2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

High Altitude Monocular Visual-Inertial State Estimation: Initialization and Sensor Fusion

Tianbo Liu and Shaojie Shen

Abstract— **Obtaining reliable state estimates at high altitude but GPS-denied environments, such as between high-rise buildings or in the middle of deep canyons, is known to be challenging, due to the lack of direct distance measurements. Monocular visual-inertial systems provide a possible way to recover the metric distance through proper integration of visual and inertial measurements. However, the nonlinear optimization problem for state estimation suffers from poor numerical conditioning or even degeneration, due to difficulties in obtaining observations of visual features with sufficient parallax, and the excessive period of inertial measurement integration. In this paper, we propose a spline-based high altitude estimator initialization method for monocular visual-inertial navigation system (VINS) with special attention to the numerical issues. Our formulation takes only inertial measurements that contain sufficient excitation, and drops uninformative measurements such as those obtained during hovering. In addition, our method explicitly reduces the number of parameters to be estimated in order to achieve earlier convergence. Based on the initialization results, a complete closed-loop system is constructed for high altitude navigation. Extensive experiments are conducted to validate our approach.**

I. Introduction

Autonomous aerial robots are becoming popular for research and for commercial and industrial applications due to their mobility, agility and the ability to achieve both high-speed flight and hovering. Many applications, for aerial robots, such as delivery, infrastructure inspection, and surveillance, involves operations at high altitude. Nowadays, most of the state estimation problems at high altitude are solved by a fusion of Inertial Measurement Unit(IMU) and GPS. However, high altitude does not guarantee good GPS reception. In fact, for operations in urban areas that involve flying between high-rise buildings or in the middle of deep canyons (Fig. 1), GPS is often blocked due to obstructed sky view. There is certainly a high demand in developing state estimation solutions that work at GPS-denied high altitude environments.

At high altitude, the nearest object can be tens or hundreds of meters away. Most active ranging sensors, such as radars, LiDARs, or RGB-D sensors, either do not work at these ranges, or are too heavy for the payload-limited aerial robots. One may opt for passive distance measurement, such as stereo camera. However, stereo camera can degenerate to the monocular case due to the small baseline comparing with the scene depth. A typical high altitude scene is illustrated in Fig.5. To this end, the monocular visual-inertial navigation system (VINS), which consists of a camera and an IMU, becomes a viable choice, as the metric scale, which is crucial for closed-loop control, may be recovered through a proper fusion of visual and IMU measurements.

Fig. 1. Illustrations of environments where GPS signal can be poor even at high altitude, such as in canyon area[1] or "city canyon"[2].

The foremost important step for using monocular VINS is initialization, during which velocity and attitude of the sensor suite, as well as gravity vector and scene depth, are recovered. The same procedure can also be applied for failure recovery of the system. Many of these quantities are not directly observable by the sensors, and appropriate motions are required to render them observable. Interestingly, motion requirements are quite different for visual and inertial sensors. Sufficient parallax between frames is required to achieve good triangulation of visual features, which leads to reliable monocular vision-based pose estimation. At high altitude, this parallax requirement translates into a long path covered by a sliding window of camera poses. Vision-based methods also favor less agile motions to achieve better feature tracking. On the other hand, IMU is best for estimating agile motions over a short period of time. We name IMU measurements that contain nontrivial acceleration as "*informative*" measurements. When the robot is hovering or moving in constant velocity, IMU measurements will degenerate and no longer provide any information about metric scale, which become "*uninformative*" measurements. At high altitude, these two motion requirements are actually contradictory to each other, as a window of camera poses is connected by an excessive number of IMU measurements, out of which many of them are uninformative. On the other hand, agile motions, which are good for IMU, are undesirable for feature tracking in vision modules. All these issues make high altitude initialization a difficult problem.

While there are methods for initializing monocular

The authors would like to thank the studentship and equipment support from DJI.

All authors are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, China. tliuam@connect.ust.hk, eeshaojie@ust.hk

[1] http://se.forwallpaper.com/wallpaper/the-grand-grand-canyon-582271.html

[2] https://500px.com/photo/133873519

Fig. 2. Our quadrotor testbed equipped with a monocular fisheye camera (marked in red), an onboard IMU, and onboard computation. Video for experiment can be accessed at `http://www.ece.ust.hk/~eeshaojie/icra2017tianbo.mp4`

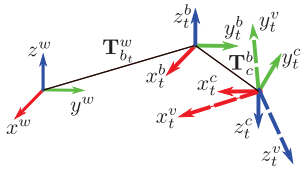


Fig. 3. Frame definition

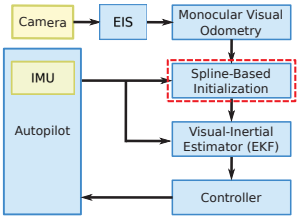


Fig. 4. System architecture. We highlight our contribution in the *Spline-Based Initialization* module.

VINS [1]–[7], none of them works reliably at high altitude due to the issues mentioned above. We observe that tightly-coupled initialization methods [7] that aim to recover all navigation quantities in one shot perform poorly, due to the attempt of solving for a large number of variables in a poorly conditioned system.

In this paper, we propose a spline-based initialization procedure with carefully designed modules to tackle all aforementioned issues at high altitude. Instead of using tightly-coupled approaches, we opt to reduce the number of unknown variables in visual-inertial alignment, by turning to a loosely-coupled formulation. Our formulation builds on top of an accurate local windowed bundle-adjustment-based visual odometry (VO) for up-to-scale pose estimation. The front-end feature tracker of the VO is assisted by an IMU-based electronic image stabilization (EIS) unit that removes all rotational components in the image. This makes our VO reliable even if the aerial robot undergoes aggressive motions. We then fit a B-Spline to the up-to-scale pose estimates, from which the continuous up-to-scale velocity and acceleration profile can be extracted. The visual scale and the gravity can be estimated by minimizing the error between the set of informative accelerometer measurements and the corresponding accelerations extracted from the spline. In this way, we are able to only use informative measurements, which correspond to motions with sufficient excitation, for the estimation of visual scale. The reduction of the number of unknown variables to only 4 (scale and gravity vector) and the exclusion of uninformative measurements significantly improve the numerical stability of the system, making it easy to converge to correct values even at high altitude.

