Monocular Visual-Inertial SLAM for Fixed-Wing UAVs Using Sliding Window Based Nonlinear Optimization

Timo Hinzmann[1(✉)], Thomas Schneider[1], Marcin Dymczyk[1], Andreas Schaffner[1], Simon Lynen[1,2], Roland Siegwart[1], and Igor Gilitschenski[1]

[1] Autonomous Systems Lab, ETH Zurich, Zurich, Switzerland
timo.hinzmann@mavt.ethz.ch
[2] Google Inc., Mountain View, USA

Abstract. Precise real-time information about the position and orientation of robotic platforms as well as locally consistent point-clouds are essential for control, navigation, and obstacle avoidance. For years, GPS has been the central source of navigational information in airborne applications, yet as we aim for robotic operations close to the terrain and urban environments, alternatives to GPS need to be found. Fusing data from cameras and inertial measurement units in a nonlinear recursive estimator has shown to allow precise estimation of 6-Degree-of-Freedom (DoF) motion without relying on GPS signals. While related methods have shown to work in lab conditions since several years, only recently real-world robotic applications using visual-inertial state estimation found wider adoption. Due to the computational constraints, and the required robustness and reliability, it remains a challenge to employ a visual-inertial navigation system in the field. This paper presents our tightly integrated system involving hardware and software efforts to provide an accurate visual-inertial navigation system for low-altitude fixed-wing unmanned aerial vehicles (UAVs) without relying on GPS or visual beacons. In particular, we present a sliding window based visual-inertial Simultaneous Localization and Mapping (SLAM) algorithm which provides real-time 6-DoF estimates for control. We demonstrate the performance on a small unmanned aerial vehicle and compare the estimated trajectory to a GPS based reference solution.

1 Introduction and Related Work

Unmanned aerial vehicles, and in particular fixed-wing UAVs, are powerful agents in scenarios where a large area needs to be scanned in a short amount of time. In recent years the interest in employing UAVs for applications such as industrial inspection, surveillance as well as agricultural monitoring has drastically increased due to the low acquisition and maintenance cost when compared to conventional aircraft. Today, GPS still remains the central source of 3-degrees of freedom (DoF) navigational information for the majority of these applications. Its limitations in certain cases, however, have motivated the exploration of alternative sources of 3-DoF signals such as ultra-sonic, laser, cameras and beacons.

DOI: 10.1007/978-3-319-50835-1_51

Due to the low cost and weight of the sensor suite motion estimation based on data from cameras and inertial-measurement units (IMU) has become a common choice to obtain a metric 6-DoF estimate of the body motion in real-time. There exists a large body of robotic research focusing on probabilistic fusion of visual and inertial data over which the publications of Nerurkar et al. [1] and [2] give an excellent overview. All these approaches perform simultaneous localization and mapping by jointly minimizing errors from reprojecting triangulated 3D-landmarks and integrating the measurements from the IMU. The currently best performing algorithms employ either (nonlinear) optimization [1–3], filtering [4–8] or combinations of both [9].

Optimization based approaches potentially achieve higher accuracy due to the ability to limit linearization errors through repeated linearization of the inherently nonlinear problem. This is particularly relevant in visual-inertial navigation systems (VINS). Here the relinearization of the IMU based propagation limits errors caused by inaccurate linearization points of IMU biases. Nevertheless, solving the covariance form of the problem in a recursive estimator is a common choice due to the lower computational requirements. Only recently formulations which allow "online mapping" in inverse form have shown real-time performance [1,2]. To retain real-time calculations these algorithms approximate the problem using one of the following approaches:

- Key-frame based concepts as proposed e.g. in [1,10]. These approaches, however, require an involved marginalization strategy and the constraints imposed by integrating the IMU measurements can theoretically span an indefinite duration.
- Fixed-lag smoothing or sliding window approaches that consider time successive states within a fixed time interval in the optimization problem but *discard* measurements outside of the sliding window. Cf. discussion in e.g. [11] of how this leads to loss of information regarding the variable interaction.
- Incremental smoothing and mapping methods such as iSAM2 [12] that optimize over all robot states and landmarks, making use of all available correlations.

