation, bundle adjustment and dense reconstruction.

Algorithm 1 Map generation

1: For every image, retrieve initial robot pose estimate from EKF and compute camera pose (II-A, II-B)
2: Extract feature tracks (II-C.1):
- Lucas-Kanade feature tracker
- Gyroscope measurement integration for feature prediction
- Feature bucketing
- Two-point RANSAC outlier rejection

3: Initialize landmarks (II-C.2):
- Check if landmark is well constrained
- Apply Gauss-Newton triangulation (inverse depth) [8] with ground plane initialization
- Check if landmark location is in visible camera cone

4: Perform sliding window bundle adjustment (II-C.3)
5: Add resulting optimized camera poses to dense reconstruction pipeline (II-D)
6: Insert landmarks into octomap (II-E)

A. EKF-based autopilot

The Pixhawk PX4 auto-pilot performs an indirect EKF-based state estimation as presented in [6]. The Kalman filter uses the linear acceleration and angular rates measurements for propagation of the state equations. The dynamic and static pressure, GPS velocity and position as well as 3D magnetometer measurements are used for the Kalman Filter state update. The estimated states consist of sensor (gyroscope and accelerometer) biases, wind field, three-dimensional airspeed as well as the IMU's attitude and position in WGS84 coordinates. For more details about the state estimation framework we refer to [6].

B. Transformations

The EKF estimates the position and orientation of IMU_1 in a global coordinate system. However, for mapping we need to compute the camera pose with respect to the global mapping coordinate system denoted by M. The transformation chain is stated in equation 1 and involves the camera (denoted with C), the IMU_1 of the autopilot which is rigidly mounted in the fuselage (denoted with B_1), as well as the IMU_2 mounted on the sensor pod (denoted with B_2) as shown in Fig. 2.

$$\mathbf{T}_C^M = \mathbf{T}_G^M \mathbf{T}_{B_1}^G \mathbf{T}_{B_2}^{B_1} \mathbf{T}_C^{B_2} \tag{1}$$

with

$\mathbf{T}_C^{B_2}$: Transf. of the camera w.r.t the sensorpod IMU
$\mathbf{T}_{B_2}^{B_1}$: Transf. of sensorpod IMU w.r.t the autopilot IMU
$\mathbf{T}_{B_1}^G$: Transf. of the autopilot IMU w.r.t global system
$\mathbf{T}_G^M$: Transf. of the global system w.r.t mapping system

The transformation of the camera with respect to the IMU of the sensorpod as well as the camera intrinsics and distortion are calibrated with the standard Kalibr stack [9]. The rotation

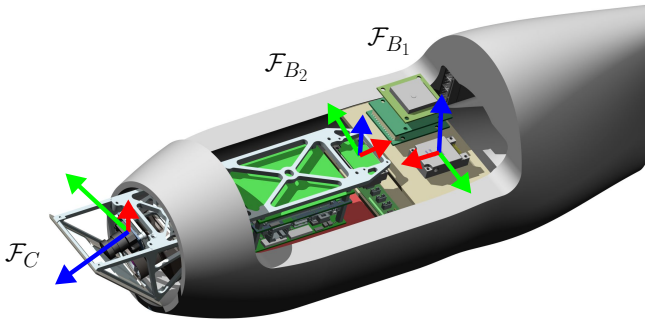

Fig. 2: Interior of the UAV platform Techpod: The PX4 autopilot as well as its GPS receiver, magnetometer, IMU and pressure sensors are rigidly mounted. The sensorpod is a modular unit that is used in several UAVs.

and translation between the IMU of the sensorpod and the IMU of the autopilot is computed with an extension of Kalibr which was proposed by Rehder and Nikolic [10] and capable of precise IMU-IMU transformation estimation. Finally, the position estimate coming from the autopilot is transformed from WGS84 coordinates into a metric UTM coordinate system. In a last step, the orientation needs to be aligned with the UTM coordinate system: In our case, the IMU assumed north direction to be x. However, in UTM coordinates, easting is x, northing is y. Consequently, the transformation matrix $\mathbf{T}_M^G$ results in a $90\,^\circ$ rotation around

the z-axis:

$$\mathbf{T}_M^G = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The transformations were validated by solving an absolute perspective n-point problem (PNP), as proposed by Kneip and implemented in OpenGV [11]. The landmark positions in UTM coordinates were obtained from geo-referenced satellite images, the feature location in the image were hand-labeled as shown in Fig. 3.

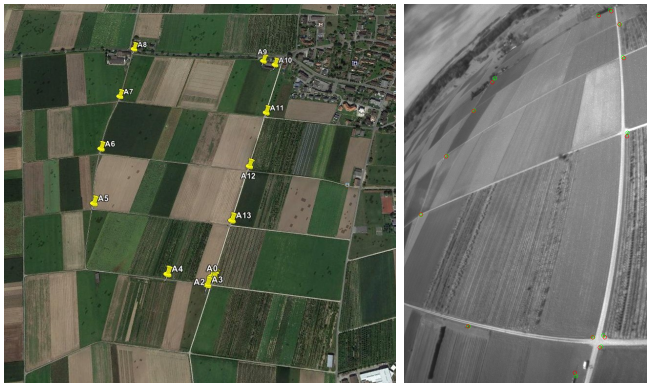

Fig. 3: Left: Landmark positions in UTM coordinates. **Right:** Hand-labeled feature positions (green) and reprojected landmark locations (red) given the camera pose estimated by the PNP.