Our initialization method can be used for both tightly-coupled and loosely-coupled VINS estimators. In this paper, we use a loosely-coupled extended Kalman Filter (EKF)-based method as an example for online estimation of position, velocity and orientation of the aerial robot, as well as metric scale for the monocular VO. We specially design the double buffered structure for relative measurements in the EKF to compensate for the large delay caused by the local bundle adjustment. Closed-loop experiment is conducted on our testbed (Fig.2).

In Sect. II, we discuss relevant literature. After the system overview in Sect. III, we detail the monocular VO sub-system in Sect. IV. Our spline-based high altitude estimator initialization is presented in Sect. V. A loosely-coupled, EKF-based visual-inertial fusion method is discussed in Sect. VI. We discuss implementation details and experimental results in Sect. VII. The paper is concluded in Sect. VIII.

II. Related Work

Loosely-coupled monocular visual-inertial systems are widely applied for aerial robots [8]–[11]. Following the classical PTAM proposed by [12], a more detailed solution for the VO module is introduced in [13]. EKF is the most popular method for visual-inertial fusion, where the metric scale is estimated as a dedicated state in the filter [9, 10]. It is mentioned in [9] that poor scale initialization may lead to divergence of the filter.

Tightly-coupled methods for solving the monocular VINS have become popular due to the high accuracy gained by directly fusing raw visual measurements. [14] proposes an EKF framework for solving monocular VINS. Graph optimization-based solutions are presented in [15]–[17]. [16] details an initialization method using optimization over a sliding window of sensor measurements for estimating velocity, attitude, and bias. This method assumes that both sufficient parallax in the visual module and sufficient IMU excitation throughout the initialization period are available.

Several initialization methods are explored for monocular VINS [1]–[5]. [1] proposes a closed-form solution for attitude, velocity, metric scale, and bias. [2] adds extrinsic calibration into the formulation. [6] investigates the effects of different variables on initialization. However, all these methods rely on integrated inertial measurements and visual measurements over a certain time for initialization. This implies that they cannot perform at high altitude due to the large drift in IMU and the existence of a large number of uninformative IMU measurements in a long-time window that satisfies parallax requirement for the visual module. [5] proposes a closed-form solution that utilizes delta-velocity from visual sensors. However, numerical differentiation of visual pose estimates leads to very noisy velocity measurements.

[18] demonstrates state estimation at high altitude with the help of external metric measurements for initialization. [19] proposes a high altitude estimator using a stereo camera. However, stereo camera can only work at limited altitude due to baseline limitations.

III. System Overview

We start with notation and frame definition. $\mathbf{T}_B^A \in \mathbf{SE}(3)$ denotes a transformation from A to B. Using $\mathbf{T}_B^A$, we can transform a vector $\mathbf{v}$ in B to A, expressed as $\mathbf{v}^A = \mathbf{T}_B^A \mathbf{v}^B$. A transformation $\mathbf{T}$ is constructed with rotation $\mathbf{R}$ and translation $\mathbf{p}$: $\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}$. Roll, pitch, yaw represent rotation around x-axis, y-axis and z-axis in $z-y-x$ sequence. In most cases, subscript denotes time, and superscript denotes frame. Frames are defined in Fig.3. World frame w is defined arbitrarily as long as z^w is aligned with gravity vector. At any time t, we have transformation $\mathbf{T}_{b_t}^w$ which transforms from w to robot body frame b. IMU frame coincides with body frame. The extrinsic parameter between IMU and camera is $\mathbf{T}_c^b$. The stabilized camera frame v (by EIS, Sect. IV-A) shares the same origin with camera frame c, but the orientation might be different.

Our system consists of six modules as illustrated in Fig.4. The DJI Autopilot provides IMU data and the interface for control. Raw images are acquired by camera driver.

Raw video stream from a fisheye camera is fed into the EIS for removal of rotational components. A monocular visual odometry (VO) takes stabilized image stream as input and provides up-to-scale pose estimation of the camera. At the VO back-end, a local sliding window bundle adjustment is processed to improve the accuracy of pose estimation (Sect. IV).

The output of monocular VO is fed into the high altitude initialization module. The module fits the position from monocular VO as a smooth B-Spline. The second-order derivative of the fitted B-Spline is taken to get the up-to-scale acceleration. By minimizing the difference between the rotation compensated informative accelerometer readings and the spline-based acceleration, we can recover the metric scale and gravity vector through least square optimization.

An EKF takes the initial values and fuses IMU together with pose estimates from monocular VO, in which the metric scale is continuously refined to account for scale drifting in monocular VO. A double buffer is designed to account for possible delays in the monocular VO. The high bandwidth state estimates from the EKF are used for closing the control loop for high-altitude autonomous navigation.

IV. Monocular Visual Odometry

Our spline-based initialization module (Sect. V) requires high accuracy up-to-scale pose estimates from the monocular VO. As such, we adopt VO pipeline similar to PTAM [12], where the camera poses are obtained through 2D-3D pose estimation, and 3D features are triangulated conditioning on available pose estimates. A local sliding window bundle adjustment is used to improve the accuracy of pose estimation.

The accuracy and reliability of the front-end feature tracking have a significant impact on the final VO result. This is particularly important for high altitude operations as we need long term feature tracking to achieve reliable triangulation. To this end, we apply EIS to all fisheye images to remove the rotational components. The stabilized images only contain translation motions, making them irrelevant to rapid roll/pitch motions of the robot. It reduces the pixel movements, which is beneficial for our optical flow-based feature tracker.