In this paper we combine the last two approaches such that poses and associated landmarks older than the constant smoother lag are *marginalized* to ensure a smooth pose estimate while keeping the problem size bounded. In contrast to [13], we only employ a short term sliding window in factor graph formulation. Furthermore, we use a tightly-coupled integration approach where the error originating from pre-integrated IMU measurements as well as the reprojection errors can be jointly optimized and thus all correlations within the optimization window are considered [14]. Our contributions lie in the modifications to increase robustness and to lower computational requirements which allow operation onboard a fixed-wing UAV. We furthermore discuss the hardware setup of our modular sensor pod and thereby demonstrate accurate SLAM on a platform with limited dimensions and payload capabilities.

2 Methodology

This section introduces the coordinate frames, states, problem statement as well as the proposed visual-inertial estimation framework.

2.1 Coordinate Frames

We distinguish between the camera frame $\mathcal{F}_C$, the body frame $\mathcal{F}_B$ as well as the global frame $\mathcal{F}_G$. Furthermore, the body frame $\mathcal{F}_B$ is assumed to be identical to the IMU frame $\mathcal{F}_I$ as illustrated in Fig. 1.

2.2 State Vector

The SLAM problem seeks to estimate the robot states $\mathbf{x}_R$ as well as the set of landmarks $\mathbf{x}_L$ with $\theta = \{\mathbf{x}_R, \mathbf{x}_L\}$. The robot state is defined as

$$\mathbf{x_R} := \left[\mathbf{p}_B^{G^\top}, \mathbf{q}_B^{G^\top}, \mathbf{v}_B^{G^\top}, \mathbf{b}_g^\top, \mathbf{b}_a^\top\right]^\top \in \mathbb{R}^3 \times \mathbb{S}^3 \times \mathbb{R}^9 \tag{1}$$

and comprises the robot pose and velocity in the global frame as well as the gyroscope and accelerometer biases.

2.3 SLAM as Maximum a Posteriori (MAP) Problem

We seek to maximize the joint probability $p(\mathbf{x}_R, \mathbf{x}_L, \mathbf{z})$ which is equivalent to minimizing the negative log-likelihood of the robot states $\mathbf{x}_R$ and set of landmarks $\mathbf{x}_L$ given the measurements $\mathbf{z}$

$$\begin{aligned} \theta^* &:= \arg\max_\theta p(\mathbf{x}_R, \mathbf{x}_L | \mathbf{z}) = \arg\max_\theta p(\mathbf{x}_R, \mathbf{x}_L, \mathbf{z}) \\ &= \arg\min_\theta \{-\log p(\mathbf{x}_R, \mathbf{x}_L, \mathbf{z})\} = \arg\min_\theta \{-\mathcal{L}(\mathbf{x}_R, \mathbf{x}_L, \mathbf{z})\}, \end{aligned} \tag{2}$$

where $\mathcal{L}$ represents the log-likelihood function.

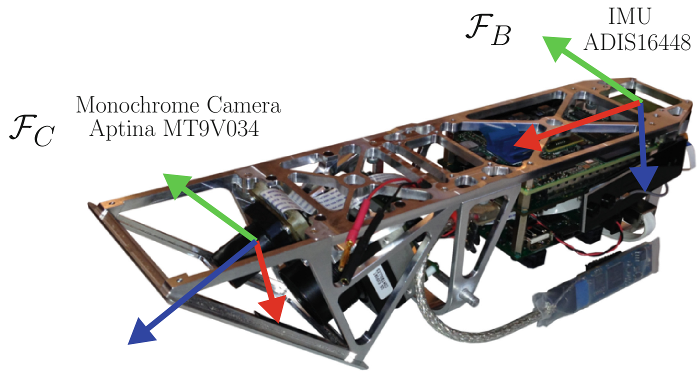


Fig. 1. Camera and body coordinate frame of the sensor processing unit.

2.4 Factor Graph Formulation

The SLAM problem can be transformed into a factor graph $\mathcal{G} = \{\mathcal{F}, \theta, \mathcal{E}\}$ which is a bipartite[1] graph consisting of a set of unknown random variables to be estimated θ and a set of factors $\mathcal{F}$. In our case, the variables θ consist of the robot states $\mathbf{x}_R$ and the set of landmarks $\mathbf{x}_L$. The factors $\mathcal{F}$ are given by prior knowledge or measurements $\mathbf{z}$. The edges connect variable and factor nodes. The factor graph uses a factorization of the function $f(\theta) = \prod_i f_i(\theta_i)$ which we seek to maximize, i.e. we want to calculate