C. Sparse point-cloud generation

The sparse map generation is presented in algorithm 1. The most relevant parts include feature tracking, feature track triangulation and sliding window bundle adjustment.

1) Feature Tracking: In a first step, features are extracted based on the approach proposed by Shi and Tomasi [12]. The features are then tracked with a Lucas-Kanade feature tracker. The gyroscope measurements of the sensorpod IMU[1] are integrated between successive frames to predict the feature location in the next frame and to constrain the search window and limit the computational cost. Feature bucketing ensures uniformly distributed feature observations across the image. Eventually, a 2-point relative translation-only RANSAC problem [13] is solved to identify and reject outliers. Fig. 4 shows the feature tracks classified by RANSAC as inliers in green and outliers in red. As visualized in blue, a mask can be used to avoid extracting features close to the horizon as the observation rays to the corresponding landmarks are almost parallel which results in high landmark position uncertainties. Alternatively, a horizon tracker or the evaluation of the observation angles can be used to reject not well constrained landmark observations.

[1]In theory, also the autopilot IMU could have been used for feature tracking. However, the camera and IMU measurents are time-synchronized on the FPGA and come in with a higher rate (200 Hz instead of 100 Hz).

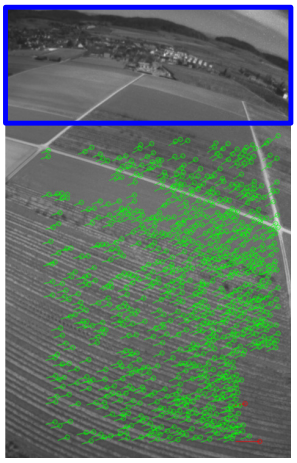

Fig. 4: Feature tracking results showing the inlier set (green), outlier set (red) as well as horizon mask (blue).

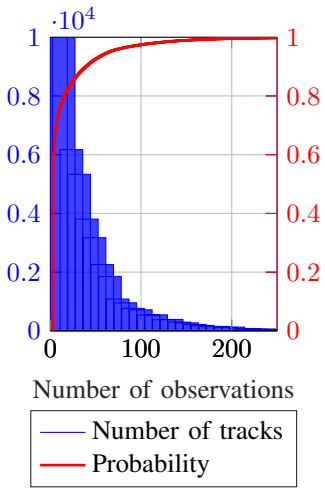

Fig. 5: Feature track histogram for 20 fps. 93 percent of the feature tracks contain less than 50 observations.

2) Feature Triangulation: The often used direct triangulation approach may suffer from local minima [7]. For better numerical stability we use the Gauss-Newton triangulation where the landmark location is in inverse-depth parametrization as proposed in [8]: The basic idea is that the position of the j-th feature observed in the i-th camera frame can be expressed in terms of the n-th camera frame:

$$\begin{aligned} \mathbf{p}_{f_j}^{C_i} &= \mathbf{C}(\bar{q}_{C_n}^{C_i})\mathbf{p}_{f_j}^{C_n} + \mathbf{p}_{C_n}^{C_i} \\ &= Z_j^{C_n}\left(\mathbf{C}(\bar{q}_{C_n}^{C_i})\begin{bmatrix}\alpha_j & \beta_j & 1\end{bmatrix}^\top + \rho_j \mathbf{p}_{C_n}^{C_i}\right) \\ &= Z_j^{C_n}\begin{bmatrix}h_{i1} & h_{i2} & h_{i3}\end{bmatrix} \end{aligned} \tag{3}$$

with $\begin{bmatrix}\alpha_j & \beta_j & \rho_j\end{bmatrix}^\top = \begin{bmatrix}\frac{X_j^{C_n}}{Z_j^{C_n}} & \frac{Y_j^{C_n}}{Z_j^{C_n}} & \frac{1}{Z_j^{C_n}}\end{bmatrix}^\top$

Equation 3 is expressed in inverse depth parametrization and α, β and ρ are the minimization variables. We provide the complete pseudo-code in the appendix. The Gauss-Newton triangulation shows a good trade-off between accuracy and calculation time. Since this is an iterative approach, the algorithm needs to be initialized appropriately. In particular, an educated guess for the landmark position in terms of the first camera frame needs to be set. For this, we assume that the landmark is located on an approximated ground plane.