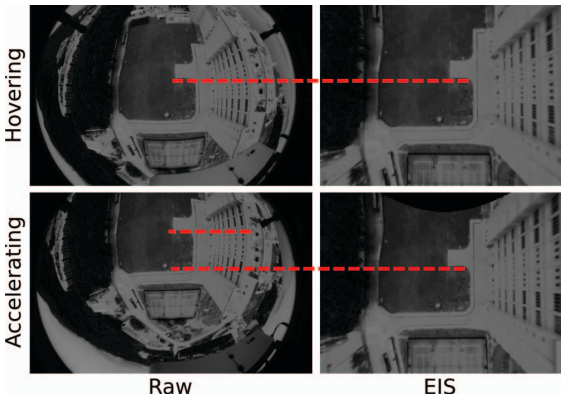


Fig. 5. EIS at high altitude. In left column are raw images from a fisheye camera. In right column are stabilized images by EIS. Accelerating of the quadrotor causes large pattern shifts in raw images. However, in the stabilized images which are rotation compensated, patterns remain at the same position, making it much easier for feature tracking.

A. Electronic Image Stabilization

EIS crops and undistorts a part of the raw fisheye image, such that the stabilized image is free of lens distortion, and is stabilized by pointing at the same global direction regardless of the motion of the robot. EIS takes advantage of the 210 degrees field of view of our fisheye lens. The intuition behind EIS is that, the region of interest (ROI) of a fisheye image can be cropped out and undistorted into another image. This creates a virtual pinhole camera, that is fixed at the same location as the actual fisheye camera, but has the freedom of rotating around three axes.

With a synchronized pair of fisheye image and robot orientation with respect to the world frame, EIS calculates the compensating rotation $\mathbf{R}_v^c$. When this rotation is applied to determine the cropped region, we obtain a stabilized image with its optical axis always pointing towards the ground regardless of robot motion. The transformation between pixels on the stabilized image and the fisheye image can be written as:

$$\mathbf{u}_c = \pi_c \left(\mathbf{R}_v^c \pi_v^{-1} \left(\mathbf{u}_v, 1 \right) \right),$$

where π_c is the projection function of the fisheye camera, with $\mathbf{u}_c$ being the pixel coordinate. Similar convention follows for the stabilized camera (π_v and $\mathbf{u}_v$). Depths are set to 1 since we are only compensating for rotations. A stabilized image can be rendered using bilinear interpolation after all coordinates are calculated. Stabilized images and corresponding compensating rotations $\mathbf{R}_v^c$ are packed in pairs to provide extrinsic parameters for other modules. The fisheye camera model comes from [20], and our implementation makes use of CamOdoCal [21].

We stress that while the EIS module requires the orientation estimation from the IMU, its performance is not affected by the slight error in the IMU orientation estimation. $\mathbf{R}_v^c$ is merely for compensation, and it is not necessary to be

exactly equal to the inverse of the robot orientation. In fact, rough compensation is still able to improve the performance of feature tracking significantly.

B. Keyframe-Based Visual Odometry Pipeline

Corner features are extracted from stabilized images and tracked by the Kanade-Lucas-Tomasi tracker. All features are transformed from the stabilized frame v back to the original image frame c using $\mathbf{R}_v^c$ from EIS.

Given feature correspondences, the monocular VO is initialized by either five-point algorithm or homography. It only initializes when sufficient parallax between the two initial keyframes are obtained. Since the images are stabilized by EIS, we do not need to consider the pure rotation case. From the initial 3D features triangulated by the first two keyframes, we proceed with nonlinear 2D-3D pose estimation by minimizing reprojection error, followed by triangulation of more feature points given the new camera pose. A new keyframe will be inserted when parallax between the current frame and the last keyframe reaches a certain threshold. All camera poses are referenced to the initial vision frame c_0.

C. Local Sliding Window Bundle Adjustment

Every time a new keyframe is added, local sliding window bundle adjustment is processed to improve the accuracy of feature locations and camera poses. Inverse depth feature parameterization [22] is used for better numerical conditioning at high altitude. The optimization can be summarized as:

$$\min_{T_{v_k}^{v_0},\rho_i} \frac{1}{2} \sum r \left(\left\| \mathbf{x}_i^{v_k} - \pi^{-1}\left(\mathbf{T}_{v_k}^{v_0}\right) \begin{bmatrix} \pi\left(\mathbf{x}_i^{v_{k_0}}, {}^1\!/_{\rho_i}\right) \\ 1 \end{bmatrix} \right\|_2 \right), \quad (1)$$

where $k \in \mathcal{W}, i \in \mathcal{M}$. $\mathcal{W}$ is the sliding window of poses. $\mathcal{M}$ is the set of features. $\|\cdot\|_2$ is the L^2 norm. $r(\cdot)$ is the robust loss function. $\pi(\cdot) : \mathbb{R}^3 \longmapsto \mathbb{R}^2$ is the projection function that projects 3D points onto image plane, and $\pi^{-1}(\cdot) : (\mathbb{R}^2, \mathbb{R}) \longmapsto \mathbb{R}^3$ is inverse projection function. $\mathbf{x}_i^{v_k}$ and $\mathbf{x}_i^{v_{k_0}}$ are projections of the i^{th} feature to keyframes v_k and v_{k_0}. Its inverse depth in keyframe v_{k_0} is denoted as ρ_i.

Nonlinear optimization is conducted to minimize reprojection error with robust norm. We lock the first pose and the distance between the first two keyframes in $\mathcal{W}$ to eliminate rank deficiency. Only poses of keyframes in $\mathcal{W}$ and the involved inverse depths of features are optimized to bound computational complexity.

For real-time onboard applications, the VO module should output regularly instead of being blocked. However, it takes hundreds of milliseconds to complete the bundle adjustment, which is unfavorable for a real-time system. Similar to [12], we implement the bundle adjustment as a background thread, and have the main VO thread continuously providing pose estimates by solving 2D-3D registration with respect to the current features.

