Duo-VIO: Fast, Light-weight, Stereo Inertial Odometry

Nicolas de Palézieux[1], Tobias Nägeli[1], Otmar Hilliges[1]

***Abstract*— We present a Visual Inertial Odometry system that enables the autonomous flight of Micro Aerial Vehicles in GPS denied and unstructured environments. The system relies on commercially available and affordable hardware both for sensing and computation. The algorithm runs in real time on an ARM based embedded micro-computer on-board an MAV. In experiments, we demonstrate the performance of the system both indoors and outdoors, in hand held an in-flight scenarios. The achieved accuracy of the experiments is competitive with other research which uses custom designed hardware and desktop-grade processors.**

I. Introduction

Estimating the motion of a robot relative to a 3D scene from a set of sensor readings such as camera images or inertial measurements is one of the fundamental problems in computer vision and robotics. While recently impressive progress in camera pose estimation has been shown for hand-held cases (often relying on desktop grade compute resources), estimating the position dynamics of small agile robots remains a challenging and unsolved research problem. In particular, quadrotors and other small flying robots render many current approaches infeasible because they move fast, produce significant high-frequency accelerations (impacting SNR on inertial measurements) and impose tight bounds on payload and battery-lifetime which limit computational resources available for on-board processing.

In this paper we present a Visual-Inertial Odometry (VIO) algorithm that has been purposefully designed for the usage on small aerial vehicles (MAVs). A major draw of the presented system is that it is designed to run on affordable and off the self hardware. The algorithm runs at 100Hz on a low-power ARM CPU and works with forward-facing cameras, allowing for fast flight and removing the need for a second camera for collision avoidance. Furthermore, the algorithm provides accurate metric scale estimates without requiring specific initialization. We detail the algorithm here and release the code as open-source software.

There is a vast body of literature on camera pose estimation, we concentrate our discussion on approaches of particular interest in the context of small, agile robots. One of the first flying robots leveraging vision was shown in [1, 2], using two cameras to estimate the 6 degrees of freedom necessary for flight stabilization. Frauendorfer et al. [3] used a downward looking camera for optical flow based flight stabilization and an additional stereo camera pair for collision avoidance, mapping and short horizon path planning. Others have used the PTAM algorithm [4] directly to fly with a downward-looking monocular camera [5] or leveraged modified versions [6, 7] in conjunction with off-board processing.

[1]Advanced Interactive Technologies Lab, Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland depnicol|naegelit|otmar.hilliges@inf.ethz.ch

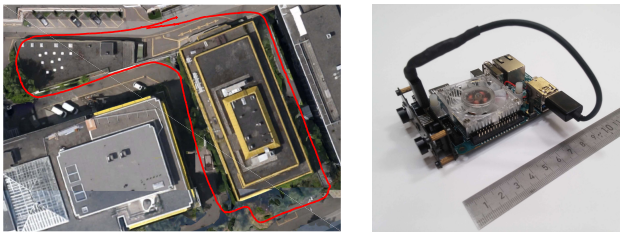


Fig. 1: Left: Outdoor trajectory as estimated with our method (length: $230m$, 2.9% drift). Right: Our hardware consisting of a low-cost stereo sensor and a single board ARM PC, weighing less than $100g$ total.

Key-frame based stereo approaches together with a loosely coupled IMU integration have been proposed to overcome the scale drift problem [8]. Recently a number of approaches have been proposed that directly use dense surface measurements instead of extracted visual features, e.g. [9]. Similarly, Forster et al. propose a sparse direct methods approach [10]. All of these methods use key-frames and hence require an explicit initialization phase to estimate scale from IMU data.

This initialization requirement can be circumvented using probabilistic approaches such as the Extended Kalman Filter (EKF), first applied to camera pose estimation in [11, 12]. In aerospace engineering, the indirect or error-state Kalman Filter has been introduced by Lefferts [13] and reintroduced by Roumeliotis et al. [14]. Similar approaches to visual odometry exist such as the Multi State Constrained Kalman Filter (MSCKF) [15], or hybrid versions [16, 17, 18]. EKF-based approaches have also been combined with direct photometric methods [19, 20].

