Visual-Inertial On-Board Throw-and-Go Initialization for Micro Air Vehicles

Martin Scheiber[1], Jeff Delaune[2], Roland Brockers[2], and Stephan Weiss[1]

***Abstract*— We propose an approach to the throw-and-go (TnG) problem for micro air vehicles (MAVs) using visual and inertial sensors. The key challenge is the fast on-board initialization of the visual odometry (VO) system, which usually requires user input to recover the visual scale. Our approach is based on the identification of the gravity vector from the acceleration data computed with images of the ground during in free fall. This enables scaling of the poses reconstructed with visual information. The proposed framework use inertial data to control the MAV attitude so the ground is visible after the throw. Using image to image homography a metric scale is estimated with which the MAV's height is propagated. Unlike existing literature, this approach requires no additional sensor nor user input or pre-throw assumptions and can recover from any initial attitude. We show results on both simulation and real data.**

I. INTRODUCTION

Micro air vehicles (MAVs) have significantly impacted robotics research in recent years. Their agility makes them useful in many situations, especially when they are autonomous. Modern use cases include field monitoring for agriculture, first responder's aid, aerial photography or planetary exploration. We reckon that autonomous MAVs will keep playing a bigger role in modern society and that they need to be deployed easily.

MAV's autonomy requires robust and accurate closed-loop control. Estimating the metric pose and velocity is a critical part of it. Research has been focusing on computer vision to enable navigation indoor or in tight spaces where MAVs are often required to operate [1]. Since parallel tracking and mapping (PTAM) [2], real-time visual odometry or simultaneous localization and mapping (SLAM) has become a fundamental part of robotics. Modern VO algorithms can run at frame rates greater than 60 fps on embedded hardware or re-localize themselves with some success [3], [4]. Yet, monocular VO approaches cannot provide the metric pose information required for MAV control because of the visual scale ambiguity.

Inertial Measurement Units (IMUs) are the most popular sensor in the MAV literature to recover metric information. [5] pioneered visual-inertial odometry (VIO) for MAVs by adapting the well-known PTAM VO algorithm [2], and using it to update an inertial prior through an Extended Kalman Filter (EKF). This approach can use any VO pose output in a loosely-coupled way, at the cost of a scalar visual scale state and visual frame orientation state in the filter. This approach is lightweight and modular, but cannot naively handle VO tracking loss when imaging conditions are difficult. Tightly-coupled VIO uses image feature tracks as measurements [6], [7]. It can work in degraded conditions with only a few features, is more accurate but requires numerous additional feature or pose states which increase the computational cost. Recently, "ultra-tightly-coupled" VIO has appeared and uses image pixel intensity directly as updates [8], [9].

Whether they are based on non-linear filtering or optimization, any of these VIO approaches require the state vector to be initialized in some way. As opposed to VO-only, the gravity vector and visual scale factor need to be approximated as well. The latter may appear as a scalar scale state in loosely-coupled approaches, or as the metric velocity vector in some tightly-coupled approaches. This initialization usually requires user input guesses [5], or solving a linear system over a short visual-inertial sequence where static objects are in the field of view [10].

Throw-and-go (TnG) defines the ability for a user to throw the copter in mid-air to start an autonomous flight. Ideally, this should succeed from any throw configuration (e.g. orientation, camera obstruction) and without requiring additional sensors. In this paper, we present a TnG initialization procedure that can be adapted to any of the aforementioned VIO approaches. It requires no sensors other than a camera and an IMU, and the MAV can be thrown from any orientation, even if the camera is initially obstructed.

The following paper is split into several parts. Section II reviews previous approaches to the TnG problem. Section III gives an overview of our solution goes into further details about the implementation. Sections IV and V go over simulation and real-world open loop tests, respectively.

The research leading to these results has received funding from the ARL within the BAA W911NF-12-R-0011 under grant agreement W911NF-16-2-0112, the Austrian Ministry for Transport, Innovation and Technology (BMVIT) under grant agreement n. 855468 (Forest-IMATE), and the Universität Klagenfurt within the Karl Popper Kolleg on Networked Autonomous Aerial Vehicles.

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

[1] These authors are with the Control of Networked Systems Group, Univeristät Klagenfurt, Austria.
mascheiber@edu.aau.at, stephan.weiss@ieee.org

[2] These authors are with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena.
{jeff.h.delaune,roland.brockers}@jpl.nasa.gov

Pre-print version, accepted Jun./2019 at IROS19, Macau,
DOI: 10.1109/IROS40897.2019.8967575 ©IEEE.

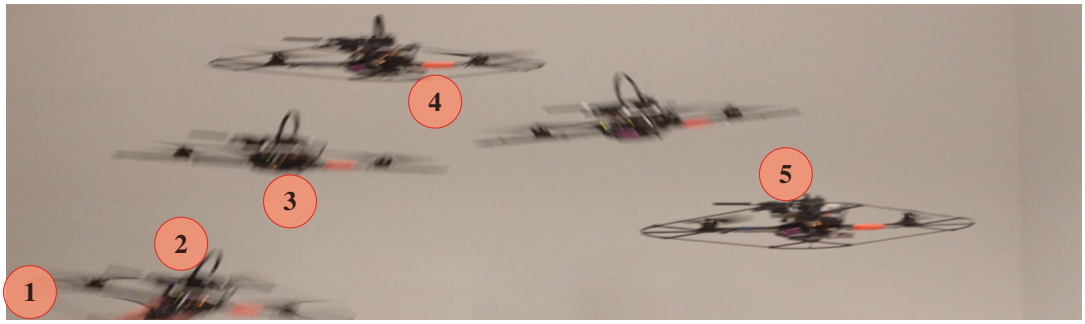

Fig. 1. Stages of the throw within our framework. The attitude is estimated (1) before the throw is detected (2), then at first the attitude is stabilized (3) and then the vision based pose calculation started (4). At last, the copter computes the metric scale and initializes the VIO framework and controller (5).