$$\begin{aligned}\theta^* &= \arg\max_\theta\{f(\theta)\} = \arg\min_\theta\{-\log f(\theta)\} \\ &= \arg\min_\theta \left\{ -\log \prod_i \exp\left(-\frac{1}{2}||z_i - h_i(\theta_i)||^2_{\Sigma_i}\right)\right\} \\ &= \arg\min_\theta \left\{ \sum_{i=1}^{I} ||z_i - h_i(\theta_i)||^2_{\Sigma_i} \right\} \\ &= \arg\min_\theta \left\{ \sum_{i=1}^{I} \mathbf{e}_i^T \Sigma_i^{-1} \mathbf{e}_i \right\} = \arg\min_\theta \left\{ \sum_{i=1}^{I} \mathbf{e}_i^T W_i \mathbf{e}_i \right\},\end{aligned} \tag{3}$$

where we used the fact that the measurement covariance matrix Σ is the inverse of the information matrix W and the Mahalanobis distance is defined as $||\mathbf{e}||^2_\Sigma := \mathbf{e}^T \Sigma^{-1} \mathbf{e}$.

Factor and Error Term Definitions. Only the factors and error terms are presented in this section. For the derivation of the Jacobians and information matrices we refer to [2,16,17].

Reprojection factor. We define the reprojection factor as

$$f_{reproj}(\mathbf{x}_R, \mathbf{x}_L) = \mathbf{d}(\mathbf{e}_{reproj}) \propto \exp\left(-\frac{1}{2}||\mathbf{e}_{reproj}||^2_{\mathbf{W}^{-1}_{reproj}}\right), \tag{4}$$

where $\mathbf{d}(\cdot)$ denotes a cost function, $\mathbf{e}_{reproj}$ is the reprojection error and $\mathbf{W}_{reproj}$ is the corresponding information matrix. We adopt the reprojection error formulation from [17]

$$\mathbf{e}^{i,j,k}_{reproj}(\mathbf{x}_r, \mathbf{x}_l, \mathbf{z}) = \mathbf{z}^{i,j,k} - \mathbf{h}_i(\mathbf{T}^{C_i}_{B_k} \mathbf{T}^{B_k}_{G} \mathbf{l}^G_j) \in \mathbb{R}^2, \tag{5}$$

where $\mathbf{z}^{i,j,k}$ is the observation of landmark j in camera i at camera frame k in image coordinates and $\mathbf{h}_i(\cdot)$ is the camera projection model. The transformation from the body to the camera frames $\mathbf{T}^C_{B_k}$ was obtained by offline calibration and we assume a rigid sensor rig, i.e. $\mathbf{T}^C_{B_k} \equiv \mathbf{T}^C_B$.

[1] I.e. every value node is *always* connected to one or multiple factor nodes and vice versa [15].

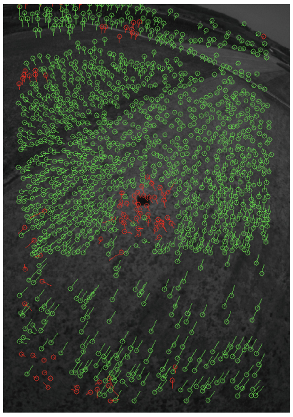
IMU factor. The IMU measurements are pre-integrated and summarized in a single relative motion constraint connecting two time-consecutive poses as described in [18,19]. The factor evaluates the residuals and Jacobians for the pose, velocity and IMU biases of the previous and current state.

2.5 Sliding Window Estimator (SWE)

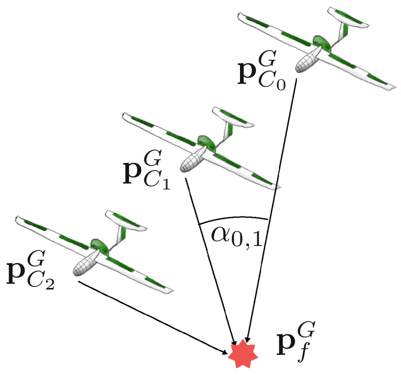
The outline of the sliding window estimator is presented in Algorithm 1. The most relevant parts of the framework include feature extraction, landmark initialization and graph optimization: Feature tracks are generated using a Lucas-Kanade tracker with gyroscope-based feature prediction for speed-up and feature bucketing as well as two-point translation-only RANSAC outlier rejection as visualized in Fig. 2a.