V. Spline-Based High Altitude Estimator Initialization

This section details a novel high altitude initialization method for monocular VINS. The same procedure can also be used for failure recovery, as it is just a re-initialization of the system on-the-fly. Several states are required to be initialized: velocity, gravity-aligned attitude, and a scale parameter that puts the monocular VO into the metric coordinate.

Our initialization procedure has two key attributes. First, we use a spline-based formulation to allow the inclusion of only informative accelerometer measurements into the initialization process. This prevents the uninformative measurements from putting the system into a poorer numerical conditioning. Second, it reduces the number of parameters by splitting the visual and inertial (metric) initialization, where the visual part is done separately as a monocular VO (Sect. IV). In the inertial initialization, we only need to estimate 4 parameters: the scale, and the gravity vector.

A. B-Spline Fitting of Up-to-Scale Position Estimates

To obtain the second order derivatives of positions, we model the trajectory of the robot as a B-Spline. We do not model the rotation part as we do not optimize it in the initialization phase. Interpolation on rotation can be utilized when necessary. Thanks to the differential flatness of multi-rotor aerial robots [23], the fifth order derivative of position maps to motor angular acceleration. Making the assumption of continuous motor speed, we choose to model robot position as a 6-order B-Spline $\mathbf{p}_s(t) \in \mathbb{R}^3$ defined as:

$$\mathbf{p}_s(t) = \sum_{i=0}^{n} \boldsymbol{\alpha}_i B_{i,d}(t) \quad (2)$$

where $i \in [t_0, t_n]$ is time for control points in a period of time, $\boldsymbol{\alpha}_i \in \mathbb{R}^3$ are the control points at time i, and $B_{i,d(t)}$ are d-th order basis functions, which can be computed with the De Boor - Cox recursive formula [24].

If sensor data in the period $[t_0, t_n]$ contains sufficient motion excitation, a least square problem is solved to fit the B-Spline. We denote the position measurements from VO as $\check{\mathbf{p}}_{t_j}^{c_0}$, where $t_j \in [t_0, t_n]$ is the time for j^{th} pose. We optimized the spline as a least square problem

$$\min_{\boldsymbol{\alpha}_i} \frac{1}{2} \left\| \check{\mathbf{p}}_{t_j}^{c_0} - \mathbf{p}_s(t_j) \right\|_2. \quad (3)$$

B. Solving for Metric Scale and Gravity

Once a spline is fitted, we can get its second order derivative by closed-form differentiation of polynomials. This contrasts to the numerical differentiation used in [5], as our global spline fitting significantly reduces the impact of noise from VO.

Let the linear acceleration in the platform body frame be $\mathbf{a}_t$, and the noisy accelerometer measurement be $\check{\mathbf{a}}_t = \mathbf{a}_t + \mathbf{n}_{a_t}$, where $t \in [t_0, t_n]$ is the time, and $\mathbf{n}_{a_t}$ is the additive Gaussian noise. Comparing the second derivative of the spline $\ddot{\mathbf{p}}_s(t)$ and $\mathbf{a}_t$, it can be seen that $\ddot{\mathbf{p}}_s(t)$ is in frame c_0 while $\mathbf{a}_t$ in body frame b_{t_k}. We have the following relation:

$$\lambda \ddot{\mathbf{p}}_s(t) = \mathbf{R}_{c_t}^{c_0} \mathbf{R}_b^c \mathbf{a}_t - \mathbf{g}^{c_0} \quad (4)$$

where $\mathbf{g}^{b_k}$ is the gravity vector expressed in the body frame. λ is the unknown scale parameter of monocular VO, $\mathbf{R}_b^c$ is

the rotation between camera and IMU. Here we ignore the camera-IMU translation as it does not pose significant effect at high altitude.

It can be observed from (4) that only when $\mathbf{R}_{c_t}^{c_0}\mathbf{R}_b^c\mathbf{a}_t$ has enough acceleration in addition to the gravity, the scale λ can be solvable. Thus, we only sample data to form the set $\mathcal{D}$ at time spots t_k when $\mathbf{a}_{t_k}$ is informative.

Applying (4) in $\mathcal{D}$, we have the optimization problem as

$$\min_{\lambda,\mathbf{g}^{c_0}} \sum_{k\in\mathcal{D}} \left\| \lambda\ddot{\mathbf{p}}_s(t_k) + \mathbf{g}^{c_0} - \mathbf{R}_{c_k}^{c_0}\mathbf{R}_b^c\check{\mathbf{a}}_{t_k} \right\|_2, \tag{5}$$

where k is the index in $\mathcal{D}$. $\mathbf{R}_{c_k}^{c_0}$ can be retrieved by interpolating rotation $\mathbf{R}_t^{c_0}$ from the VO. This is a linear problem with only 4 unknowns and good numerical conditioning due to the choice of data.

C. Additional Implementation Details

When assembling $\mathcal{D}$, we define $\|\check{\mathbf{a}}_t - \check{\mathbf{a}}_{avg}\|_2 >= \epsilon_{\mathbf{g}}$ as the criteria for informative measurements, where $\epsilon_{\mathbf{g}}$ is a tunable threshold, and $\check{\mathbf{a}}_{avg}$ is the averaged acceleration reading in $[t_0, t_n]$. We set $\epsilon_{\mathbf{g}} = 0.2$ m/s^2.

After the VO is up and running, the initialization module will collect all VO poses with IMU data. At least 5 seconds of data is collected. If sufficient number of informative IMU is collected, the initialization module starts by first using B-Spline fitting to find $\mathbf{p}_s(t)$, and then using optimization to find the estimated scale $\hat{\lambda}$ and gravity vector $\hat{\mathbf{g}}^{c_0}$.