3) Sliding Window Bundle Adjustment: The visual bundle adjustment considers only the last N camera poses to retain real-time processing, where N denotes the size of the sliding window. The bundle adjustment is formulated in a factor graph and optimized with *Georgia Tech Smoothing and Mapping* (GTSAM) [14], [15]. The insertion of the factor nodes, value nodes, the marginalization strategy as well as outlier rejection is explained based on the sample factor graph shown in Fig. 6: Once a new pose-image pair is available, the camera pose is inserted in the factor graph and initialized with the estimate from the Extended Kalman Filter. To constrain the factor graph, each camera pose is

associated with a prior factor based on the mean and standard deviation from the EKF. The triangulated landmarks are inserted as value nodes and connected to the individual camera poses via reprojection factors. The reprojection error formulation is adopted from [5]:

$$\mathbf{e}_r^j = \mathbf{z}^j - \mathbf{h}(\mathbf{T}_M^C \mathbf{l}_j^M) \tag{4}$$

where $\mathbf{h}(\cdot)$ stands for the camera projection and $\mathbf{z}^j$ is the feature measurement of landmark j in image coordinates.

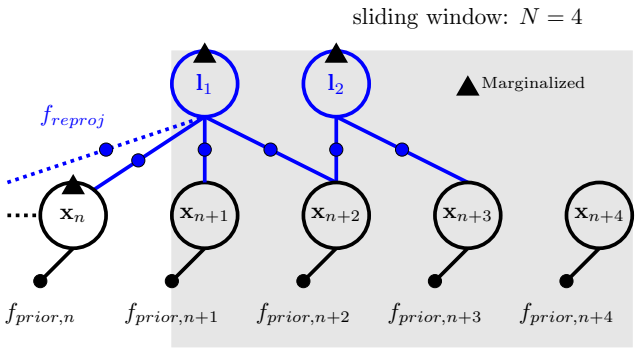

Fig. 6: Sliding window bundle adjustment in factor graph formulation.

Every reprojection factor has a Cauchy M-Estimator to reduce the influence of outliers[2]. The factor graph is optimized, visual measurements that are still identified as outliers are rejected and the graph is optimized again. After the graph optimization, all landmarks and all camera poses (i.e. the oldest one) outside of the sliding window are marginalized. The selection of the sliding window size N constitutes a trade-off between computational costs and accuracy of the state estimates. The window size should be set after evaluating the expected length of the feature tracks to ensure that the majority of observations of the feature tracks are included as reprojection factors in the optimization problem. Fig. 5 shows the feature track length histogram for the dataset I presented in Sec. V. The empirical cumulative distribution function illustrates that around 97 percent of all feature tracks contain less than 100 observations which corresponds to a 5.0 s sliding window. Choosing a too small value for the sliding window results in a shorter baseline of first-to-last camera pose in the feature track and consequently in less constrained landmarks as well as camera poses.

D. Dense point-cloud generation

The image stream and the corresponding optimized camera poses are then used as input for the dense reconstruction algorithms. The dense reconstruction approaches can be divided into rectification-based and patch-based methods [16], [17]. The rectification-based methods seem more suited for our purpose since they are computationally more efficient. In general, the goal of rectification is to transform an arbitrarily arranged stereo pair into a rectified virtual stereo pair, in which epipolar lines become collinear and horizontal. In the rectified stereo pair the correspondence search becomes easier since matches lie on horizontal lines of the rectified images and efficient blockmatching algorithms can be employed [18], [19]. The output of the dense reconstruction module is a geo-referenced dense point-cloud which is colored by pixel intensities. The next section shortly describes the planar and polar rectification procedures and their advantages and disadvantages based on our camera configuration.

(a): Nadir **(b): Highly oblique**

Fig. 7: (a): Nadir (down-looking) camera configuration. The epipoles are at infinity for the forward moving UAV and both, planar and polar rectification can be used. **(b):** Highly-oblique camera configuration. The epipoles might be close or even within the images. In this case the planar rectification fails and the polar rectification algorithm needs to be used for dense reconstruction.

1) Planar rectification: One of the properties of the virtual ideal stereo camera is, that it has parallel optical axes, which are perpendicular to the baseline. This ensures that the epipoles are at infinity and hence epipolar lines become collinear and parallel. The nadir (down-looking) camera configuration inherently fulfills these requirements for planar rectification as shown in Fig. 7. However in case of an oblique camera, the epipoles might be close or even within the images depending on the concrete motion of the fixed-wing UAV. In that case, pixels around the epipole project to infinitely far away points on the rectified image plane, which would result in extremely large and distorted rectified images. For implementation details we refer to [20].

2) Polar rectification: In contrast to planar rectification, the approach proposed by [21] can be employed for generic camera motion. The approach takes directly the pixel intensities along corresponding epipolar lines and inserts them into the same row of the rectified images. The rectified images are finally built up by circularly scanning the original images around the epipoles. A detailed description can be found in [21]. Compared to the planar rectification, the polar rectification algorithm is computationally more involved and produces more outliers. To benefit from both approaches, we keep a small buffer of frames and compute the geometry of the current camera frame to the camera frames in the buffer. If the epipoles are close or within all stereo pair combinations, we use polar otherwise we employ planar rectification.

E. Octomap interface

The generated point-cloud is inserted into OctoMap [22]. Two possible ways to insert the point-cloud exist:

- Endpoint-only: Only encodes the occupied space by inserting the landmarks directly.

[2]The Cauchy weight is $k^2/(k^2+e^2)$, where e is the residual and k is a constant set to 3.0.