II. RELATED WORK

To the best of our knowledge, two approaches have been proposed for MAV's throw-and-go.

[11] tested TnG with an EKF-based framework using inertial propagation and optical flow measurements, previously proposed in [12]. This only required two consecutive image frames to provide a measurement. It overcame the manual initialization process required by [2], in addition to being more robust to tracking failure. However, the orientation, velocity and a visual scale factor still need to be initialized manually before the throw, and an assumed ground plane has to be in the field of view the whole time.

The approach proposed in [13] can start from any attitude and with an obstructed view since they use inertial-only attitude observer to estimate the gravity vector during the throw. As soon as the MAV is released, attitude control levels out the MAV and points a downward-facing camera at the assumed ground plane. A laser altimeter is used to eliminate the descent rate and provide the scale factor. The residual lateral velocity provides the translation to initialize the semi-direct monocular visual odometry (SVO) algorithm [3]. This approach does not need user inputs and can recover from any throw attitude or camera obstruction, but at the cost of an extra sensor.

Our work also solves TnG without the need for any user inputs, with any initial attitude or camera obstruction during the throw, but without any additional sensors other than a camera and an IMU.

III. SYSTEM OVERVIEW AND APPROACH

Within the scope of this paper we split a conventional MAV throw into several stages, as shown in Fig. (1). Similarly our framework can be split up into several components, representing each throw stage.

We introduce in Fig. 2 the coordinate frames used in our computation. The world frame $\mathcal{W}$ is affixed to the ground, with the z axis pointing upwards. The camera frame $\mathcal{C}$ and the IMU frame $\mathcal{I}$ are rigidly attached to the MAV. We define a homography frame $\mathcal{H}$ static in the world frame, resulting from a rotation of the first camera frame $\mathcal{C}_0$ about its x and y axis so as to point z upwards. $\mathcal{V}$ is the reference VO frame used in loosely-coupled approaches. $\mathbf{q}_{AB}$ is the Hamilton quaternion modeling the rotation from $\mathcal{A}$ to $\mathcal{B}$. ${}_A\mathbf{p}_{AB}$ is the translation from $\mathcal{A}$ to $\mathcal{B}$, expressed in $\mathcal{A}$ coordinates.

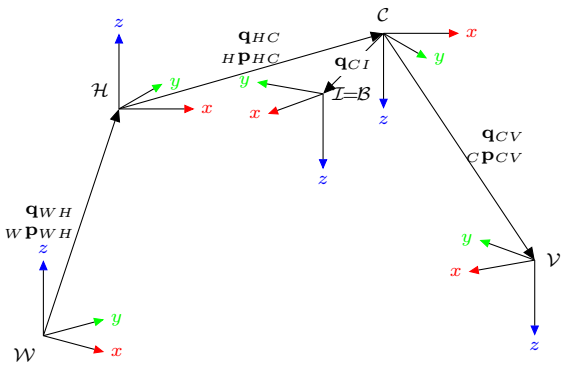

Fig. 2. Within this paper we use several different coordinate frames. Within the MAV the two frames for IMU $\mathcal{I}$ and Camera $\mathcal{C}$ are used, although the Body frame aligns per our definition with the IMU frame (hence $\mathcal{I} = \mathcal{B}$). The homography frame $\mathcal{H}$ is used for simplification while the MAV is in the air and is defined as the upward rotated first camera frame, hence $\mathcal{H} = \mathbf{R}(\mathbf{q}_{xy})\mathcal{C}_0$. The vision frame $\mathcal{V}$ is used for the VIO framework once initialized.

Shortly before the throw is detected the attitude observer, providing us an attitude estimate $\hat{\mathbf{q}}_{WB}$ from the IMU measurements ${}_B\breve{\mathbf{a}}$ and ${}_B\breve{\boldsymbol{\omega}}$, is started. This is necessary, as a good attitude estimate is required for mid-air attitude stabilization. As soon as the throw is detected the attitude stabilization and recovery starts. Once stabilized, the feature tracker is started and with it an unscaled MAV pose calculated. At last, the scaling algorithm derives the metric scale needed for the VO initialization using the throw trajectory attributes. A state diagram of our proposed system is represented in Fig. 3.

The next subsections cover the mathematical approach used in identifying the gravity vector in visual and inertial data to first estimate the attitude during the throw, and later the visual scale during free fall.