Since B-Spline trends to be over-fitting near the beginning and the end of the sliding window, we choose the latest B-Spline control point before end point as the initial time t_{init}. Together with the estimated scale $\hat{\lambda}_v$, the first order derivative of $\mathbf{p}_s(t)$ is taken at t_{init} to obtain the initial velocity:

$$\hat{\mathbf{v}}_{init}^{c_0} = \hat{\lambda}\dot{\mathbf{p}}_s(t_{init}). \tag{6}$$

In addition, we can recover rotation $\hat{\mathbf{R}}_{c_0}^w$ according to $\hat{\mathbf{g}}^{c_0}$ up to an arbitrary yaw angle. Then velocity in world frame can be calculated as:

$$\hat{\mathbf{v}}_{init}^w = \hat{\mathbf{R}}_{c_0}^w \hat{\mathbf{v}}_{init}^{c_0}. \tag{7}$$

Until now, we have initialized attitude $\hat{\mathbf{R}}_{c_0}^w$, velocity $\hat{\mathbf{v}}_{init}^w$ and the scale $\hat{\lambda}$ for monocular VO.

VI. EKF-Based Visual-Inertial Fusion

Initialized states obtained from the method described in Sect. V can be used with both tightly-coupled or loosely-coupled VINS estimators. Here we use a loosely-coupled EKF as an example. Monocular VO and IMU are fused to get a more accurate estimation with lower latency for closed-loop control. The filter predicts states using IMU propagation while processing measurement update using VO poses.

Similar to [10], we treat VO poses as up-to-scale relative measurements. This is consistent with the nature of keyframe-based VO. It is also aligned with the nature of scale, which is a local relative quantity. Finally, it can handle the complex temporal relationship caused by large delays, which is the side effect of bundle adjustment.

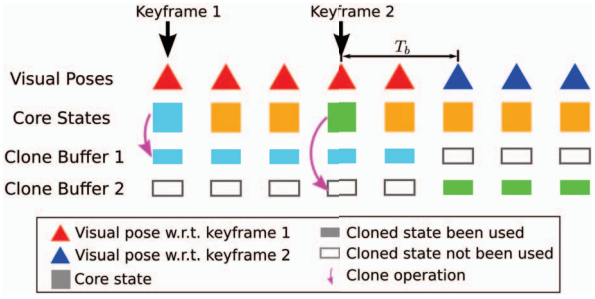


Fig. 6. Illustration of interaction between monocular VO and EKF. Blue state is cloned to buffer 1, green state is cloned to buffer 2. Visual measurements with respect to keyframe 1 are updated using buffer 1, and measurements to keyframe 2 are updated using buffer 2. Extra buffer space is used to save states that might be used in the upcoming updates. T_b is time spent in bundle adjustment. Details are described in Sec.VI-B.

We assume that extrinsic parameters (i.e. rotation and translation between IMU and camera) are fixed from offline calibration. The filter works the same way as common EKF, thus we do not discuss the standard framework here. Readers may refer to [10] for the standard EKF framework.

A. Core States and Propagation of EKF

We define the core states of the filter as $\mathbf{x}_{core} = \begin{bmatrix}\mathbf{p}_t^w & \mathbf{v}_t^w & \mathbf{q}_t^w & \mathbf{b}_a & \lambda_v\end{bmatrix}^T$, which are position, velocity, attitude of the robot in world frame, accelerometer bias, and scale of the monocular VO. We use the error state formulation for handling rotation error and covariance. It is represented as $\delta\mathbf{x}_{core} = \begin{bmatrix}\delta\mathbf{p}_t^w & \delta\mathbf{v}_t^w & \delta\boldsymbol{\theta}_t^w & \delta\mathbf{b}_a & \delta\lambda_v\end{bmatrix}^T$. Further details about the error formulation can be found in [9]. State propagation is driven by accelerometer $\mathbf{a}_t^b$ and gyroscope $\boldsymbol{\omega}_t^b$, with the same propagation model in [10].

As mentioned in Sect. V, we can initialize $\mathbf{R}_{c_0}^w$, $\mathbf{v}_{init}^w$ and λ into the filter. Here, we initialize λ_v as $1/\lambda$ since the two scale factors are multiplied at different variables. The accelerometer bias is set to $\mathbf{0}$. Position is set arbitrarily.

B. Double Buffered State Cloning

We use stochastic cloning [25] to handle relative measurements, with a double buffered mechanic to compensate for the large latency caused by bundle adjustment. We reserve double space for cloned states. They are cloned and propagated in the same way as in [25]. Fig.6 illustrates the interaction between monocular VO and EKF. Assume that up to now all poses are with respect to keyframe at time t_{k_1}. The VO decides to insert a new keyframe at time t_{k_2}, at which EKF should clone the core states. However, when bundle adjustment is running for time period T_b, all poses are still with respect to k_1. If there is only one group of cloned states, measurements in T_b will update with cloned states at t_{k_2}, while actually they are with respect to k_1. With the double buffer for cloned states, we can clone core states at t_{k_2} to buffer 2 while keep updating measurements in T_b with cloned states at t_{k_1} in buffer 1. As a result, consistency between monocular VO and EKF is maintained.

C. Measurement Update

Poses from monocular VO are transformed to body frame and updated as relative measurements. Similar to [10], position error can be defined as

$$\begin{aligned}\delta\mathbf{z}_p &= \check{\mathbf{z}}_p - \hat{\lambda}_v\hat{\mathbf{z}}_p \\ &= (\check{\mathbf{R}}_k^0)^T\left(\check{\mathbf{p}}_t^0 - \check{\mathbf{p}}_k^0\right) - \hat{\lambda}_v(\hat{\mathbf{R}}_k^w)^T\left(\hat{\mathbf{p}}_t^w - \hat{\mathbf{p}}_k^w\right),\end{aligned} \tag{8}$$

where $\hat{(\cdot)}$ denotes estimated variables, $\check{(\cdot)}$ denotes measurements transformed to body frame. Subscript k is the time of the keyframe, and t is the time of current frame. Rotation error can be modeled as

$$\begin{aligned}\delta\mathbf{z}_q &= 2\left[\check{\mathbf{z}}_q - \hat{\mathbf{z}}_q\right]_{xyz} \\ &= 2\left[(\check{\mathbf{q}}_k^w)^{-1}\otimes\check{\mathbf{q}}_t^w\otimes\left((\hat{\mathbf{q}}_k^w)^{-1}\otimes\hat{\mathbf{q}}_t^w\right)^{-1}\right]_{xyz} \\ &= 2\left[\check{\mathbf{q}}_t^{t_k}\otimes(\mathbf{q}_t^w\otimes\delta\hat{\mathbf{q}}_t^w)^{-1}\otimes(\mathbf{q}_k^w\otimes\delta\hat{\mathbf{q}}_k^w)\right]_{xyz}.\end{aligned} \tag{9}$$