Landmark Initialization. Only well constrained features are used as potential landmarks. We define the quality of a landmark observation by the number of observations, the minimum and maximum distance and by the angle between the incident rays:

$$\begin{aligned}
d_{min} &= \min(d_{min}, ||\mathbf{p}_{C_i}^G - \mathbf{p}_f^G||_2),\ \ i \in [0, M) \\
d_{max} &= \max(d_{max}, ||\mathbf{p}_{C_i}^G - \mathbf{p}_f^G||_2),\ \ i \in [0, M) \\
\alpha_{min} &= \min(\alpha_{min}, \cos^{-1}(||\mathbf{p}_{C_i}^G - \mathbf{p}_f^G||_2 \cdot ||\mathbf{p}_{C_j}^G - \mathbf{p}_f^G||_2)),
\end{aligned}$$



(a) The image visualizes the output of the vision front-end: Inlier feature tracks are shown in green. Tracks classified as outliers by RANSAC are visualized in red - here mainly due to the UAV's shadow violating the static feature assumption.



(b) Definition of well-constrained landmarks illustrated by the incident rays of three landmark observations. The following heuristics are used for the UAV SKATE datasets: $M_{min} = 30$, $d_{min} = 0.1\,\text{m}$, $d_{max} = 100\,\text{m}$, $\alpha_{min} = 0.5\,\text{deg}$.

Fig. 2. Vision front-end: Feature tracking and landmark initialization. (Color figure online)

where $i \in [0, M)$, $j \in [i+1, M)$ and M represents the number of landmark observations. The basic idea as well as the used heuristics during flight are presented in Fig. 2b.

After triangulating the feature tracks the estimated landmark locations are validated by back-projecting the landmarks in the frames from which they were observed. The landmark is only inserted as a state if it lies in the visible camera cone. That is, e.g. landmarks that are triangulated behind the cameras are rejected.

Graph Optimization. Every factor and value node is associated with a marginalization strategy:

- Robot states: Marginalized after the graph update if the corresponding value node falls outside the sliding window.
- Landmarks: Landmark nodes are marginalized before the graph update if
  1. *opportunistic feature:* the feature track has been terminated in the previous camera frame due to the temporal condition or the feature has not been re-detected. Opportunistic features ensure that every frame is connected to *sufficient* reprojection factors.
  2. *persistent feature:* the feature has not been re-detected in the previous camera frame. Persistent features are included in the factor graph to reduce temporal drift.

A toy example of the marginalization strategy with a sliding window of $N = 4$ frames is presented in Fig. 3. In order to marginalize variable nodes the factors associated with these variables are identified and removed. The marginalized factors are replaced by the linearized factors which corresponds to a transformation into a Bayes tree. Before marginalization, the variables in the graph need to be reordered to preserve sparsity: For this purpose we use the column approximate minimum degree heuristic (COLAMD) [20]. The graph is linearized using

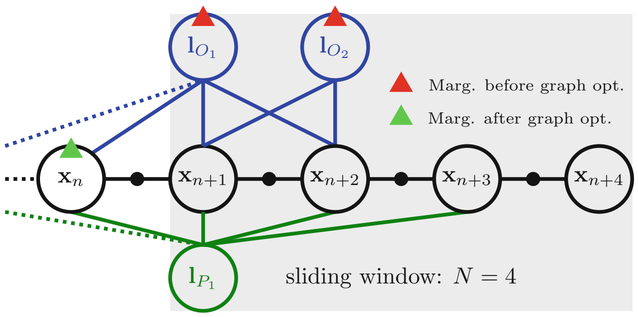


Fig. 3. Marginalization strategy: At step $k = n + 3$ the VI-node is propagated and the graph is augmented with the new state $\mathbf{x}_{n+4}$. Due to the temporal condition (TC) the value node $\mathbf{x}_n$ is to be marginalized. Also, the opportunistic landmark nodes $\mathbf{l}_{O_1}$ (TC) and $\mathbf{l}_{O_2}$ (not re-detected) are to be marginalized.