A. System Overview

The main goal of this work is the robust, fast and accurate estimation of the pose of a quadrotor without external sensing infrastructure, such as GPS or markers on the ground. Therefore, an important design goal was to perform all sensing and computing on-board, leveraging readily procurable off-the-shelf hardware only. Furthermore, due to the power constraints on small robots the algorithm needs to be computationally efficient. To fulfill these constraints we contribute three main aspects:

a) Stereo-Initialization, Monocular Tracking: We use a small baseline stereo camera with an integrated Inertial Measurement Unit (IMU). Stereo measurements are only

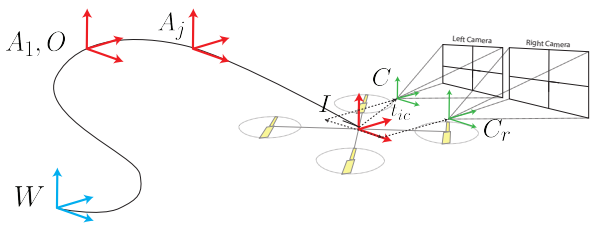


Fig. 2: The coordinate frames used in the Kalman Filter framework. The camera and anchor poses are expressed in the origin frame O, which coincides with the pose of an anchor, here A_1.

used to initialize the depth of features, while tracking over time is performed monocularly. This combines the efficiency of monocular approaches with reliable scale estimation via stereo.

b) Iterated Error State Kalman Filter: An IESKF fuses IMU measurements and feature observations extracted from the camera images to estimate the position, orientation, and velocity of the camera. Furthermore we estimate additive IMU biases and a compact 3D map of feature point locations.

c) Anchor-centric Parameterization: Feature points are parameterized by their inverse depth on an iteratively updated small set of past camera poses. This set of pose estimates are expressed in a reference frame that is moved along with the current frame, keeping the camera pose and map uncertainty bounded and hence reducing drift over time.

II. Preliminaries

A. Notation and Coordinate Frames

The following coordinate frames are used throughout this paper and are illustrated in Fig. 2: W – the inertial world frame; O – the origin frame; A_j – anchor frames; C – the camera frame; I – the IMU frame.

We follow the standard notation proposed in literature. Translation vectors between two frames A and B, expressed in frame A, are denoted by $\boldsymbol{t}_{AB}$. Rotation matrices performing rotations from frame A to frame B are denoted by $\boldsymbol{R}_{BA} = \boldsymbol{R}(\bar{q}_{BA}) \in SO(3)$, where $\bar{q}_{BA}$ is the corresponding quaternion. We adhere to the JPL quaternion definition [21] and denote a quaternion by $\bar{q} = [q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k} + q_w] = [\boldsymbol{q}, q_w]^T$. Quaternion multiplication is denoted by $\otimes$.

Expected or estimated values of a variable x are denoted by $\mathbb{E}[x] = \hat{x}$, errors are written as δx. Orientation errors are described in $so(3)$, the tangent space of $SO(3)$, and are written as $\delta\theta$. Measurements of a quantity x affected by white Gaussian noise are written as $z = x + \nu$ with $\nu \sim \mathcal{N}(\sigma)$.

B. Error State Kalman Filter

A quaternion uses 4 dimensions to describe 3 degrees of freedom. Because of this, the 4×4 covariance matrix of an estimated quaternion is singular. This issue is avoided with the *Error State Kalman Filter* (ESKF) formulation [13].

We define the error of an orientation using an error quaternion $\delta\bar{q}_{A\hat{A}}$, a small rotation between the estimated ($\hat{A}$) and true (A) orientation. The error quaternion can be assumed to be small and hence the small angle approximation holds $\delta\bar{q}_{A\hat{A}} \approx \begin{bmatrix}\frac{1}{2}\delta\boldsymbol{\theta} & 1\end{bmatrix}^T$ [22]. Using $\delta\boldsymbol{\theta}$ to represent orientations in the Kalman filter reduces their dimensionality to 3. This is both computationally advantageous and circumvents the singularity issues with a 4×4 orientation covariance matrix.

In the ESKF, the *error state* $\delta\boldsymbol{x}$ is the quantity being estimated and the covariance matrix $\boldsymbol{P}$ describes its uncertainty. The *total state* $\hat{\boldsymbol{x}}$ is always updated such that the expected value of the error state $\mathbb{E}[\delta\boldsymbol{x}] = \mathbf{0}$. For further information the reader is referred to a very good introduction in [14].