A. Attitude observer

The attitude must be controlled as early as possible after the throw to ensure the camera is pointing at the ground. We implemented an attitude observer similar to the one proposed

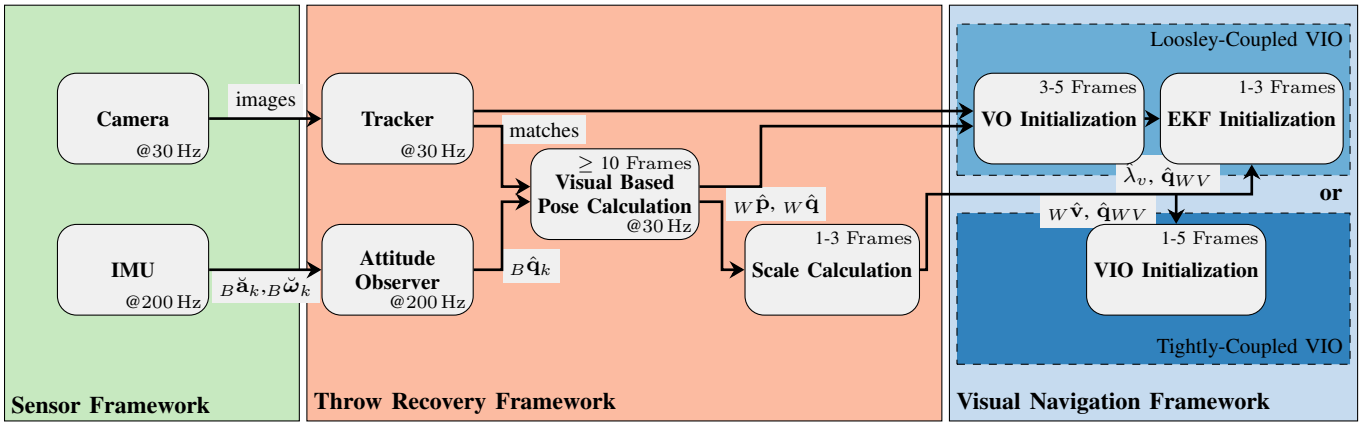

Fig. 3. The state diagram of our proposed framework. It consists of the two sensors camera and IMU with image, angular velocity and linear acceleration measurements. Once the throw is detected, our framework takes these measurements and calculates the current pose using a homography-based approach. After detecting a z-velocity change, these pose measurements are scaled metrically and the visual-inertial state-estimation is initialized.

in [13]. In their approach, they split the estimation in a prediction ${}_B\mathbf{q}_{\text{pred}}$ and update ${}_B\mathbf{q}_{\text{corr}}$ part.

$$ {}_B\mathbf{q}_{\text{pred},k} = \left(\mathbf{I}_4 \cdot \cos\left(\frac{\|{}_B\breve{\boldsymbol{\omega}}_k\| \Delta t}{2} \right) + \frac{2}{\|{}_B\breve{\boldsymbol{\omega}}_k\|} \cdot \Lambda({}_B\breve{\boldsymbol{\omega}}_k) \cdot \sin\left(\frac{\|{}_B\breve{\boldsymbol{\omega}}_k\| \Delta t}{2} \right) \right) \cdot \hat{\mathbf{q}}_{k-1} \tag{1} $$

The prediction is calculated with the zero-th order quaternion integration shown in Eq. (1). It assumes the body rates are constant over Δt. $\mathbf{I}_4 \in \mathbb{R}^{4\times4}$ denotes the identity matrix and ${}_B\breve{\boldsymbol{\omega}}_k$ is the current gyroscope measurement. $\breve{\boldsymbol{\omega}}$ of vector $\boldsymbol{\omega}$ denotes its raw sensor measurement, $\hat{\boldsymbol{\omega}}$ its estimate. $\Lambda({}_B\breve{\boldsymbol{\omega}}_k)$ is the skew-symmetric rotation matrix w.r.t. the angular velocity ${}_B\breve{\boldsymbol{\omega}}_k$.

In order to align the predicted attitude estimate with the corresponding body z direction a correction step is performed. In general the estimated body z direction ${}_B\hat{\mathbf{e}}_{z,pred}$ is rotated towards the measured acceleration ${}_B\breve{\mathbf{a}}$. The predicted estimate of body z direction is calculated by ${}_B\hat{\mathbf{e}}_{z,pred} = \mathbf{R}({}_B\mathbf{q}_{\text{pred}}) \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\mathsf{T} \mathbf{R}({}_B\mathbf{q}_{\text{pred}})^{-1}$. The quaternion multiplication operator is denoted by $\otimes$.

$$ \alpha = \arccos\left({}_B\hat{\mathbf{e}}_{z,\text{pred}} \cdot \frac{{}_B\breve{\mathbf{a}}}{\|{}_B\breve{\mathbf{a}}\|} \right) \tag{2} $$

$$ {}_B\mathbf{h} = \frac{{}_B\breve{\mathbf{a}} \times {}_B\hat{\mathbf{e}}_{z,\text{pred}}}{\|{}_B\breve{\mathbf{a}} \times {}_B\hat{\mathbf{e}}_{z,\text{pred}}\|} \tag{3} $$

$$ \mathbf{q}_{\text{corr},k} = \begin{bmatrix} \cos(c_{\text{corr}} \cdot \frac{\alpha}{2}) \\ {}_B\mathbf{h} \sin(c_{\text{corr}} \cdot \frac{\alpha}{2}) \end{bmatrix} \tag{4} $$

$$ {}_B\hat{\mathbf{q}}_k = {}_B\mathbf{q}_{\text{pred},k} \otimes {}_B\mathbf{q}_{\text{corr},k} \tag{5} $$

Note that Eq. (1) can only be calculated if $\|{}_B\breve{\boldsymbol{\omega}}_k\| \neq 0$. Otherwise the attitude prediction is kept constant. Further the update step is only performed if the currently measured body rates are small, i.e. $\|{}_B\breve{\boldsymbol{\omega}}_k\| < 0.5\,\text{rad}\,\text{s}^{-1}$ and $\text{abs}\left(\|{}_B\breve{\mathbf{a}}_k\| - \mathbf{g}\right) < 1.5\,\text{m}\,\text{s}^{-2}$.