Algorithm 1. Estimator pipeline (running)

```
1: Extract opportunistic and persistent feature tracks:
     – Lucas-Kanade feature tracker
     – Gyroscope measurement integration for feature prediction
     – Feature bucketing
     – Two-point RANSAC outlier rejection
2: Initialize landmarks:
     – Check if landmark is well constrained
     – Triangulate feature tracks
     – Check if triangulated landmark location is in visible camera cone
3: Add landmarks to factor graph (Fig. 4, b)
4: Detect stationary phases:
     – Add velocity priors if stationary phase detected (take-off, landing)
5: Update factor graph
     – Apply marginalization strategy
     – Linearize and optimize factor graph
     – Detect and remove outliers
6: Pre-integrate the IMU measurements and propagate the current VI-node
7: Augment the factor graph with the propagated VI-node (Fig. 4, c)
8: Add the IMU factor to the factor graph (Fig. 4, c)
```

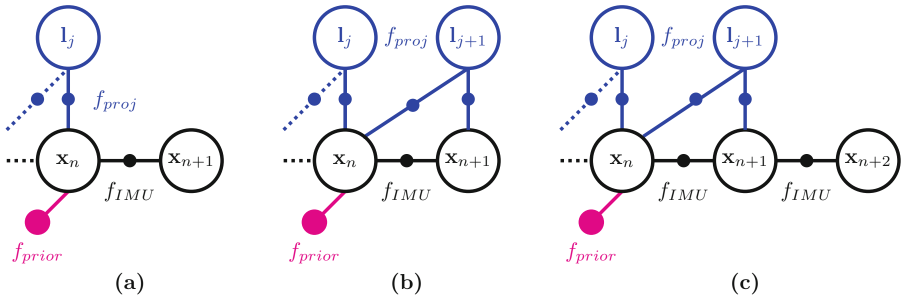


Fig. 4. Factor and value node insertion into the factor graph. **(a)** Initial setup at $k = n+1$. **(b)** Insertion of landmark values and reprojection factors. **(c)** Pre-integration of IMU measurements and graph augmentation.

Cholesky factorization and optimized using Levenberg-Marquardt. For marginalization and optimization of the factor graph we use the *Georgia Tech Smoothing and Mapping* (GTSAM) [21] framework.

Estimator Initialization. For coarse gravity alignment, the accelerometer readings $\mathbf{a}^B$ are averaged over a short static period before take-off. The calculations are performed in axis-angle notation:

$$(axis, angle)_B^G = (-W_g \times \|\bar{\mathbf{a}}^B\|, \cos^{-1}(W_g^T \|\bar{\mathbf{a}}^B\|)) \tag{6}$$

with $W_g = \left[0.0\ 0.0\ 1.0\right]^T$. As noted in [14], a purely visual SLAM problem has six degrees of freedom (DoF). The visual-inertial problem has only four DoF since gravity renders two additional DoF observable (around world x- and y-axis). Based on these calculations the initial transformation $T_{B_0}^{G}$ is estimated and the estimator switches to the routine described in Algorithm 1.

3 Experimental Results

In this section we outline the hardware setup and present the results generated by the real-time visual-inertial navigation framework.

3.1 Sensor and Processing Unit

The sensor and processing unit used for the experiment is shown in Fig. 1: It is equipped with an Aptina MT9V034 grayscale global shutter camera which is able to record images at up to 60 fps with a resolution of 752×480 pixels. The MEMS inertial measurement unit (IMU) ADIS 16448 measures angular velocities as well as linear accelerations. The camera and IMU are integrated into an ARM-FPGA-based Visual-Inertial (VI) sensor system [22] that time-synchronizes IMU and camera on a hardware level. The Intel Atom CPU has access to the synchronized visual-inertial data stream. All components are mounted on an aluminium frame that guarantees a rigid camera-IMU transformation throughout the flight. For more details we refer to [23]. The camera intrinsics (i.e. focal length and principal point) as well as extrinsics (i.e. camera-IMU transformation T_C^B) are estimated offline using the calibration tool *Kalibr* [14].