C. Modeling

The state space of the IESKF consist of two parts, the Camera state and the Map state.

a) Camera State: The camera state describes the camera's estimated pose (position and orientation) and velocity, as well as the estimated accelerometer and gyroscope biases, denoted by $\boldsymbol{b}_a \in \mathbb{R}^3$ and $\boldsymbol{b}_\omega \in \mathbb{R}^3$, respectively. Further, the orientation of the origin frame in the inertial world frame, $\bar{q}_{OW} \in \mathbb{R}^4$, is included. The purpose of this additional orientation is explained in Sec III-B.

$$\boldsymbol{x}_c = \begin{bmatrix}\boldsymbol{t}_{OC},\ \bar{q}_{CO},\ \boldsymbol{v}_{OC},\ \boldsymbol{b}_a,\ \boldsymbol{b}_\omega,\ \bar{q}_{OW}\end{bmatrix}^T \in \mathbb{R}^{20}$$

b) Map State: We parameterize the map of feature points by their inverse depths from the camera pose at which they are first seen. These past poses are termed anchor poses and denoted by $\left(\boldsymbol{t}_{OA_j} \in \mathbb{R}^3,\ \bar{q}_{A_jO} \in \mathbb{R}^4\right)$. The unit norm vector $\boldsymbol{m}_i$ encoding the ray in the anchor frame on which a feature i lies is stored statically for each map feature.

By bundling several map features to the same anchor pose, a very efficient map state is achieved [23, 19]. The total map state $\boldsymbol{x}_m$ is composed of l anchor states:

$$\boldsymbol{x}_m = \begin{bmatrix}\boldsymbol{x}_{A_1}, & \ldots, & \boldsymbol{x}_{A_l}\end{bmatrix}^T \in \mathbb{R}^{(7+n)l}$$
$$\boldsymbol{x}_{A_j} = \begin{bmatrix}\boldsymbol{t}_{OA_j}, & \bar{q}_{A_jO}, & \rho_1, & \ldots, & \rho_n\end{bmatrix}^T, \in \mathbb{R}^{(7+n)}$$

The total state of the IESKF has thus the following form:

$$\boldsymbol{x} = [\boldsymbol{x}_c,\ \boldsymbol{x}_m]^T \in \mathbb{R}^{20+(7+n)l \times 20+(7+n)l}$$

Due to the error state formulation, the covariance matrix needed to estimate the camera pose and l anchors, each with n features is $\boldsymbol{P} \in \mathbb{R}^{18+(6+n)l \times 18+(6+n)l}$.

III. Algorithm

We now discuss the most important aspects of the proposed algorithm.

A. Feature Initialization

Upon filter initialization or once features can no longer be tracked, new features need to be inserted into the state space. For this, salient features are extracted from both the left and right camera image and their inverse depth is initialized by triangulation in a least squares fashion. Together with a new anchor pose $\left(\boldsymbol{t}_{OA_j}\ \bar{q}_{A_jO}\right)$, corresponding to the current camera pose estimate $\boldsymbol{t}_{OA_j} = \boldsymbol{t}_{OC}$ and $\bar{q}_{A_jO} = \bar{q}_{CO}$, these point estimates are added to the state space.

To capture that the new anchor pose estimate is identical to the current camera pose, the covariance matrix is updated with the Jacobian of the insertion function.

$$\boldsymbol{P}^{+} = \boldsymbol{J}\boldsymbol{P}^{-}\boldsymbol{J}^{T} \qquad \boldsymbol{J} = \frac{\partial \delta \boldsymbol{x}^{+}}{\partial \delta \boldsymbol{x}^{-}}\Big|_{\delta \boldsymbol{x}^{-}=\boldsymbol{0}},$$

where $(\cdot)^{-}$ and $(\cdot)^{+}$ denote the time instances before and after the insertion.

Finally, the inverse depth uncertainties $\sigma_{\rho_{init}}$ are inserted to the covariance matrix for each new feature. $\sigma_{\rho_{init}}$ can be computed, given known camera intrinsics and extrinsics. Due to the inverse depth parameterization, $\sigma_{\rho_{init}}$ does not depend on the feature's depth.

B. Anchor-centric Estimation

In traditional approaches camera and feature locations are estimated relative to a global world reference frame and hence the uncertainty of the (unobservable) absolute camera position grows unbounded. This is detrimental to the filter performance, as large uncertainties result in large linearization errors in both the Kalman Filter propagation and update step [24]. Our approach circumvents this issue by marginalizing the unobservable component of the global position uncertainty out of the state space – by moving a relative reference frame with the current camera pose estimate. This improves the filter performance as less linearization error is incurred due to bounded uncertainty.

This bears similarity to so-called robo-centric EKF formulations [25, 26, 24, 20]. An important difference is that in our anchor-centric approach, the reference frame O, which is chosen to coincide with one of the anchor poses, is only updated when the corresponding anchor pose is removed from the state space (see Fig. 2), rather than updating it on every iteration as is the case in robo-centric approaches. This is both computationally more efficient and reduces drift, as shown in experiments (see Sec IV-B).