This attitude observer depends on the update step to be performed in order to converge. Thus it relies on the assumption that the accelerometer measures the gravity direction on average. This is true when being held stationary, false however when maneuvering or in free fall. Hence starting the attitude observer mid-air does not yield the update step being performed. The attitude observer therefore has to be running throughout the throw. However tests showed, that starting it just before throwing is enough for its estimates to be within reasonable margins, i.e. less than $5°$ off the ground truth.

B. Pose Calculation

Once having an attitude estimate we detect the throw with the current raw acceleration measurement ${}_B\breve{\mathbf{a}}$. As soon as the measurement falls below a certain threshold a_{thresh} a throw is detected. Additionally the idle rotor thrust c_{idle} has to be taken into account.

$$ \|{}_B\breve{\mathbf{a}}_k\| < c_{\text{idle}} + a_{\text{thresh}} \tag{6} $$

When having a successful throw detection and a stabilized attitude we start our tracker framework. For simplicity, we describe here how we calculate the camera pose using homographies computed from frame-to-frame image feature matches. The homography ground plane assumption is not intrinsic to our framework and could be lifted by leveraging more advanced pose estimation techniques using visual and pre-integrated inertial constraints like in [10]. However, that causes a complexity cost which was not deemed necessary by the authors to confirm that identifying the gravity vector acceleration was a viable solution for TnG.

The homography matrix consists of the rotation, scaled translation and normal of the plane in sight, Eq. (7) as presented in [14]. $\mathbf{R}_{C_{k-1}C_k}$ and ${}_{C_{k-1}}\mathbf{t}_{C_{k-1}C_k}$ describe the camera rotation and translation from the previous image frame at time step $k-1$ to the current one at k, respectively. ${}_Cd_{k-1}$ is the scene depth from the previous image and ${}_C\mathbf{n}_k$ is the scene plane normal. For simplicity $\mathbf{R}_{k-1,k}$, $\mathbf{t}_{k-1,k}$,

d_k and $\mathbf{n}_k$ will be used for the above mentioned variables, respectively. Important to note here is that the translation is normalized with respect to the current image depth.

$$\mathbf{H}_k = \mathbf{R}_{k-1,k} + \frac{\mathbf{t}_{k-1,k}}{d_k}\,\mathbf{n}_k^\mathsf{T} \tag{7}$$

When decomposing the homography matrix however, we can only derive the normalized, estimated translation $\mathbf{u}_{k-1,k} = \frac{\mathbf{t}_{k-1,k}}{d_k}$. So the derived translation has an arbitrary scale, which needs to be determined later on. Moreover, using the method proposed in [14] at most two pairs of mathematical possible solutions can be derived for each homography matrix decomposition Eq (8).

$$\mathbf{H}_k \Rightarrow \begin{cases} \left\{\mathbf{R}^a_{k-1,k},\, \mathbf{u}^a_{k-1,k},\, \mathbf{n}^a_k\right\} \\ \left\{\mathbf{R}^a_{k-1,k},\, -\mathbf{u}^a_{k-1,k},\, -\mathbf{n}^a_k\right\} \\ \left\{\mathbf{R}^b_{k-1,k},\, \mathbf{u}^b_{k-1,k},\, \mathbf{n}^b_k\right\} \\ \left\{\mathbf{R}^b_{k-1,k},\, -\mathbf{u}^b_{k-1,k},\, -\mathbf{n}^b_k\right\} \end{cases} \tag{8}$$

To eliminate further solutions the physical constraint that each feature has to lie on the visible plane can be applied. Without loss of generality, the solutions with negative z-normals of Eq (8) have hence been removed. This leads to two different, yet possible solutions.

At this point, we assume to be having an attitude estimate from our observer. Further, the MAV's attitude should be stabilized at this point resulting in a "stable" fall. Thus the relative rotation between the last two camera frames $\mathbf{q}_{k-1,k}$, derived from the homography decomposition, should be the same as the estimated rotation difference in the world frame, $\hat{\mathbf{q}}^{-1}_{WC_{k-1}} \otimes \hat{\mathbf{q}}_{WC_k}$. Since both quaternions are normalized we can use their scalar product to determine which decomposition is closer to 1, meaning equality, as seen in Eq (9).

$$\min_i \left\{ \text{abs}\left(\left\langle \hat{\mathbf{q}}^{-1}_{WC_{k-1}} \otimes \hat{\mathbf{q}}_{WC_k}, \mathbf{q}^i_{k-1,k} \right\rangle \right) - 1 \right\} \tag{9}$$

C. Scaling

We already explained the scaling issue when decomposing the homography matrix. However, for the filter framework initialization, we need the metric scale of our system. As already discussed each decomposed translation $\mathbf{u}_{k-1,k}$ has an arbitrary scale or current scene depth d_k. Also, each homography decomposition might have its own scale. Since we do not track the same features over several frames, even these scale values might not depend directly on each other. We, therefore, propose a normalization of the scale to our initial depth guess with which we will be able to propagate the position. By defining our system this way not only do the depths become dependent on each other but further, the camera's position can be propagated. Furthermore, both become correct up to a single metric scale, which can be stated as the offset of our initial guess.