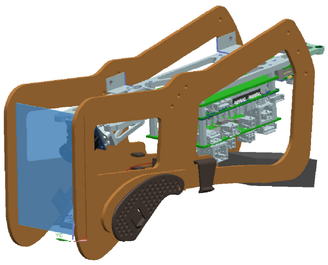
3.2 Platform: Aurora SKATE

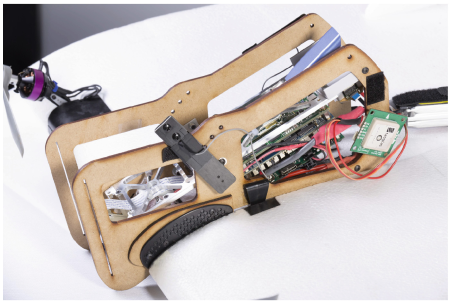
For demonstrating visual-inertial navigation, the sensor & processing unit is mounted on the small fixed-wing UAV SKATE. The light flying wing shown in Fig. 5 was manufactured by *Aurora*, features two independently articulated propulsion units and is controlled by an elevator control surface. Using thrust vectoring it is able to rapidly transition between vertical and horizontal flight and thus provides high agility and maneuverability. Different control modes (auto, manual) can be set by the hand-held remote control and allow for easy and safe operation of the vehicle. Note that the presented flight was performed in manual mode. The original processing unit shown in Fig. 5 is replaced by an interface specifically designed for this carrier. The interface mounts the sensor pod safely, powers it, is rigid and easily removable at the same time.

3.3 Simultaneous State Estimation and Sparse Map Generation

This section presents the state estimates and sparse point-cloud generated by the proposed nonlinear estimation framework based on datasets recorded by the small unmanned aerial vehicles SKATE. The test site is characterized by flat

(a) SKATE by Aurora in factory configuration. The head piece contains batteries and two down-looking cameras.


(b) CAD model of sensor & processing unit as well as its power interface to SKATE. The sensors consist of an IMU and a gray-scale camera.


(c) Sensor & processing unit mounted on SKATE. The measurements from the GPS receiver are used to generate a reference trajectory.

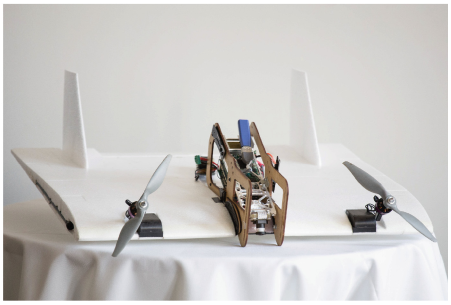

(d) Sensor & processing unit mounted on SKATE.

Fig. 5. Hardware modifications made to SKATE to allow for accurate and time-synchronized visual-inertial localization and mapping.

terrain with few three-dimensional structure which makes the underlying scale estimation challenging.

Figure 6 shows the state estimates of the robot's position, velocity and IMU biases. The estimated altitude after the vehicle has landed is used as a first validation method. For this flight we registered a final altitude of around 3.1 m which demonstrates the limited translational drift. Figure 7 represents the sparse point-cloud and the estimated trajectory flown by the fixed-wing UAV. In particular take-off and landing spot show a high point density which enables automatic take-off and landing procedures. To obtain a further quantitative accuracy measurement, an error is defined as the Euclidean distance to the GPS position obtained from a uBlox LEA-6H sensor. For the complete trajectory, from take-off to landing, the RMSE amounts to 14.6 m with respect to the aligned GPS based reference trajectory. Both trajectories are shown in Fig. 8: Especially at low altitudes, the GPS measurements show large jumps due to multi-path effects while our proposed visual-inertial navigation system produces smooth estimates, including the take-off and landing phase. The runtime evaluated for the

