Stereo Visual Inerial Odometry for UAV

*

Chunshang Li
Institute for Aerospace Studies
University od Toronto
Toronto, Canada
chunshang.li@mail.utoronto.ca

Chengyao Li
Institute for Aerospace Studies
University od Toronto
Toronto, Canada
chengyao.li@mail.utoronto.ca

***Abstract*—The usage of Unmanned Aerial Vehicles (UAVs) is increasing dramatically, manifested through applications such as inspection, monitoring, mapping, and safety surveillance. Being able to localize the UAV itself while gather information about the environment is crucial for many of these applications. In the past, research has shown that odometry for UAVs can be accomplished with vision systems. However, if there are not enough features in the environment or if the motion of the UAV is aggressive, an odometry system that purely relies on vision often does not work well. To fix this issue, combining inertial measurements with visual measurements has been a common approach. This report presents the process of investigating and implementing the current state-of-the-art visual inertial odometry. A naive frame to frame visual odometry is studied and modified into a visual inertial odometry using bundle adjustment.**

***Index Terms*—Localization; Aerial Systems: Perception and Autonomy;**

I. Introduction

In recent years, Unmanned Aerial Vehicles (UAVs) have become a key research area for military and civilian applications. Localization is essential for many of these applications, such as mapping and package delivery. In certain scenarios, a global positioning system (GPS) may not be either available or reliable enough. As a result, in order for a UAV to navigate in an uncontrolled environment more robustly, visual odometry or SLAM is often adopted to provide the estimated states. Combining both visual and inertial measurements has been a common and popular means for addressing localization tasks such as visual odometry and SLAM [1]. Vision measurements in images and inertial measurements given by gyroscopes and accelerometers have been proven to perform well in GPS-denied environments to provide estimations of an autonomous robot and environment [1].

Visual-inertial pose estimation problem can be performed using an filter-based approach or a optimization-based. Regarding the filter-based method, the current state-of-the-art algorithm is S-MSCKF [2]. They estimate the state of the micro aerial vehicle using Multi-State Constraint Kalman Filter. This method takes advantage of the inertial measurements from IMU. To solve the IMU synchronization issue, they employ a 4th order Runge-Kutta numerical integration to propagate the estimated IMU state. The authors demonstrate that their algorithm provides better robustness while requiring similar computational power compared with the current state-of-art monocular solutions. They also declare their work is the first open-source filter-based stereo VIO that does not require GPU acceleration to run on a common laptop. In terms of the optimization-based method, the first do to so successfully was Open Keyframe-based Visual-Inertial SLAM (OKVIS) [8]. OKVIS formulated visual and inertial measurements into a single optimization problem, and demonstrated superior results than a vision only method. However, their formulation requires to redo IMU integration with every single step of the optimization, which is not computationally efficient. This is solved by using IMU pre-integration, which is thoroughly discussed in [6]. IMU pre-integration formulates IMU integration in a way such that the integral only need to be calculated once, thus increasing computational efficiency. Qin et al. [3] proposed a monocular tightly-coupled visual inertial full SLAM system, which is called VINS-Mono. It uses a single camera and a low-cost IMU to provide a robust and versatile state estimation. The major components of VINS-Mono include initialization, camera and IMU measurements processing, relocalization, and global pose optimization. In the initialization step, they propose a loosely-coupled sensor fusion method, which aligns metric IMU pre-integration with the visual-only Structure from Motion (SfM) results, to recover the scale. In the vision front end, all the features are tracked using KLT sparse optical flow algorithm and use keyframes to reduce the computational cost. This paper uses IMU pre-integration formula from [6]. The challenge of integrating IMU is that IMU has a higher rate than cameras and thus it is impossible to estimate the state of the camera at all the IMU states. One solution to this problem is to perform IMU pre-integration. They extend the IMU pre-integration by incorporating IMU bias correction in this paper. IMU state propagation requires rotation, position, and velocity of the body frame, and when the pose is adjusted the IMU measurements need to be re-propagated, which is computational demanding. The IMU pre-integration technique reduces the times of performing re-propagation and thus reduces the computational cost. A sliding window and marginalization scheme are also used to reduce the computational cost.