Assuming the previous scene depth is known, the current depth can be computed by projecting the translation onto the normal of the plane, Eq. (10).

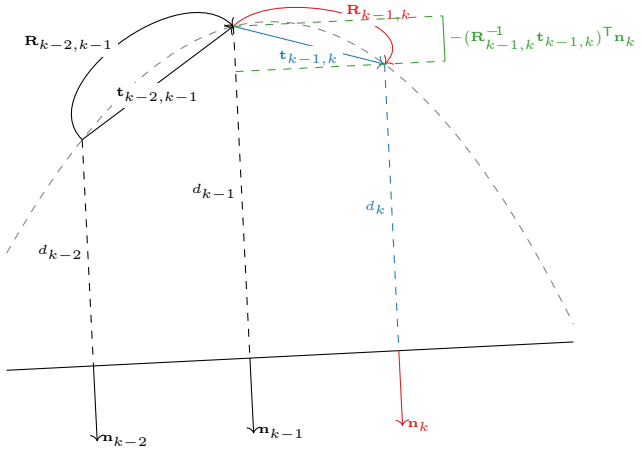

Fig. 4. The homography decomposition and pose propagation for a sample throw trajectory shown in gray. $\mathbf{R}_{k-1,k}$ and $\mathbf{t}_{k-1,k}$, short for $\mathbf{R}_{C_{k-1}C_k}$ and ${}_{C_{k-1}}\mathbf{t}_{C_{k-1}C_k}$, refer to the rotation and translation from the previous camera frame $\mathcal{C}_{k-1}$ to the current camera frame $\mathcal{C}_k$, respectively, $\mathbf{n}_k$ describes the plane normal as seen in the current camera frame ${}_{C_k}\mathbf{n}_k$ and d_k is the current scene depth. Values directly derived from the homography decomposition are in red, indirectly derived ones in blue and computed ones in green.

$$d_k = d_{k-1} + \mathbf{t}^\mathsf{T}_{k,k-1}\,\mathbf{n}_k \tag{10}$$

$$\mathbf{t}_{k-1,k} = -\mathbf{R}_{k-1,k} \cdot \mathbf{t}_{k,k-1} \tag{11}$$

Combining Eq. (10) and (11) the current scene depth can then be derived, Eq. (12).

$$d_k = \frac{d_{k-1}}{1 + \left(\mathbf{R}^{-1}_{k-1,k} \cdot \mathbf{u}_{k-1,k}\right)^\mathsf{T}\mathbf{n}_k} \tag{12}$$

We can then use the current scene depth to derive the unscaled translation and propagate the MAV position within the homography frame. By designing our system this way, all homography decomposition become dependent of each other as Eq. (13) depicts.

$${}_H\mathbf{p}_k = {}_H\mathbf{p}_{k-1} + \mathbf{R}_{HC_{k-1}}\,\mathbf{u}_{k-1,k}\,d_k \tag{13}$$

In order to reduce the image noise of the homography decomposition, we apply a linear-least-square (LLS) optimization. A parabola model as shown in Eq. (14) is used since any throw or drop without external influences can be described by it.

$${}_H\mathbf{p}(t) = {}_H\mathbf{p}(t_0) + t \cdot {}_H\mathbf{v}(t_0) + t^2 \cdot \frac{{}_H\mathbf{a}(t_0)}{2} \tag{14}$$

For each instance of the homography decomposition Eq. (14) can be applied. Thereby we derive the system by Eq. (15), which can be written in the form of $\mathbf{Y} = \hat{\boldsymbol{\beta}}\mathbf{X}$.

Property Name		Value	
Image noise	n_{px}	1	px
IMU acceleration noise	n_a	0.0083	$\mathrm{m/s^2}$
IMU gyroscope noise	n_ω	0.0013	rad/s
IMU acceleration bias	b_a	0.00013	$\mathrm{rad/s^2}$
IMU gyroscope bias	b_ω	0.00083	$\mathrm{m/s^3}$
Image Resolution	res_{IMG}	640×480	px
RNASAC px threshold	th_{px}	1	px
Max. Features	$ft_{\max}$	400	
Min. Matches	$m_{\min}$	200	
Max. Iterations	$it_{\max}$	2000	

TABLE I

Values used for simulating our proposed framework. Please note that the noise and bias random walk refer to the standard deviation of a zero-mean based Gaussian distribution.

$$\begin{bmatrix} {}_H\mathbf{p}(t_1)^\mathsf{T} \\ \vdots \\ {}_H\mathbf{p}(t_n)^\mathsf{T} \end{bmatrix} = \mathbf{I}_{n\times 1}\ {}_H\mathbf{p}(t_0)^\mathsf{T} + \begin{bmatrix} t_1 \\ \vdots \\ t_n \end{bmatrix} {}_h\mathbf{v}(t_0)^\mathsf{T} + \begin{bmatrix} t_1^2 \\ \vdots \\ t_n^2 \end{bmatrix} \frac{{}_H\mathbf{a}(t_0)^\mathsf{T}}{2} \tag{15}$$