- Ray-casting: Encodes the occupied and free space in a probabilistic manner by casting rays from the camera position to the landmark location which is useful for obstacle avoidance and path-planning. However, the calculation costs increases with the distance to the landmark.[3]

The point-cloud generated onboard of the fixed-wing UAV could then be visualized directly on the ground-control computer if the UAV is in WiFi range.

III. SIMULATION

To validate the performance of the sliding window bundle adjustment under realistic conditions it was simulated based on a real-world point-cloud[4] and camera poses recorded by the presented fixed-wing UAV. The nominal flight altitude in the simulation is around $150\,\mathrm{m}$. The camera intrinsics are identical to the ones used for the real-world datasets. The camera positions are disturbed with Gaussian white noise $\mathcal{N}(0, (0.5\,\mathrm{m})^2)$; the feature observations with $\mathcal{N}(0, (0.2\,\mathrm{pixel})^2)$. The minimal and maximal track length were set to 2 and 50 respectively.

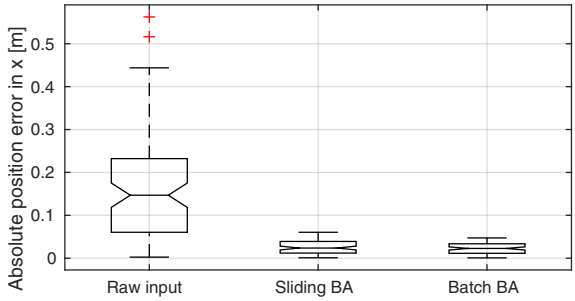

Fig. 8: Performance comparison of sliding window bundle adjustment and batch bundle adjustment.

The absolute position errors in x of the disturbed input, the result of the sliding window bundle adjustment and of the batch bundle adjustment are shown in Fig. 8. The accompanying data is given in Table I.

	$\mu(\lvert e_x \rvert)$	$\sigma(\lvert e_x \rvert)$	$\mu(\lvert e_y \rvert)$	$\sigma(\lvert e_y \rvert)$	$\mu(\lvert e_z \rvert)$	$\sigma(\lvert e_z \rvert)$
Input	0.1610	0.1177	0.1605	0.1155	0.1431	0.0973
Sliding	0.0246	0.0150	0.0163	0.0221	0.0210	0.0165
Batch	0.0231	0.0135	0.0128	0.0073	0.0192	0.0045

TABLE I: Mean μ and standard deviation σ of the camera position errors in meter. The batch bundle adjustment performs only slightly better compared to the sliding window approach.

As expected, the batch bundle adjustment performs slightly better than the sliding window bundle adjustment since the whole factor graph is available for optimization.

[3]For a nominal flight altitude of around $150\,\mathrm{m}$ and several hundred landmarks per image ray-casting becomes computationally costly. Compare performance discussion in [22].

[4]Dataset *cadastre* of the Pix4D example datasets.

IV. PLATFORM

The small unmanned research plane Techpod was used for the experiments presented in this paper. It has a wingspan of $2.60\,\mathrm{m}$, classic T-tail configuration and is equipped with one propeller. The sensor pod and PX4 auto-pilot are placed inside the fuselage as shown in Fig. 2 and allow autonomous mission execution such as GPS-waypoint following. The technical specifications of the sensor and processing unit are listed in Table II. As exteroceptive sensor it features an Aptina MT9V034 monochrome global shutter camera capturing images with a resolution of 752×480 pixels at up to $60\,\mathrm{fps}$. It is rigidly mounted with an oblique field of view of around $45\,\mathrm{deg}$. For measuring angular velocities and angular accelerations, the sensor pod is equipped with a MEMS inertial measurement unit (IMU). The camera and IMU are integrated into an ARM-FPGA-based Visual-Inertial (VI) sensor system [23] allowing hardware-synchronized IMU and camera data. An Intel Atom CPU (four cores at 1.92 GHz) is connected to the VI sensor system and the PX4 auto-pilot. All components are mounted on an aluminium frame that guarantees a rigid camera-imu transformation throughout all flight scenarios.

Sensor pod	Monochrome camera:	Aptina MT9V034
	IMU	ADIS16448
	Thermal camera	FLIR Tau 2
	Processing board	Kontron COMe-mBT10
	Processor	Intel Atom (4 cores, 1.91 GHz)
Auto-pilot	IMU	ADIS16448
	GPS	uBlox LEA-6H
	Processor	Cortex M4F (168 MHz)
	RAM	192 kB

TABLE II: The sensor and processing unit, nicknamed sensor pod, and the PX4 Pixhawk auto-pilot used for the experiments.

More technical details about the deployed camera is presented in the appendix.

V. REAL WORLD EXPERIMENTS

The mapping pipeline was evaluated based on two datasets, denoted with Set I and Set II, which were recorded by the fixed-wing UAV presented in Section IV. Set I is characterized by a nominal altitude of $100\,\mathrm{m}$ to $150\,\mathrm{m}$ and a flat scenery as shown in Fig. 3. The sparse point-cloud generated during this flight is shown in Fig. 9. The landmarks are inserted into OctoMap once they leave the sliding window and are colored by height. The estimated landmark locations agree with the ground control points obtained from satellite data as presented in Section V-A.