When the current anchor frame is removed from the state space, O is moved to coincide with the anchor frame with the lowest uncertainty, denoted by A_O.

$$A_O \in A_1, ..., A_l \quad \text{s.t.} \quad \|\boldsymbol{P}(A_O)\| \leq \|\boldsymbol{P}(A_j)\|, \quad j = 1, ..., l,$$

where $\|\boldsymbol{P}(A_j)\|$ denotes the matrix 2-norm of the entries of the covariance matrix $\boldsymbol{P}$ pertaining to anchor A_j. The relative translation and rotation between the old origin frame and the new one is given by $\boldsymbol{t}_{O_\kappa O_{\kappa+1}} = \hat{\boldsymbol{t}}_{OA_O}$ and $\bar{q}_{O_{\kappa+1}O_\kappa} = \hat{\bar{q}}_{A_O O}$, respectively, where κ and $\kappa + 1$ denote the time instances before and after the move, respectively.

They are used to transform the camera pose and velocity as well as the anchor poses into the new origin frame. The bias states, $\boldsymbol{b}_a$, $\boldsymbol{b}_\omega$, and inverse depths ρ_i do not depend on the origin frame and therefore do not need to be transformed.

As the absolute position of the origin frame O in the world frame W is not observable, estimating this translation does not improve the performance of the filter. Therefore it is not part of the state space, but is stored statically and updated only when the origin frame is moved.

The orientation of the origin frame in the world frame, $\bar{q}_{OW}$, however, is included in the state space, as its roll and the pitch axes are observable through the gravity measurement from the IMU. Estimating these two components of the origin orientation allows for the pose estimate and the map to become aligned with gravity.

Whenever the O is moved, the covariance matrix is updated using the Jacobian of the transformation function.

C. IESKF State Propagation

The estimated state is propagated whenever measurements from the IMU become available. These measurements are affected by process noise and bias. Following [15, 19], the gyroscope and accelerometer process noise, denoted by $\boldsymbol{n}_g$ and $\boldsymbol{n}_a$, respectively, are modeled as white Gaussian noise processes with respective variances $\boldsymbol{\sigma}_a$ and $\boldsymbol{\sigma}_g$. The biases are modeled as random walks $\dot{\boldsymbol{b}}_\omega = \boldsymbol{n}_{\boldsymbol{b}_\omega}$, $\dot{\boldsymbol{b}}_a = \boldsymbol{n}_{\boldsymbol{b}_a}$, where $\boldsymbol{n}_{\boldsymbol{b}_\omega}$ and $\boldsymbol{n}_{\boldsymbol{b}_a}$ are zero mean white Gaussian noise processes.

The IMU measurements and camera dynamics are modeled as in [22]. The dynamics of the camera state are discretized with a zero order hold strategy and are propagated using the expected values of the linear acceleration and rotational velocity. The map is assumed to be static. Thus, it remains unchanged in the propagation.

Note that, after propagating the total state with the IMU measurements, the expected error state is still zero.

The covariance matrix is propagated using the Jacobians of the error state dynamics [27], taken with respect to the error state $\delta\boldsymbol{x} = \boldsymbol{0}$ and the process noise $\boldsymbol{n} = \boldsymbol{0}$:

$$\boldsymbol{F} = \frac{\partial \dot{\delta \boldsymbol{x}_c}}{\partial \delta \boldsymbol{x}_c}\Big|_{\substack{\delta\boldsymbol{x}=\boldsymbol{0} \\ \boldsymbol{n}=\boldsymbol{0}}} \qquad \boldsymbol{G} = \frac{\partial \dot{\delta \boldsymbol{x}_c}}{\partial \boldsymbol{n}}\Big|_{\substack{\delta\boldsymbol{x}=\boldsymbol{0} \\ \boldsymbol{n}=\boldsymbol{0}}} \tag{1}$$

The covariance matrix is propagated using zero order hold discretization of the Jacobian $\boldsymbol{F}$ and the process noise [28].

D. IESKF State Update

A state update is performed whenever image data becomes available. First, all currently estimated features are tracked from the previous to the current image of the left camera using the KLT tracker implemented in OpenCV[1].

1) Outlier Rejection: Features may be badly tracked due to e.g. specular reflections or moving objects, necessitating the detection and rejection of these outliers. We apply two methods of outlier rejection consecutively.

a) 1-Point RANSAC: This consensus based method builds on the standard RANSAC algorithm by taking into account prior information about the model, dramatically reducing the computational complexity of the algorithm [26].