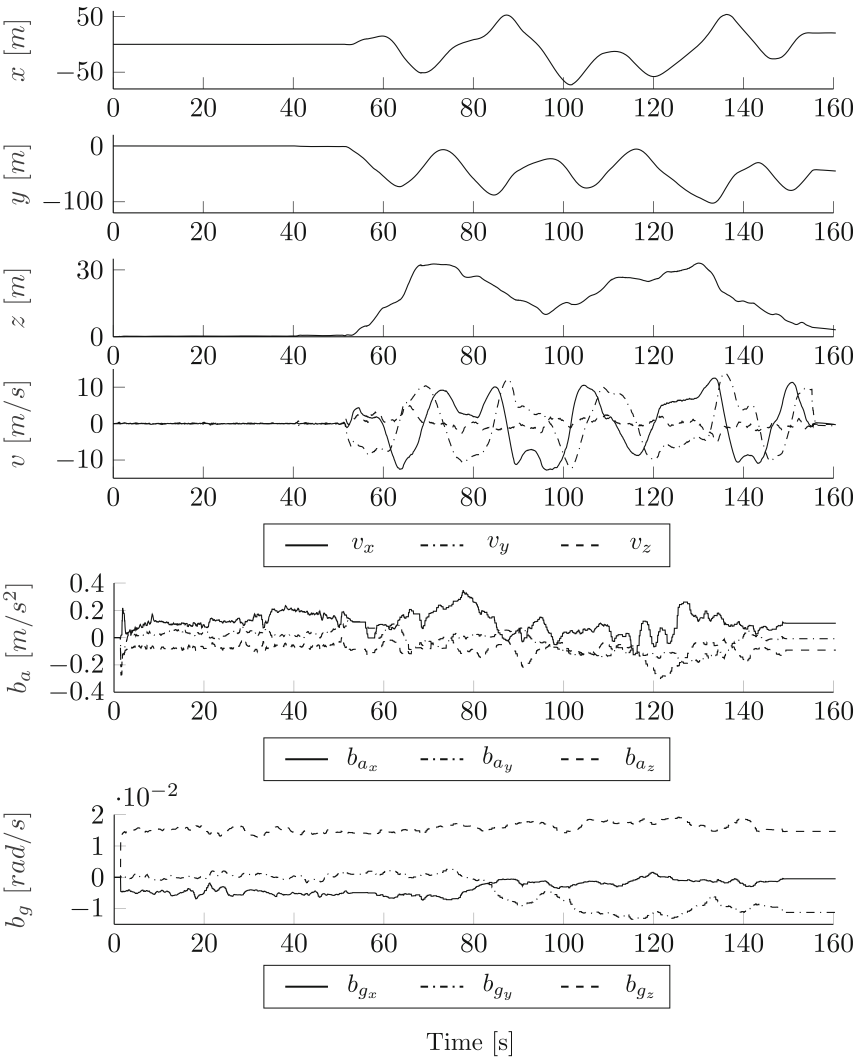


Fig. 6. State estimates of robot's position, velocity, accelerometer as well as gyroscope biases.

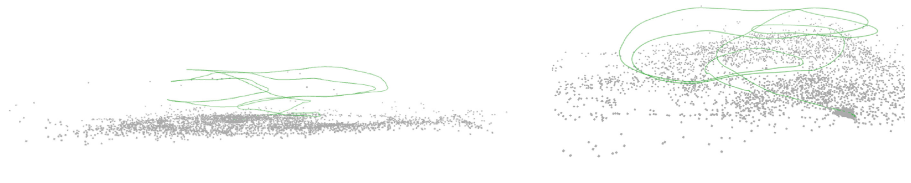
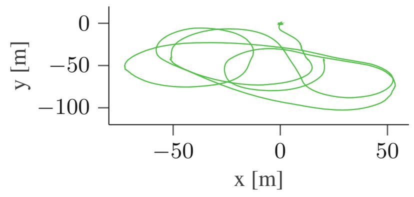
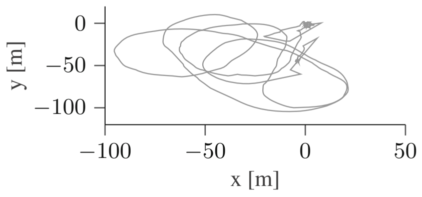


Fig. 7. Side-view of the generated sparse point-cloud and the estimated trajectory flown by the UAV. The right figure shows take-off and landing spot with increased point-cloud density.



(a) Top-view of the estimated trajectory in local vision frame. The position is set to zero during initialization. The trajectory shows smooth estimates during take-off and landing phase.



(b) Top-view of the GPS trajectory in local UTM coordinates where the first GPS measurement defines the origin. Especially at low altitudes, the GPS measurements show large jumps due to multi-path effects.

Fig. 8. Comparison of estimated trajectory and GPS based reference trajectory.

Table 1. Runtime in ms for one frame.

	Mean [ms]	Std. dev. [ms]
Feature tracking*	19.976	7.267
Landmark initialization	0.269	0.113
State augmentation	3.455	2.497
Variable reordering (COLAMD)	0.207	0.079
Marginalization	4.321	0.935
Graph optimization	27.370	17.325
Frame processing	42.249	21.805

*Runs in a separate thread and does not count towards total frame processing time.

SKATE mission is shown in Table 1 and was performed on an Intel(R) Core(TM) i7-4800MQ CPU @ 2.70 GHz. The profiling indicates that a camera stream of 23 Hz can be processed by the estimator in real-time.