Set II was recorded during the final demonstration of the ICARUS FP7 project in March-en-Famenne, Belgium. The nominal altitude of around $70\,\mathrm{m}$ is lower than in Set I as can be seen from Fig. 10 and 12. Due to the 3-dimensional structure, it is used to present the results of the dense reconstruction pipeline.

A. Landmark accuracy validation

For fixed-wing UAVs it remains a challenge to obtain high-quality ground-truth data of the camera poses. Hence,

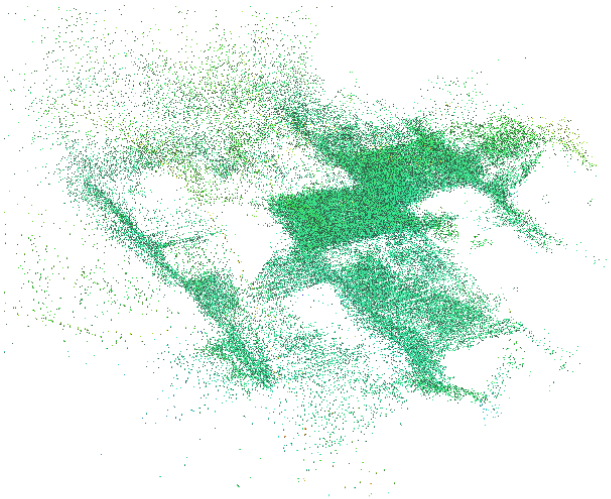

Fig. 9: Sparse point-cloud visualized in octomap, colored by height: $2264 \times 1473 \times 245\,\mathrm{m}^3$ with 264793 landmarks, leaf size is $1\,\mathrm{m}$.

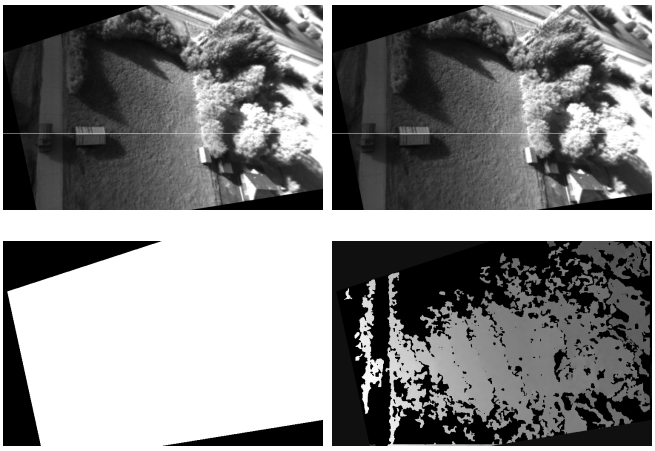

Fig. 10: Planar rectification: The images on the top show the rectified images of the virtual stereo pair. The images on the bottom shows the employed mask and the disparity image seen from the virtual stereo rig.

we assess the quality of our approach based on ground control points obtained from satellite images. Care was taken to use permanent and easily identifiable ground control points suchs as house corners or road crossings. The process of obtaining the ground control point is as follows: The optimized landmark is obtained from the mapping pipeline where every landmark is associated with the corresponding image, keypoint and 3D position. The keypoint location in the image is visualized and the location is labeled in the satellite image. The results for Set I and II are presented in Table III.

	$\mu(\lvert e_x \rvert)$	$\sigma(\lvert e_x \rvert)$	$\mu(\lvert e_y \rvert)$	$\sigma(\lvert e_y \rvert)$	$\mu(\lvert e_z \rvert)$	$\sigma(\lvert e_z \rvert)$
Set I	7.9840	7.7916	3.7913	2.7068	6.3316	0.2593
Set II	2.2571	0.7771	4.5677	1.4708	3.2798	2.0935

TABLE III: Landmark accuracy validation based on two real-world datasets. The Table shows the mean μ and standard deviation σ of estimated landmark location to ground control points in meter. For each set we gathered 10 ground control points.

Overall, Set II achieves a higher accuracy compared to Set I which can be explained by the different flight altitudes. At higher flight altitudes, the same orientation, distortion or intrinsics error result in a larger landmark position error when the ray is projected on the ground. Possible error sources of the accuracy assessment include inaccuracies of the satellite images and of the ground control point labeling process.

B. Dense reconstruction

In this section, the output of the dense reconstruction module is presented to underline the accuracy of the bundle adjusted camera poses. Fig. 10 visualizes the rectified images and disparity map generated by the planar rectification algorithm.

Feature correspondences can be searched across horizontal lines as illustrated in the top row of Fig. 10. The point-cloud and virtual stereo pair are visualized in Fig. 11.

The generated landmarks are colored by the pixel intensities of the original image and are geo-referenced in UTM coordinates. The inter-keyframe baseline is $1.35\,\mathrm{m}$, the epipoles are located at $e_1 = [990.483,\ 302.183]$ and $e_2 = [1005.53,\ 289.519]$. From visual inspection, the planar rectification approach generates a consistent point-cloud with few outliers.