The residual of a randomly selected feature is computed by predicting the measurement according to the *a priori* state estimate:

$$\boldsymbol{r}_i = \boldsymbol{z}_i - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, i), \tag{2}$$

[1] www.opencv.org/

where $\boldsymbol{h}(\boldsymbol{x}, i)$ is the map from the total state to image coordinates [29]. An intermediate total state is then computed:

$$\boldsymbol{K}_{hyp} = \boldsymbol{P}\boldsymbol{H}_i^T \boldsymbol{S}_i^{-1} \tag{3a}$$

$$\delta\boldsymbol{x}_{hyp}^{apo} = \boldsymbol{K}_{hyp}\boldsymbol{r}_i \tag{3b}$$

$$\hat{\boldsymbol{x}}_{hyp} = \hat{\boldsymbol{x}}_{k|k-1} \boxplus \delta\boldsymbol{x}^{apo}, \tag{3c}$$

where $\boldsymbol{H}_i$ is the Jacobian of (2) taken with respect to $\delta\boldsymbol{x}$, linearized around its expected value, $\mathbb{E}[\delta\boldsymbol{x}] = \boldsymbol{0}$, and $\boldsymbol{S}_i = \boldsymbol{H}_i\boldsymbol{P}\boldsymbol{H}_i^T + \boldsymbol{R}_i$ the measurement innovation. $\boxplus$ denotes the fusion of the *a priori* total state and the *a posteriori* error state. Linear quantities are updated additively, while rotational entries are updated multiplicatively.

Features which now have a small residual are considered inliers of this hypothesis. Equations (2) and (3) are applied repeatedly with different measurements. The iteration is stopped according to standard RANSAC criteria about the expected inlier ratio [30].

Once the algorithm has terminated, we have a set of *low innovation inliers*. The complementary set is termed the set of *high innovation candidates*, which will be further processed as described in the following.

A state update is performed with the low innovation inliers analogously to (3), with the difference that now the residuals of all low innovation inliers are used by stacking them into a column vector. The covariance matrix is updated with the standard Kalman Filter equation:

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\,\boldsymbol{P}_{k|k-1} \tag{4}$$

b) χ^2 Test: Following the state update with the low innovation inliers, the high innovation candidates are further separated into *high innovation inliers* and outliers by testing their measurement likelihood:

$$\chi_i^2 = \boldsymbol{r}_i^T \boldsymbol{S}_i^{-1} \boldsymbol{r}_i \leq \chi_{thresh}^2 \tag{5}$$

The high innovation inliers are fused into the state estimate as described in the following.

2) Iterated State Update: The Kalman gain $\boldsymbol{K}$ is computed using the linearization $\boldsymbol{H}$ of the measurement model $\boldsymbol{h}(\cdot)$, evaluated at the current state estimate. The computed *a posteriori* error state $\delta\boldsymbol{x}^{apo}$ is thus only a first order approximation of the true error state. The accuracy of the state estimate can be improved by repeatedly performing an update with a set of measurements. This is particularly the case for features with a high innovation. Therefore we apply an iterated state update according to Algorithm 1 [28, 31] with the high innovation inliers.

The iteration is stopped if a maximum number of iterations has been reached or when $\delta\boldsymbol{x}^{apo}$ is very small. Note that, irrespective of the number of iterations performed, the covariance matrix $\boldsymbol{P}$ is updated only once.

Algorithm 1 IESKF State Update

Require: Previous state estimate: $\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{P}_{k|k-1}$

$\eta^0 = \hat{\boldsymbol{x}}_{k|k-1}$
$\delta\eta^0 = \boldsymbol{0}$
for $j = 0$ to `max_it` **do**
  $\boldsymbol{r}^j = \boldsymbol{z} - \boldsymbol{h}(\eta^j)$
  $\boldsymbol{H}^j = \frac{\partial \boldsymbol{r}^j}{\partial \delta\boldsymbol{x}}\Big|_{\hat{\boldsymbol{x}}=\eta^j}$
  $\boldsymbol{S}^j = \boldsymbol{H}^j\boldsymbol{P}_{k|k-1}(\boldsymbol{H}^j)^T + \boldsymbol{R}$
  $\boldsymbol{K}^j = \boldsymbol{P}_{k|k-1}(\boldsymbol{H}^j)^T\left(\boldsymbol{S}^j\right)^{-1}$
  $\delta\eta^{j+1} = \boldsymbol{K}^j\left(\boldsymbol{r}^j + \boldsymbol{H}^j\delta\eta^j\right)$
  $\eta^{j+1} = \hat{\boldsymbol{x}}_{k|k-1} \boxplus \delta\eta^{j+1}$
  if $\|\delta\eta^{j+1}\|$ small **then**
    Stop iteration
  end if
end for
$\hat{\boldsymbol{x}}_{k|k} = \eta^{j+1}$
$\boldsymbol{P}_{k|k} = \left(\boldsymbol{I} - \boldsymbol{K}^j\boldsymbol{H}^j\right)\boldsymbol{P}_{k|k-1}$