Hence the estimated factors $\hat{\boldsymbol{\beta}}$ of our system can be computed with the LLS optimization method of Eq. (16). The aim of this operation is to estimate the initial throw parameters $\hat{\boldsymbol{\beta}}$.

$$\hat{\boldsymbol{\beta}} = \begin{bmatrix} {}_H\hat{\mathbf{p}}(t_0)^\mathsf{T} \\ {}_H\hat{\mathbf{v}}(t_0)^\mathsf{T} \\ \frac{1}{2}\ {}_H\hat{\mathbf{a}}(t_0)^\mathsf{T} \end{bmatrix} = \left(\mathbf{X}^\mathsf{T}\mathbf{X}\right)^{-1} \cdot \left(\mathbf{X}^T\mathbf{Y}\right) \tag{16}$$

In our current system definition, all camera positions are dependent on the first choice of the scene depth. Thus all position propagations are off metric by the same scale λ. By using simple differentiation rules, it can further be shown that even the camera's velocity and acceleration are off their metric value by this scale.

$$ {}_W\mathbf{p}(t) = \lambda \left(\mathbf{e}_z\, d_0 + R_{WH}\ {}_H\mathbf{p}(t)\right) \tag{17}$$

$$ {}_W\mathbf{v}(t) = \lambda\, R_{WH}\ {}_H\dot{\mathbf{p}}(t) = \lambda\, R_{WH}\ {}_H\mathbf{v}(t) \tag{18}$$

$$ {}_W\mathbf{a}(t) = \lambda\, R_{WH}\ {}_H\ddot{\mathbf{p}}(t) = \lambda\, R_{WH}\ {}_H\mathbf{a}(t) \tag{19}$$

Moreover, while in free fall the acceleration of the MAV does not change. It is known to be gravity subtracted by the idle motor thrust c_{idle}. Hence the scale of our system λ can be derived by comparing the estimated acceleration ${}_H\mathbf{a}$ to gravity ${}_W\mathbf{g}$.

$$ {}_W\mathbf{a} = \mathbf{R}_{WH}\left(\lambda\ {}_H\hat{\mathbf{a}}(t_0)\right) = {}_W\mathbf{g} \tag{20}$$

IV. EXPERIMENTAL SIMULATION

To verify our proposed initialization algorithm we initially simulated the framework in MATLAB. Within our simulation, we dynamically generated features to compute scaled position and velocity. We further added realistic

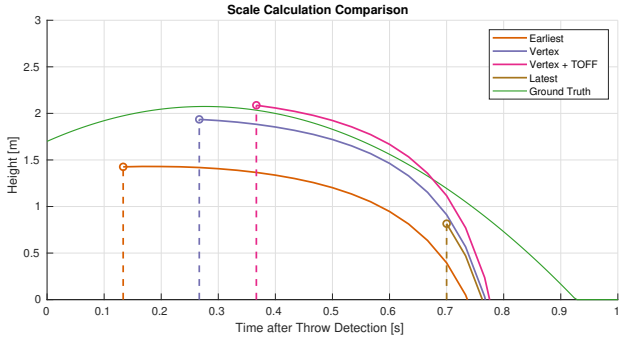

Fig. 5. The scale and scene depth was calculated at different times in simulation to derive the optimal time. This figure compares a sample throw trajectory (green) to different possible calculation points: after 3 frames (orange), at the parabola apex (violet), 0.1 s (or t_{OFF}) after the parabola apex (pink), and shortly before hitting the ground (green). Please note that we did not simulate any MAV control. As a result the height propagations diverge when getting close to the ground as good visible features get rarer.

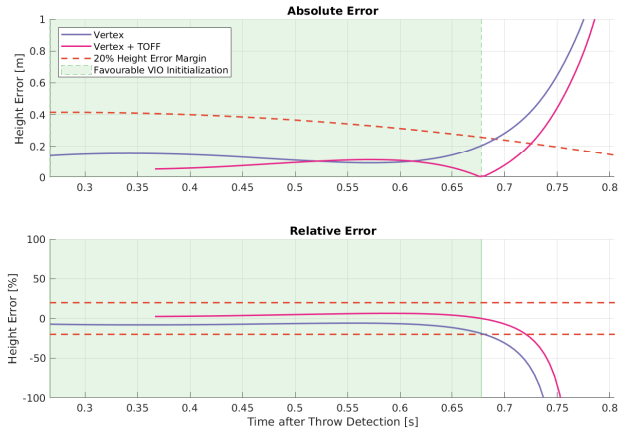

Fig. 6. Comparison of simulated scaled height propagation to our set margins of 20 % of the ground truth height (orange dotted). At approximately 0.68 s the propagations start to diverge as the MAV is getting close to the ground. This gives us an upper time limit for the VIO initialization and MAV control takeover, with the lower boundary being the time of scale calculation (green area).

noise and bias values to our measurements to complete the simulation. The maximum standard deviation used for each value is shown in Table I. The goal of this experimental simulation was to confirm that scaling the scene depth and thus the homography decompositions with a gravity-based approach is possible.

The only force acting on the copter once thrown is gravity. Hence we used a parabola model for the throw trajectory as described in Eq. (14). As initial conditions a starting height of 0.8 m and an initial velocity of $v_0 = \left[1, 1, 5\right]\,\mathrm{m\,s^{-1}}$ were used. In addition minor attitude changes, with up to 20° before being stabilized and up to 5° after being stabilized were added to reflect real-world behavior.