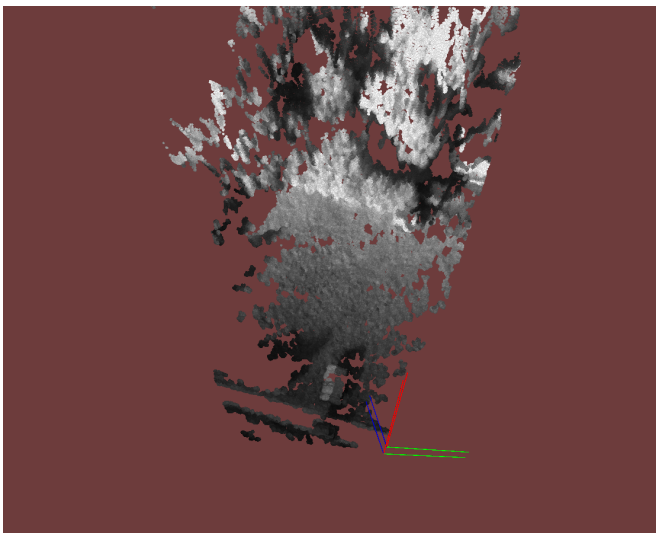

Fig. 11: Geo-referenced point-cloud generated with planar rectification based on two views. The inter-keyframe baseline is $1.35\,\mathrm{m}$. Both epipoles are outside of the images.

This is in particular true in the well observed region below the inter-keyframe baseline.

Analogously, the images rectified by the polar rectification approach, the disparity map as well as the generated point-cloud are shown in Fig. 12. Compared to the results of the planar rectification, the point-cloud generated by polar rectification shows decisively more outliers. As a conclusion, the planar rectification outperforms the polar rectification in terms of accuracy and runtime as presented in Table V. Therefore, the usage of the planar rectification is to be maximized to produce consistent point-clouds. Nevertheless,

polar rectification is used in cases that the planar rectification fails due to the geometry of the virtual stereo pair.

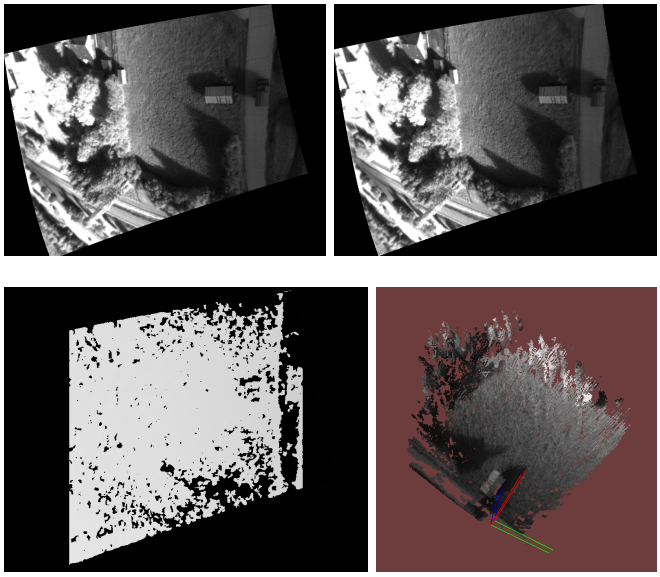

Fig. 12: Polar rectification: The images on the top show the rectified images of the virtual stereo pair. The image on the bottom left visualizes the disparity image seen from the virtual stereo rig. On the right is the geo-referenced point-cloud generated with polar rectification based on two views.

C. Runtime

The preliminary runtime results of the mapping pipeline are shown in Table IV. The validation was performed on an Intel(R) Core(TM) i7-4800MQ CPU @ 2.70GHz for reference. The runtime results look promising and future work will include the optimization of the pipeline in order to make it run in real-time onboard of the fixed-wing UAV.

	mean $[ms]$	std. dev. $[ms]$
Feature tracking*	19.129	5.631
Feature triangulation	0.525	0.266
Bundle adjustment	26.571	17.511

TABLE IV: Runtime in ms for one frame. (*Runs in a separate thread and does not count towards total frame processing time.)

The runtime for the planar and polar densification step is shown in Table V. The increased complexity of the polar rectification algorithm is revealed in the runtime. The computational costs of both approaches can be decreased by downsampling the original image if necessary.

	Planar rectific. $[ms]$	Polar rectific.$[ms]$
Rectification	6.89	15.31
Correspondence search	14.54	24.13
Point-cloud generation	4.08	6.96
Frame processing	25.51	46.41

TABLE V: Average runtime (200 runs) per image in ms for the planar and polar rectification. The original image size (752×480) was used for the rectification.