IV. Experimental Results

A. Hardware Setup

Our localization system consists of a small baseline stereo camera connected to a single-board ARM computer on which the presented algorithm runs. The system is depicted in Fig. 1. Both devices are commercially available and affordable and make for a very small and light-weight system, weighing less than $100g$. Such a small and portable form factor makes the localization system suitable for a large variety of applications, particularly the use on board MAVs designed to fly in the close vicinity of people.

We use a DUO MLX camera by Duo3d[2], featuring two monochrome global shutter cameras with a $30mm$ baseline and a 6 degree of freedom IMU. Inertial measurements are provided at 100 Hz, while the image frame rate is configured at 50 Hz with a resolution of 320x240 pixels.

The VIO algorithm runs on a Hardkernel Odroid XU4[3], equipped with a Samsung Exynos5422 ARM processor.

For flight experiments we mount the VIO system on a Parrot Bebop[4] equipped with a PixFalcon PX4 Autopilot.

B. Experiments

We demonstrate the performance of the presented VIO system with several experiments.

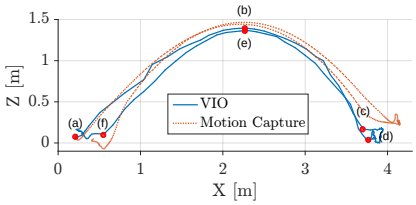
1) Hand-held Accuracy:

TABLE I: Hand-held Trajectory of Length 230 m

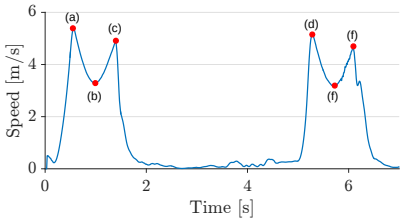
Experiment	1	2	3	4	5	Mean
Rel. drift [%]	2.88	3.57	2.96	3.23	3.43	3.21

As baseline and for comparison with the current state-of-the-art, we evaluate the system's accuracy in a hand-held scenario, where we walk around several buildings, a $230m$ long trajectory, and compute the relative position drift of the trajectory. One such trajectory is shown in Fig. 1. The same experiment is performed several times to assess repeatability. The relative drift of each repetition of the experiment is

[2] www.duo3d.com/product/duo-minilx-lv1
[3] www.hardkernel.com/main/products/prdt_info.php
[4] http://www.parrot.com/products/bebop-drone/



(a) Trajectory estimated by the VIO system compared to motion capture ground truth.



(b) Motion capture ground truth speed.

Fig. 3: Fast motion experiment. The VIO system is thrown over a distance of $4m$. The estimated trajectory is compared to ground truth data from a motion capture system.

shown in Table I. We observe that the system reproducibly estimates its trajectory accurately.

Note that all computations are performed on-board, whereas the literature often reports results from off-board computations.

2) Fast Motions: To evaluate the system's robustness and ability to track fast motions, we throw the system back and forth over a distance of $4m$. The estimate is compared to ground truth form a motion capture system in Fig. 3(a). Fig. 3(b) shows the speed reached by the device. Fig. 4 shows three frames of the experiment with the corresponding time instances labeled in Fig. 3.

The system successfully tracks ego motion even at very high speeds of close to 6 m/s. Despite the high accelerations and motion blur present in this scenario, the estimated trajectory does not deviate significantly from the ground truth. The ability to track fast motions is expected to enable faster flight compared to an optical flow sensor, which is limited to tracking speeds of about $2m/s$ [32].

3) In-Flight Ground Truth Comparison: We demonstrate the performance of the system during flight where a MAV is controlled to repeatedly fly a predefined figure-8 trajectory, using positional information from a motion capture system for reference. We compare the estimated trajectory with the ground truth in Fig. 6. The flown trajectory is $123m$ long and the estimate shows a drift in position of 0.46% and 1.17% yaw drift relative to the ground truth. Drift in y appears significantly bigger than in x, which is due to the fact that positional drift is not independent of drift in yaw. Inspection of Fig. 6 suggests that the under-estimation of the yaw angle causes an under-estimation of the world y position.