Finally, measurement update can be written as:

$$\begin{bmatrix}\delta\mathbf{z}_p\\ \delta\mathbf{z}_q\end{bmatrix} = \mathbf{H}\delta\hat{\mathbf{x}}, \tag{10}$$

where $\mathbf{H}$ is the Jacobian linearized with respect to the filter error states $\delta\hat{\mathbf{x}}$.

VII. Experimental Results

A. Implementation Details

We use the DJI M100[1] for real-time closed-loop control experiment. All algorithms run onboard an Intel NUC computer (i5-4250U, 16GB RAM). The monocular camera is a mvBlueFOX-MLC200wG equipped with a 210° fisheye lens. IMU data is acquired from the autopilot on M100.

All modules are written in C++, using ROS[2] as communication middleware. Ceres-solver [26] is applied for camera pose estimation and local windowed bundle adjustment. B-Spline fitting is conducted using GSL[3]. Image sequence is at 30 Hz, and the monocular VO runs at the same frequency. EKF runs at 100 Hz, which is the frequency of IMU sequence from the DJI autopilot. Nonlinear camera pose estimation for a single pose takes about 30 ms. Local windowed bundle adjustment takes 100 $\sim$ 300 milliseconds according to the quantity and structure of features and keyframes.

B. Performance of Up-to-scale Monocular Visual Odometry

As shown in Fig.7, up-to-scale visual odometry is aligned and scaled manually with respect to GPS trajectory. Final position $\mathbf{p}_{end} = [-1.410, 0.812, -3.526]$. The overall error is 3.88 m, which concludes 0.33% accuracy compared to overall distance 1165.87 m. Here we are not highlighting the low overall error which is not very meaningful, when the visual trajectory is rescaled manually. However, we would like to emphasize that: First, in a short period of time (e.g. 30s, 1min), visual odometry has no scale drift, and is accurate enough for inertial initialization. Second, the up-to-scale VO can be manually scaled to match with the metric measurement (GPS) using only one global scale parameter. This matches the reduced dimensionality approach in our algorithm.

[1] https://developer.dji.com/matrice-100/
[2] http://www.ros.org/
[3] http://www.gnu.org/software/gsl/

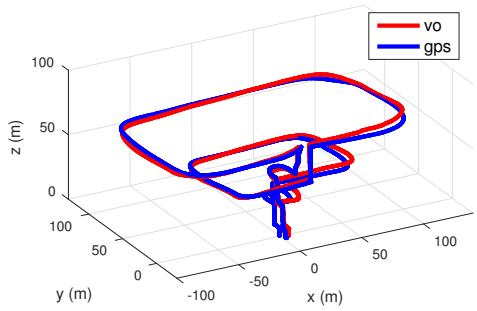


Fig. 7. Monocular visual odometry compared with GPS. The odometry is rescaled and aligned manually w.r.t. GPS trajectory.

C. Performance of High Altitude Estimator Initialization

In order to show the "launch anywhere" capability of our algorithm, we run our initialization at two time instances. The first trial starts when hovering. The second trial starts several seconds after the first trial is started, when the robot has already moved. We would like to prove that our algorithm converges to the same result when choosing two near but different sequences for initialization. The altitude is about 40 meters high.

Fig.8 compares B-Splines fitting against monocular VO poses. Knots are chosen uniformly every one second when constructing the B-Spline. We set 0 as the start time of the first trial, and the second trial starts at 7.5 sec. We can see that the fitted curves align well with VO poses, which means that the robot motion can be fully modeled by the B-Splines.

With fitted B-Spline, scale and gravity are solved by aligning second order derivative $\ddot{\mathbf{p}}_s(t)$ with respect to the gravity-compensated acceleration measurements $\bar{\mathbf{a}}_t = \mathbf{R}_{c_k}^{c_0}\mathbf{R}_b^c\check{\mathbf{a}}_t - \hat{\mathbf{g}}^{c_0}$. In Fig.9 we can see that $\hat{\lambda}\ddot{\mathbf{p}}_s(t)$ approximately equals to $\bar{\mathbf{a}}_\mathbf{t}$ extracted from IMU. Although not all details are represented by $\hat{\lambda}\ddot{\mathbf{p}}(t)$, the overall shape suggests that we can recover both scale and gravity. Define alignment error as

$$\epsilon_a = \|\hat{\lambda}\ddot{\mathbf{p}}_s(t) - \bar{\mathbf{a}}_t\|_2.$$

Standard deviation of ϵ_a is 0.26 m/s^2 for the first trial and 0.53 m/s^2 for the second trial. We can also see that the first trial, which contains less excited motion, takes longer time than the second trial.

D. Performance of EKF Fusion

Once initialized, the EKF estimates position, velocity, orientation, accelerometer bias, and visual scale. In Fig.10, EKF trajectory and GPS trajectory are compared by manually aligning the initial sequence. It shows that the EKF trajectory matches well with ground truth with correct scale and minor drift in position.