VI. CONCLUSIONS

In this paper we presented a robust mapping framework that generates dense, geo-referenced point-clouds based on camera pose priors. The novelty of this approach lies in the use of a state-of-the-art smoothing and mapping framework in combination with camera pose priors to *iteratively* obtain depth estimates of the environment onboard of a fixed-wing UAV. The sliding window bundle adjustment was validated on a realistic synthetic dataset and achieved comparable results to the full batch bundle approach. Real world experiments were performed on a small fixed-wing UAV equipped with a low-resolution monochrome camera which was mounted with a highly-oblique view. The experiments underlined the generality of the approach that does not make any assumptions on the observed structure: The landmark accuracy which was validated with ground control points performed equally good for a flat as well as for a 3D-structured scenery. The dense reconstruction pipeline considers the oblique view by checking the geometry of the stereo pairs and either performs planar or polar rectification. A rigorous outlier rejection in several layers of the pipeline ensures an accurate dense point-cloud.

The proposed framework is used as a robust backbone for autonomous UAV missions where obstacle avoidance and path-planning as well as autonomous landing and take-off are required. As a next step, we seek to improve the map coverage by feeding the map information to the path planning and control algorithms. In future work, the free space is to be computed in a more efficient manner, building up on the OctoMap implementation for ray-casting. Furthermore, for the results shown in this paper a constant camera rate was used. By using a camera view selection algorithm, the computational cost could be decreased.

APPENDIX

A. Gauss-Newton iterative multi-view triangulation

In this section, the pseudo-code for the iterative Gauss-Newton triangulation from subsection II-C.2 is presented in Algorithm 2. The notation used in the pseudo-code of the Gauss-Newton triangulation is adopted from [8].

B. Camera parameters

The camera field of view is given by

$$FOV = 2\tan^{-1}\left(\frac{n_{x,y}x}{2f}\right) \tag{5}$$

which computes to $77.72\,^\circ$ and $54.43\,^\circ$ for the horizontal and vertical FOV respectively given the camera parameters presented in table VI.

The ground sampling distance is then given by

$$GSD = xhf^{-1} \tag{6}$$

The image footprint is given by:

$$IFP = GSD\begin{bmatrix} n_x & n_y \end{bmatrix}^T \tag{7}$$

For the nominal flight altitude of $h = 200\,\mathrm{m}$, one pixel corresponds to $0.43\,\mathrm{m}$ on the ground and the image foot

Algorithm 2 Gauss-Newton triangulation

T : Precision threshold
r : Measurement residual
J : Measurement Jacobian
$h_{m,i}$: Observation of the feature in camera i
$h_{p,i}$: Predicted observation of the feature in camera i
M : Number of observations for the feature

function GAUSSNEWTON($\alpha_{init}, \beta_{init}, \rho_{init}, iter_{max}, T$)
  $\alpha \leftarrow \alpha_{init},\ \beta \leftarrow \beta_{init},\ \rho \leftarrow \rho_{init}$
  while $\|r_{last} - r_{current}\|_2 > T$ and $iter < iter_{max}$ **do**
    for $i = 1$: all camera views M **do**
      ${}^{C_n}\mathbf{C}_{C_i} \leftarrow {}^{C_n}\mathbf{C}_G \cdot {}^{G}\mathbf{C}_{C_i}$
      ${}^{C_i}\mathbf{p}_{C_n} \leftarrow {}^{C_i}\mathbf{C}_G\, {}^{G}\mathbf{p}_{C_n} - {}^{C_i}\mathbf{C}_G\, {}^{G}\mathbf{p}_{C_n}$
      $h_{p,i} \leftarrow {}^{C_n}\mathbf{C}_{C_i} \cdot \begin{bmatrix}\alpha & \beta & 1\end{bmatrix}^\top + \rho \cdot {}^{C_i}\mathbf{p}_{C_n}$
      $h_{m,i} = \frac{1}{Z_{C,m}} \begin{bmatrix}X_{C,m} & Y_{C,m}\end{bmatrix}^\top$
      $r_i \leftarrow h_{p,i} - h_{m,i}$ ▷ Residuals in camera coordinates
      $r \leftarrow \begin{bmatrix}r & r_i\end{bmatrix}^\top$ ▷ Stack residuals
      $J_p \leftarrow \frac{1}{h_{p,i(3)}} \begin{bmatrix}1 & 0 & -\frac{h_{p,i(1)}}{h_{p,i(3)}} \\ 0 & 1 & -\frac{h_{p,i(2)}}{h_{p,i(3)}}\end{bmatrix}$ ▷ J. persp. camera
      $J_\alpha \leftarrow {}^{C_n}\mathbf{C}_{C_i} \begin{bmatrix}1 & 0 & 0\end{bmatrix}^\top$
      $J_\beta \leftarrow {}^{C_n}\mathbf{C}_{C_i} \begin{bmatrix}0 & 1 & 0\end{bmatrix}^\top$
      $J_\rho \leftarrow {}^{C_i}\mathbf{p}_{C_n}$
      $J_i \leftarrow J_p \begin{bmatrix}J_\alpha & J_\beta & J_\rho\end{bmatrix}$
      $J \leftarrow \begin{bmatrix}J & J_i\end{bmatrix}^\top$ ▷ Stack Jacobians
    end for
    $\Delta \leftarrow (J^\top J)^{-1} J^\top r$ ▷ $J = 2M \times 3$, $r = 2M \times 1$
    $\begin{bmatrix}\alpha & \beta & \rho\end{bmatrix}^\top \leftarrow \begin{bmatrix}\alpha & \beta & \rho\end{bmatrix}^\top - \Delta$
  end while
  ${}^{G}\mathbf{p}_f \leftarrow \frac{1}{\rho} {}^{G}\mathbf{C}_{C_n} \begin{bmatrix}\alpha & \beta & 1\end{bmatrix}^\top + {}^{G}\mathbf{p}_{C_n}$
end function

print is $322.29 \times 205.71\,\mathrm{m}^2$. N.b. that these equations are correct for the ideal case of a nadir-looking camera.