In comparing the ground truth trajectory with the estimated motion, this experiment demonstrates that the algorithm correctly estimates the scale of the camera motion while

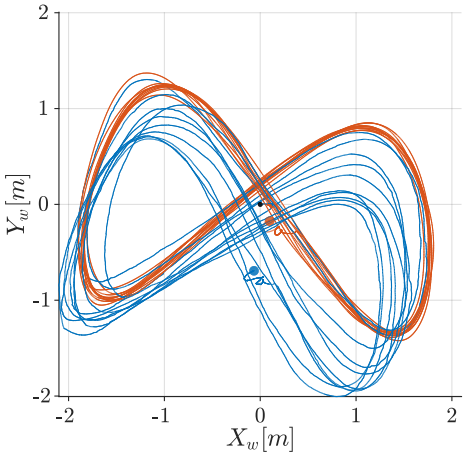


Fig. 6: In flight trajectory estimated by VIO (blue) compared to ground truth (red).

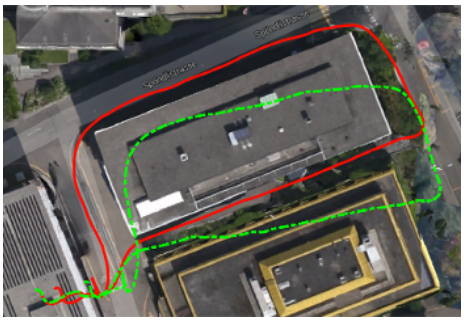


Fig. 7: The trajectory estimated by the anchor-centric parameterization (red) is compared to the estimate with a world centric parameterization (green). The trajectories are aligned with each other and the satellite image at the start.

only using depth information about the features during initialization.

4) Disturbance Rejection: The system's capabilities in a control loop of an MAV is tested by commanding the MAV to hover at a set-point and repeatedly disturbing it. Fig. 5 shows still frames from this experiment and we observe that the MAV returns to its set-point after each disturbance.

5) Validation of Anchor-Centric Approach: The anchor-centric parameterization is relevant particularly for longer trajectories. We analyze its effect based on an outdoor trajectory. Fig. 7 shows the estimated trajectory with the anchor-centric parameterization in red and with a world centric parametrization in green.

We observe drift in yaw, as seen by the misalignment of the trajectories as well as drift in position of close to three times as much as with the anchor-centric parameterization.

V. Conclusion

We presented a Visual-Inertial Odometry system that relies solely on off-the-shelf and light-weight components. All computations are performed on-board an embedded ARM processor and no external sensors or beacons are required.

We have demonstrated that the system is able to accurately track long trajectories and is robust with respect to fast

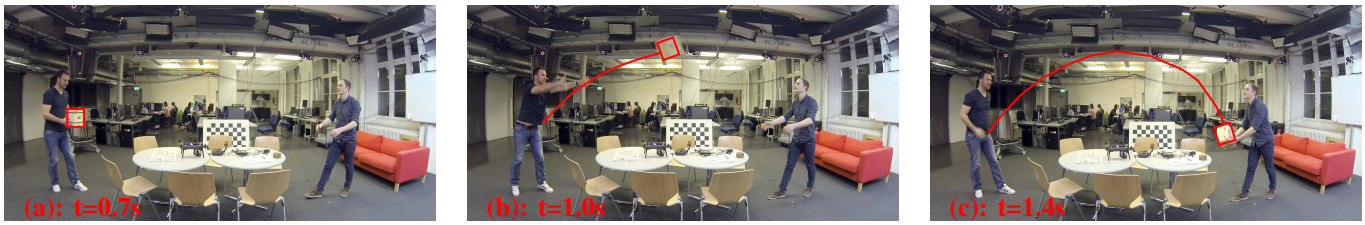


Fig. 4: Still images of the throwing experiment. The labels denote the same time instances as in Fig. 3

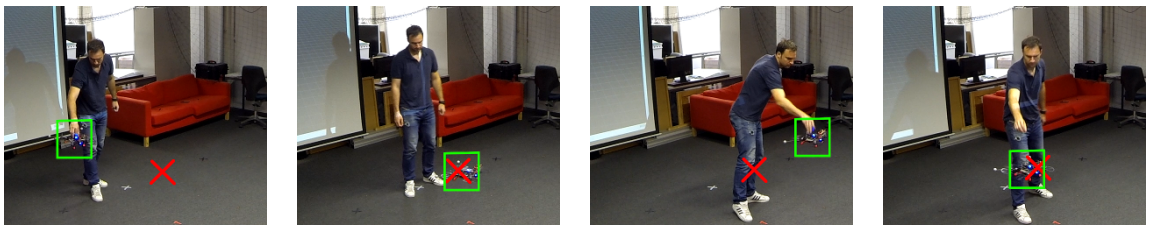


Fig. 5: Still images of the disturbance rejection experiment. The MAV (green) is disturbed from its setpoint (red) and returns to it.

motions. We have further shown that the system enables the stabilization of a Micro Aerial Vehicle's position in flight.