To determine the robustness of our approach regarding the time of scale calculation we tested our proposed algorithm on several different simulations. An example of such a run can

be seen in Fig. 5. We generally compared the following scale computation times: after the minimum three required frames for acceleration calculation, at the apex of the trajectory, and shortly before reaching the ground. Due to the fact that in real-world scenarios it is very unlikely to detect the apex of a parabola perfectly, as a result of the discrete frame rate and computation times, we additionally opted to evaluate the scale computation at $0.1\,\mathrm{s}$ after reaching the apex.

As expected, simulation showed us that both calculations done around the apex of the trajectory yield feasible results. As shown in Fig. 6 both of them lay within our set $20\,\%$ off ground truth margins. They further remain within said margins and remain within them for most of the remainder of the throw. Please note that the simulated MAV is not controlled at this point and the vision based approach fails at approximately $0.68\,\mathrm{s}$ when getting close to the ground, as unique features get rarer. We further confirmed, that using the calculated scale a propagation of the MAV height was possible using the visual pose calculations. These results show that for a near ideal throw we would have at least $0.5\,\mathrm{s}$ for the VIO initialization and control takeover.

V. REAL-WORLD RESULTS

This section covers our experimental, real-world setup and results of our proposed framework.

A. Implementation

We already explained that our system could be used to initialize loosely as well as tightly coupled VIO frameworks. For simplicity we chose to use a slightly modified version of the open-source VO SVO [3] in combination with the sensor fusion framework presented in [15]. Most of our changes to the publicly available source code involved minor speedup optimization.

To speedup the initialization, we removed the feature triangulation within the SVO initialization process. Instead we directly provided the transform from our homography computation to SVO. Moreover, as soon as the visual front-end was initialized, the EKF was started. The corresponding state-vector of this framework is described in Eq. (21). It includes the MAV position, velocity and attitude expressed in the world frame, the biases for the accelerometer and gyroscope, the visual scale and the extrinsic camera to IMU calibration.

$$\mathbf{X} = \left[\mathbf{p}_{WB},\, \mathbf{v}_{WB},\, \mathbf{q}_{WB},\, \mathbf{b}_{\omega},\, \mathbf{b}_{a},\, \lambda_{v},\, \mathbf{p}_{BC},\, \mathbf{q}_{BC}\right] \quad (21)$$

Here the initial position and attitude for $\mathbf{p}_{WB}$ and $\mathbf{q}_{WB}$ are chosen according to our current MAV world position, estimated with our framework. Similarly the current velocity $\mathbf{v}_{WB}$ is derived. The biases $\mathbf{b}_{\omega}$ and $\mathbf{b}_{a}$ are initialized with similar values to Tab I, whereas the camera to IMU translation and rotation, $\mathbf{p}_{BC}$ and $\mathbf{q}_{BC}$, are known from calibration. Important to note is that the visual scale λ_v within the EKF framework is not equal to our derived scale λ. Instead the visual scale λ_v depends on the distance to the feature plane of the visual framework ${}_{V}df$. Using Eq. (17) the conversion from one scale to the other can be derived. ${}_{H}p_z$ is the MAV height described in the homography $\mathcal{H}$ frame.

$$\lambda_v = \frac{{}_{V}df}{{}_{W}h} = \frac{{}_{V}df}{\lambda \cdot \mathbf{R}_{WH}\,{}_{H}p_z} \quad (22)$$

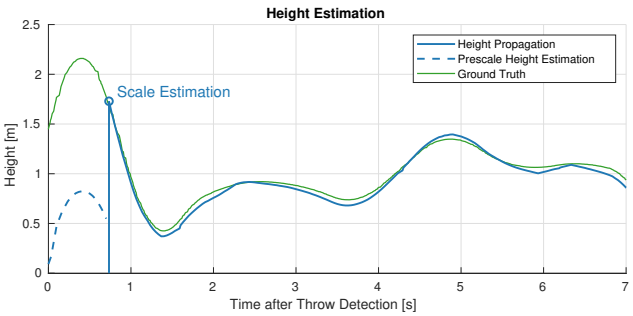

Fig. 7. The estimated height computed and propagated with our algorithm, compared to the ground-truth provided by Optitrack. After the throw is detected the unscaled height is derived from the homography decompositions (blue dashed). The metric scale is then estimated shortly after the throw apex and height propagation continues with the newly derived metric scale. At approximately $1.15\,\mathrm{s}$ the external, manual MAV is started. Although this leads to altitude changes, the propagated height is still able to follow the ground truth for several more seconds, before the MAV lands at approximately $7.5\,\mathrm{s}$, at which point the propagation fails.

B. Setup

We implemented our system on a octacore @$2\,\mathrm{GHz}$ ODROID-XU4 mounted on an Asctec Hummingbird. As visual sensor we used a mvBlueFox-MLCw with a resolution of 752×480 px. This setup allows us to have a lightweight, very agile MAV, fast enough to respond and stabilize when thrown. Moreover we designed our framework to work with an IMU update rate of $200\,\mathrm{Hz}$ and a camera image frame rate of $30\,\mathrm{FPS}$.