4 Conclusion and Future Work

This paper presented an inverse-form visual-inertial navigation system that fuses inertial measurements and visual cues into a sliding window estimator applied to a small fixed-wing unmanned aerial vehicle. The state estimates of the robot's pose, velocity, IMU biases, and landmarks which were generated from a flight are presented and the trajectory is compared to a GPS based reference solution. The experiments showed that the locally consistent sparse point-cloud can be employed for static obstacle avoidance in terms of automatic take-off and landing for fixed-wing UAVs. Future work in the context of visual-inertial SLAM for fixed-wing UAVs will comprise further drift reduction by means of vanishing point detection, horizon tracking and loop closure. Furthermore, the experiments showed that the accuracy of the reference trajectory needs to be enhanced, e.g. by using DGPS, to allow for a more expressive quantitative evaluation.

Acknowledgements. The research leading to these results has received funding from the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°600958 (SHERPA) and was sponsored by Aurora Flight Sciences.

References

1. Nerurkar, E., Wu, K., Roumeliotis, S.: C-KLAM: constrained keyframe-based localization and mapping. In: ICRA (2014)
2. Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., Furgale, P.: Keyframe-based visual-inertial odometry using nonlinear optimization. In: IJRR (2015)
3. Li, M., Mourikis, A.I.: Optimization-based estimator design for vision-aided inertial navigation. In: RSS (2013)
4. Mourikis, A.I., Roumeliotis, S.I.: A multi-state constraint Kalman filter for vision-aided inertial navigation. In: ICRA (2007)
5. Huang, G.P., Mourikis, A.I., Roumeliotis, S.I.: An observability-constrained sliding-window filter for SLAM. In: IROS (2011)
6. Hesch, J.A., Kottas, D.G., Bowman, S.L., Roumeliotis, S.I.: Camera-IMU-based localization: observability analysis and consistency improvement. In: IJRR (2014)
7. Martinelli, A.: Visual-inertial structure from motion: observability vs. minimum number of sensors. In: ICRA (2014)
8. Li, M., Kim, B.H., Mourikis, A.I.: Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera. In: ICRA (2013)
9. Mourikis, A.I., Roumeliotis, S.I.: A dual-layer estimator architecture for long-term localization. In: CVPRW (2008)
10. Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., Furgale, P.: Keyframe-based visual-inertial SLAM using nonlinear optimization. In: IJRR (2014)
11. Sibley, G., Matthies, L., Sukhatme, G.S.: Sliding window filter with application to planetary landing. In: JFR (2010)
12. Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J., Dellaert, F.: iSAM2: incremental smoothing and mapping using the Bayes tree. In: IJRR (2012)
13. Chiu, H.P., Williams, S., Dellaert, F., Samarasekera, S., Kumar, R.: Robust vision-aided navigation using sliding-window factor graphs. In: ICRA (2013)
14. Furgale, P., Rehder, J., Siegwart, R.: Unified temporal and spatial calibration for multi-sensor systems. In: IROS (2013)
15. Grisetti, G., Kummerle, R., Stachniss, C., Burgard, W.: A tutorial on graph-based SLAM. In: ITSM
16. Leutenegger, S.: Unmanned solar airplanes. Ph.D. thesis, Dissertion, ETH Zürich, Nr. 22113 (2014)
17. Furgale, P.: Extensions to the visual odometry pipeline for the exploration of planetary surfaces. University of Toronto (2011)
18. Forster, C., Carlone, L., Dellaert, F., Scaramuzza, D.: IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In: RSS (2015)
19. Carlone, L., Kira, Z., Beall, C., Indelman, V., Dellaert, F.: Eliminating conditionally independent sets in factor graphs: a unifying perspective based on smart factors. In: ICRA (2014)

20. Davis, T.A., Gilbert, J.R., Larimore, S.I., Ng, E.G.: A column approximate minimum degree ordering algorithm. ACM Trans. Math. Softw. **30**(3), 353–376 (2004)
21. Dellaert, F.: Factor graphs and GTSAM: a hands-on introduction. Technical report GT-RIM-CP&R-2012-002, GT RIM (2012)
22. Nikolic, J., Rehder, J., Burri, M., Gohl, P., Leutenegger, S., Furgale, P.T., Siegwart, R.: A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM. In: ICRA (2014)
23. Oettershagen, P., Stastny, T.J., Mantel, T., Melzer, A., Rudin, K., Agamennoni, G., Alexis, K., Siegwart, R.: Long-endurance sensing and mapping using a hand-launchable solar-powered UAV. In: FSR (2015)