The affordable hardware in combination with the algorithm available as open-source software presents a pose estimation system that is easily incorporated in a wide variety of applications.

References

[1] E. Altuğ, J. P. Ostrowski, and C. J. Taylor, "Quadrotor control using dual camera visual feedback," in *ICRA*, 2003.

[2] ——, "Control of a quadrotor helicopter using dual camera visual feedback," *The International Journal of Robotics Research*, 2005.

[3] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor mav," in *IROS*, 2012.

[4] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, 2007.

[5] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based mav navigation in unknown and unstructured environments," in *ICRA, 2010 IEEE international conference on*, 2010.

[6] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robotics and Autonomous Systems (RAS)*, 2014.

[7] ——, "Camera-based navigation of a low-cost quadrocopter," in *Proc. of the International Conference on Intelligent Robot Systems*, 2012.

[8] S. Leutenegger, P. T. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial slam using nonlinear optimization." in *Robotics: Science and Systems*, 2013.

[9] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for rgb-d cameras," in *ICRA, International Conference on*. IEEE, 2013.

[10] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014.

[11] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003.

[12] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2007.

[13] E. J. Lefferts, F. L. Markley, and M. D. Shuster, "Kalman filtering for spacecraft attitude estimation," *Journal of Guidance, Control, and Dynamics*, 1982.

[14] S. Roumeliotis, G. Sukhatme, G. A. Bekey, *et al.*, "Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization," in *Robotics and Automation*. IEEE, 1999.

[15] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, A. Ansar, and L. Matthies, "Vision-aided inertial navigation for spacecraft entry, descent, and landing," *Robotics, IEEE Transactions on*, 2009.

[16] K. Tsotsos, A. Chiuso, and S. Soatto, "Robust inference for visual-inertial sensor fusion," *arXiv preprint arXiv:1412.4862*, 2014.

[17] E. S. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," *The International Journal of Robotics Research*, 2011.

[18] M. Li and A. I. Mourikis, "Optimization-based estimator design for vision-aided inertial navigation," in *Robotics: Science and Systems*, 2013.

[19] P. Tanskanen, T. Naegeli, M. Pollefeys, and O. Hilliges, "Semi-direct ekf-based monocular visual-inertial odometry," *IROS*, 2015.

[20] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 298–304.

[21] W. Breckenridge, "Quaternions proposed standard conventions," *Jet Propulsion Laboratory, Pasadena, CA, Interoffice Memorandum*, 1979.

[22] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, A. Ansar, and L. Matthies, "Vision-aided inertial navigation for spacecraft entry, descent, and landing," *IEEE Transactions on Robotics*, 2009.

[23] Pietzsch, "Efficient Feature Parameterisation for Visual SLAM Using Inverse Depth Bundles," *Bmvc*, 2008.

[24] R. Martinez-Cantin and J. a. Castellanos, "Bounding uncertainty in EKF-SLAM: The robocentric local approach," *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, 2006.

[25] J. A. Castellanos, J. Neira, and J. D. Tardos, "Limits to the consistency of EKF-based SLAM," 2004.

[26] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel, "1-point RANSAC for EKF-based structure from motion," *IROS*, 2009.

[27] N. Trawny and S. I. Roumeliotis, "Indirect Kalman Filter for 3D Attitude Estimation," *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, 2005.

[28] B. P. Gibbs, *Advanced Kalman filtering, least-squares and modeling*. John Wiley & Sons, 2011.

[29] J. Montiel, J. Civera, and A. Davison, "Unified inverse depth parametrization for monocular SLAM," *Analysis*, 2006.

[30] M. a. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, 1981.

[31] W. F. Denham and S. Pines, "Sequential estimation when measurement function nonlinearity is comparable to measurement error." *AIAA journal*, 1966.

[32] D. Honegger, P. Greisen, L. Meier, P. Tanskanen, and M. Pollefeys, "Real-time velocity estimation based on optical flow and disparity matching," in *IROS*, Oct 2012.