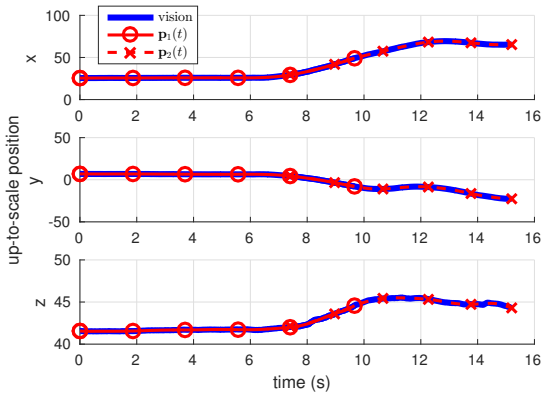


Fig. 8. Fitted splines versus up-to-scale VO. VO trajectory is in blue. Solid red line with circle markers and dashed red line with cross markers represent two different trails. Altitude is at about 40 m.

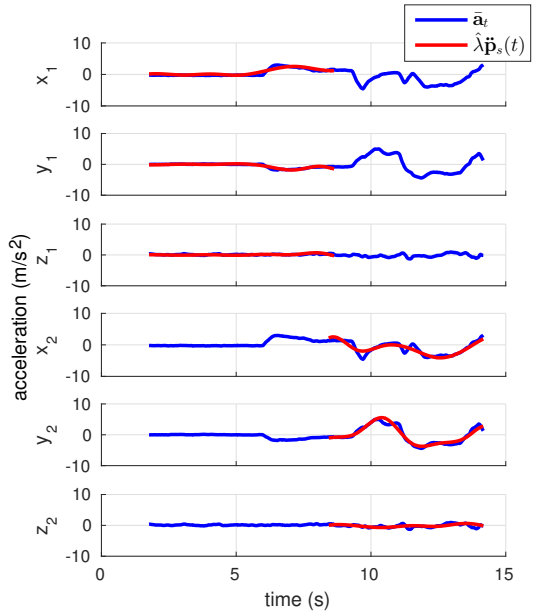


Fig. 9. Second order derivative of the rescaled B-Spline $\hat{\lambda}\ddot{\mathbf{p}}_s(t)$ (red) and gravity-compensated acceleration measurements $\bar{\mathbf{a}}_t$ (blue) approximately coincide. Red curves are fitted B-Spline in the two trials multiplied by the estimated scale respectively. Time for the two trials are aligned for comparison.

Fig.11 shows the comparison of GPS body frame velocity and EKF velocity. The velocity estimated by EKF approximately equals to that of GPS, thus the scale is fully estimated with respect to the ground truth. Velocity error is defined as

$$\epsilon_x = v^b_{x,ekf} - v^b_{x,gps}.$$

ϵ_y, ϵ_z are defined similarly. Standard deviation of $\epsilon_x, \epsilon_y, \epsilon_z$ are 0.35 m/s, 0.30 m/s, 0.35 m/s. We additionally note that GPS may not be accurate enough to serve as the ground truth. We use it in our case due to the lack of other ground truth sources.

E. Closed-Loop Control Experiment

We conducted the closed-loop control experiment outdoor using fused odometry from EKF as feedback. All modules

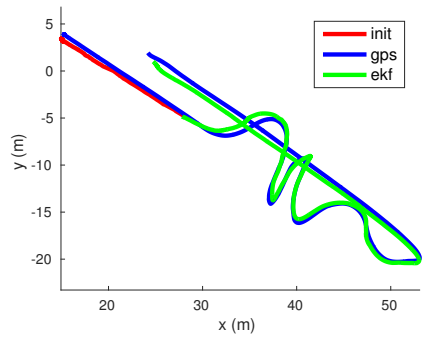
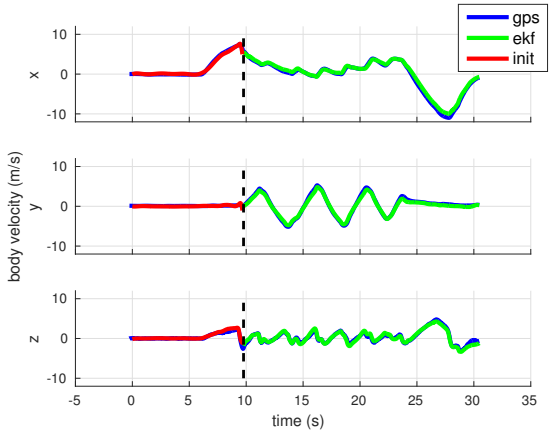
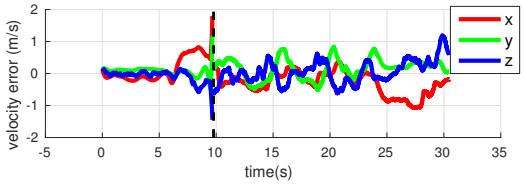


Fig. 10. X-y plot of trajectories. Rescaled spline $\hat{\lambda}\mathbf{p}_s(t)$ (in red) and position estimated by EKF (in green) compare well with respect to GPS trajectory (in blue).



(a) Velocity comparison



(b) Velocity error

Fig. 11. Comparison of EKF and GPS velocities in the robot body frame. Rescaled spline velocity transformed to body frame $\lambda\dot{\mathbf{p}}^b(t)$ is in red. Body velocity estimated by EKF is in green. GPS velocity projected to body frame is in blue.

ran onboard and were launched in air at about 30 meters high. As shown in Fig.12, after initialization, the quadrotor autonomously flew two figure-eight patterns, followed with an elevation to about 50 meters and repeated similar patterns. Body velocity is estimated as illustrated in Fig.13. The whole system is proved to be capable for high altitude task with only inertial and visual sensors.

In addition, the trajectory of EKF drifts away from GPS as traveled distance grows primarily due to the unobservability of the global position. We do notice minor scale drift due to very limited accelerations at high altitude operations. Accelerometer bias terms are almost unobservable in these cases, which might incorrect the result of scale estimation as all metric information comes from the accelerometer.