Currently we select the throw threshold a_{thresh}, Eq. (6), with $1\,\mathrm{m/s^2}$. Tests further showed, that the idle thrust is on average below $0.5\,\mathrm{m/s^2}$, hence c_{idle} is set to that. The implemented feature tracker and matcher uses a FAST Lucas-Kanade approach with the same values as in our simulation (see Tab. I). Similarly we used a RANSAC based approach for the homography decomposition with the simulation values for maximum iterations and pixel re-projection threshold. The planar assumption made earlier using the homography for simplicity often holds indoors. Tests showed that small, static objects with an elevation of less than $10\,\%$ of the scene depth still allow the planar assumption in our framework. For outdoor usage, this signals that our approach may be valid at higher altitude with respect to the objects on the ground, e.g. MAV drops from balloon or tall buildings.

C. Results

Using the above provided implementation and setup we threw the copter. A sample of such a throw is depicted in Fig. 7.

Although using a simple homography based framework, which ran without complications at $30\,\mathrm{Hz}$ on our Laptops,

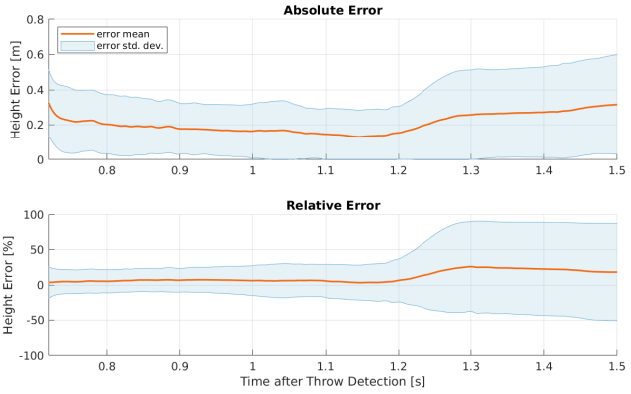

Fig. 8. The mean height error and its standard deviation for 10 throws. The scale is calculated just after 0.7 s and the external, manual control starts at approximately 1.15 s. As can be seen the propagated height error on average remains quite stable within our set limits of $\pm 20\,\%$ until the manual control starts.

the Odroid was on its computational limit. Trying to run both SVO and our framework simultaneously yielded in longer computation times. This in turn meant fewer evaluated images per second which in turn decreased the amount of data points for our LLS based scale estimation leading to an increased computed scale and homography decomposition error. As a result we opted to use an external, manual control instead to show that our approach could still be used to estimate the scaled height and continue propagating it with the decomposed homography.

Fig. 7 shows free-fall and manual control takeover of the thrown MAV, with the latter one starting at approximately 1.15 s after throw detection. As can be seen our framework was able to scale correctly a little after the apex of the trajectory. Moreover, it was able to continue propagating the height estimates quite accurately although still having altitude changes present. This shows us to that a VIO initialization dependent on the current metric height is possible with our proposed approach. We further noted, that the on-going correct height propagation provides sufficient time to initialize the VIO framework once the control has started.

What is more, the mean scale and propagation error for our throws remained within our set limits for the duration of the free-fall, as seen in Fig 8. However, as soon as the MAV control starts at approximately 1.15 s the average error increases. We derived that this is due to the initial thrust experienced by the MAV, which can cause quite some image blur. This leads to the motivation to initialize the scale of the VIO framework within the period of free fall.

VI. CONCLUSIONS

With this paper, we provided a new way for initializing a VIO MAV framework with TnG. We showed that, while in free fall, it is possible to get a sufficiently good estimation for modern VIO system-initialization using our homography based IMU supported approach.

Compared to state-of-the-art, our approach does not need specific assumptions, additional sensors, or an initial guess. All information, including the gravity direction used for subsequent attitude alignment, can be derived from and during the process of the throw. A single camera and IMU are thus sufficient to enable unprepared throw-and-go capability on an MAV.

REFERENCES

[1] S. M. Ettinger, M. C. Nechyba, P. G. Ifju, and M. Waszak, "Vision-guided flight stability and control for micro air vehicles," *Advanced Robotics*, vol. 17, no. 7, pp. 617–640, 2003.

[2] 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*, pp. 225–234, IEEE, 2007.

[3] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO : Fast Semi-Direct Monocular Visual Odometry," in *IEEE International Conference on Robotics and Automation*, 2014.

[4] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems," *IEEE Transactions on Robotics*, vol. PP, no. 99, 2016.

[5] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4531–4537, 2011.

[6] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[7] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3565–3572, 2007.

[8] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *IEEE International Conference on Intelligent Robots and Systems*, 2015.

[9] V. Usenko, J. Engel, J. Stückler, and D. Cremers, "Direct visual-inertial odometry with stereo cameras," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1885–1892, IEEE, 2016.

[10] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," 2018.

[11] S. Weiss, R. Brockers, S. Albrektsen, and L. Matthies, "Inertial optical flow for throw-and-go micro air vehicles," in *Proceedings - 2015 IEEE Winter Conference on Applications of Computer Vision, WACV 2015*, pp. 262–269, 2015.

[12] S. Weiss, R. Brockers, and L. Matthies, "4DoF drift free navigation using inertial cues and optical flow," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 4180–4186, 2013.

[13] M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza, "Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 1722–1729, IEEE, 2015.

[14] E. Malis and M. Vargas, "Deeper understanding of the homography decomposition for vision-based control," Research Report RR-6303, INRIA, 2007.

[15] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4531–4537, 2011.