Monochrome camera	Resolution n_x, n_y	752, 480
	Pixel size x	$6\,\mu$m
	Lense f	2.8 mm
	Max. rate	60 Hz
	Delpoyed rate	20 Hz

TABLE VI: Camera parameters.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Commission's Seventhes Framework Programme (FP7/2007-2013) under grant agreement n°285417 (ICARUS) and n°600958 (SHERPA). Furthermore, the authors thank Jörn Rehder and Janosch Nikolic for their support in terms of the IMU-IMU and IMU-Camera calibration framework and Andreas Jäger and Sammy Omari in terms of the dense reconstruction algorithms.

REFERENCES

[1] P. Oettershagen, A. Melzer, T. Mantel, K. Rudin, R. Lotz, D. Siebenmann, S. Leutenegger, K. Alexis, and R. Siegwart, "A Solar-Powered Hand-Launchable UAV for Low-Altitude Multi-Day Continuous Flight," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 3986–3993, IEEE, 2015.

[2] H. Eisenbeiß, *UAV Photogrammetry*. ETH Zurich, Switzerland:, 2009.

[3] F. Remondino, L. Barazzetti, F. Nex, M. Scaioni, and D. Sarazzi, "UAV photogrammetry for mapping and 3D modeling – Current status and future perspectives," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, no. 1, p. C22, 2011.

[4] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *Computer Vision–ECCV 2014*, pp. 834–849, Springer International Publishing, 2014.

[5] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization," in *Proceedings of Robotics: Science and Systems*, (Berlin, Germany), June 2013.

[6] S. Leutenegger, A. Melzer, K. Alexis, and R. Siegwart, "Robust state estimation for small unmanned airplanes," in *Control Applications (CCA), 2014 IEEE Conference on*, pp. 1003–1010, Oct 2014.

[7] A. Irschara, C. Hoppe, H. Bischof, and S. Kluckner, "Efficient Structure from Motion with Weak Position and Orientation Priors," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pp. 21–28, June 2011.

[8] A. I. Mourikis and S. I. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," in *Robotics and Automation (ICRA), 2007 IEEE International Conference on*, pp. 3565–3572, April 2007.

[9] P. Furgale, J. Rehder, and R. Siegwart, "Unified Temporal and Spatial Calibration for Multi-Sensor Systems," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 1280–1286, Nov 2013.

[10] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, "Extending kalibr: Calibrating the Extrinsics of Multiple IMUs and of Individual Axes," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Stockholm, Sweden), 16–21 May 2016.

[11] L. Kneip and P. Furgale, "OpenGV: A Unified and Generalized Approach to Real-Time Calibrated Geometric Vision," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 1–8, May 2014.

[12] J. Shi and C. Tomasi, "Good Features to Track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pp. 593–600, Jun 1994.

[13] L. Kneip, *Real-time Scalable Structure from Motion: From Fundamental Geometric Vision to Collaborative Mapping*. Dissertation, ETH Zurich, 2012.

[14] F. Dellaert, "Factor Graphs and GTSAM: A Hands-on Introduction," Tech. Rep. GT-RIM-CP&R-2012-002, GT RIM, Sept 2012.

[15] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation," in *Proceedings of Robotics: Science and Systems*, (Rome, Italy), July 2015.

[16] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Towards Internet-scale Multi-view Stereo," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 1434–1441, June 2010.

[17] Y. Furukawa and J. Ponce, "Accurate, Dense, and Robust Multiview Stereopsis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, pp. 1362–1376, Aug. 2010.

[18] K. Konolige, "Small Vision Systems: Hardware and Implementation," in *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, pp. 111–116, 1997.

[19] H. Hirschmuller, "Stereo Processing by Semiglobal Matching and Mutual Information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 328–341, Feb 2008.

[20] A. Fusiello, E. Trucco, and A. Verri, "A Compact Algorithm for Rectification of Stereo Pairs," *Mach. Vision Appl.*, vol. 12, pp. 16–22, July 2000.

[21] M. Pollefeys, R. Koch, and L. V. Gool, "A simple and efficient rectification method for general motion," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, pp. 496–501 vol.1, 1999.

[22] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A Probabilistic, Flexible, and Compact 3D map Representation for Robotic Systems," in *In Proc. of the ICRA 2010 workshop*, 2010.

[23] 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 *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pp. 431–437, 2014.