VIII. Conclusion and Future Work

In this paper, we propose a novel spline-based estimator initialization method for high altitude operations. Experiments show the performance of our initialization method, and prove the functionality of our system at high altitude. Future work may include integrating the initialization method with a tightly-coupled system. The initialization accuracy may further be improved if we jointly optimize both spline fitting and scale initialization.

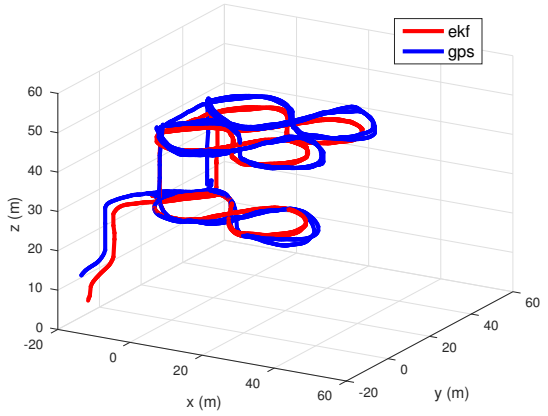


Fig. 12. 3-D trajectory for closed-loop control test with GPS comparison.

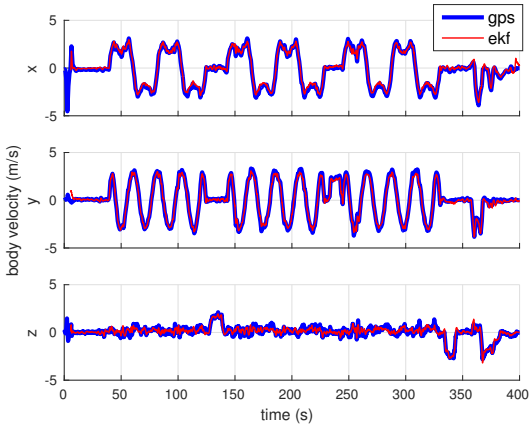


Fig. 13. Body velocity in the closed-loop control experiment. GPS body velocity is in blue, EKF body velocity is in red.

References

[1] A. Martinelli, "Closed-form solution of visual-inertial structure from motion," *Intl. J. Comput. Vis.*, vol. 106, no. 2, pp. 138–152, 2014.

[2] T. C. Dong-Si and A. I. Mourikis, "Estimator initialization in vision-aided inertial navigation with unknown camera-IMU calibration," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Oct 2012, pp. 1064–1071.

[3] L. Kneip, A. Martinelli, S. Weiss, D. Scaramuzza, and R. Siegwart, "Closed-form solution for absolute scale velocity determination combining inertial measurements and a single feature correspondence," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, May 2011, pp. 4546–4553.

[4] V. Lippiello and R. Mebarki, "Closed-form solution for absolute scale velocity estimation using visual and inertial data with a sliding least-squares estimation," in *21st Mediterranean Conference on Control and Automation*, June 2013, pp. 1261–1266.

[5] L. Kneip, S. Weiss, and R. Siegwart, "Deterministic initialization of metric state estimation filters for loosely-coupled monocular vision-inertial systems," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Sept 2011, pp. 2235–2241.

[6] J. Kaiser, A. Martinelli, F. Fontana, and D. Scaramuzza, "Simultaneous state initialization and gyroscope bias calibration in visual inertial aided navigation," *IEEE Robot. and Autom. Lett.*, vol. 2, no. 1, pp. 18–25, Jan 2017.

[7] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Initialization-free monocular visual-inertial estimation with application to autonomous MAVs," in *Proc. of the Intl. Sym. on Exp. Robot.*, 2014.

[8] K. Jonghyuk and S. Salah, "Real-time implementation of airborne inertial-SLAM," *IEEE Robot. and Auton. Syst.*, vol. 55, no. 1, pp. 62–71, 2007.

[9] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, May 2011, pp. 4531–4537.

[10] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to MAV navigation," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.* IEEE, 2013, pp. 3923–3929.

[11] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, May 2014, pp. 4974–4981.

[12] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nov. 2007.

[13] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robot. Autom. Mag.*, vol. 18, no. 4, pp. 80–92, 2011.

[14] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, April 10-14 2007, pp. 3565–3572.

[15] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Information fusion in navigation systems via factor graph based incremental smoothing," *IEEE Robot. and Auton. Syst.*, vol. 61, no. 8, pp. 721–738, 2013.

[16] Z. Yang and S. Shen, "Monocular visual-inertial state estimation with online initialization and camera-IMU extrinsic calibration," *IEEE Trans. Autom. Sci. Eng.*, vol. PP, no. 99, pp. 1–13, 2016.

[17] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *Intl. J. Robot. Research*, vol. 34, no. 3, pp. 314–334, 2015.

[18] F. Caballero, L. Merino, J. Ferruz, and A. Ollero, "Vision-based odometry and SLAM for medium and high altitude flying UAVs," *Journal of Intelligent and Robotic Systems*, vol. 54, no. 1, pp. 137–161, 2009.

[19] W. Michael and U. Ben, "High altitude stereo visual odometry," in *Proc. of Robot.: Sci. and Syst.*, June 2013.

[20] C. Mei and P. Rives, "Single view point omnidirectional camera calibration from planar grids," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2007, pp. 3945–3950.

[21] L. Heng, M. Bürki, G. H. Lee, P. Furgale, R. Siegwart, and M. Pollefeys, "Infrastructure-based calibration of a multi-camera rig," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2014, pp. 4912–4919.

[22] J. Civera, A. J. Davison, and J. M. M. Montiel, "Inverse depth parametrization for monocular SLAM," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 932–945, Oct 2008.

[23] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, May 2011, pp. 2520–2525.

[24] S. Lovegrove, A. Patron-Perez, and G. Sibley, "Spline Fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2013.

[25] S. I. Roumeliotis and J. W. Burdick, "Stochastic Cloning: A generalized framework for processing relative state measurements," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, vol. 2, 2002, pp. 1788–1795 vol.2.

[26] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.