The objective of this project is to investigate and implement the current state-of-the-art IMU integration method. We adopted the IMU formulation from VINS-MONO. Due to the time limit, we choose to modify an existing SLAM algorithm which is Pro-SLAM [4]. It is is a stereo visual SLAM system that is based on a frame to frame optimization. We firstly modify the frame to frame optimization to be a bundle adjustment, and afterwards IMU measurement constraints are added to the pose graph. The entire project is coded in C++ and the optimization is implemented using g2o library. We select to use EuRoc dataset to evaluate our algorithm, and it is determined that by adding bundle adjustment and inertial measurements, the error of the odometry can be reduced.

In the remainder of this report, we cover the measurement pre-processing in II and the back-end optimization in III. In IV, we compare results obtained with our visual inertial odometry with the original Pro-SLAM. Finally, we summarize this report and the project and suggest potential future work.

II. Measurement Preprocessing

This section presents the preprocessing steps for both visual and inertial measurements. For visual measurements, we adopt the algorithm from Pro-SLAM. For inertial measurements, we follow the same IMU pre-integration approach as [3]. We use the notation introduced in Barfoot et al [5], $\mathbf{C}_{v_k i}$ is the rotation of inertial frame i with respect to vehicle frame at time k v_k, $\mathbf{p}_i^{v_k i}$ is position of frame v_k with respect to frame i expressed in frame i. $(\hat{\cdot})$ is used to denote noisy or estimated quantities.

A. Vision Processing

For vision measurements, a standard pinhole camera model is assumed for all projection related operations. For each pair of images, ORB descriptor is used to detect and extract the features. In the next step, a regularization unit is applied to ensure that all the keypoints are evenly distributed in the left image [4]. The regularization unit divides all the keypoints into bins arranged as a fine grid on the image. For each bin, only the "best" feature is kept while all the other keypoints are discarded. Since it is assumed that all the input pictures are undistorted and rectified, the stereo keypoint pairs can be determined by searching only on the epipolar line, and the best match is found by the Hamming distance between the descriptors. Afterwards, triangulation is performed to find the depth and the 3D position of the landmarks, and the landmark correspondences are determined by projecting the landmark in the previous frame into the current left camera frame.

Given the 3D position and correspondences of the landmarks, we can derive the camera projection measurement model. Firstly, the 3D landmark position is projected into the camera frame as follows:

$$\mathbf{p}_{c_k}^{p_j c_k} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{C}_{cv}(\mathbf{C}_{iv_k}^T(\mathbf{p}_i^{p_j i} - \mathbf{r}_i^{v_k i}) - \mathbf{r}_v^{cv}) \tag{1}$$

where $\left\{\mathbf{r}_i^{v_k i}, \mathbf{C}_{v_k i}\right\}$ is the transform from world frame to IMU frame and $\left\{\mathbf{r}_v^{cv}, \mathbf{C}_{cv}\right\}$ is the transform from IMU frame to camera frame. $\mathbf{p}_i^{p_j i}$ is the landmark position in world frame. Then the landmark position is projected from the camera frame to the image coordinate plane as follow.

$$\begin{bmatrix} u_l \\ v_l \\ u_r \end{bmatrix} = \pi(\mathbf{p}_{c_k}^{p_j c_k}) = \frac{1}{z}\begin{bmatrix} f_u x \\ f_v y \\ f_u(x-b) \end{bmatrix} + \begin{bmatrix} c_u \\ c_v \\ c_u \end{bmatrix} \tag{2}$$

where u_l, v_l are the left image coordinates, u_r is the x coordinate in the right image, c_u, c_v are the principle points and b is the baseline.

B. IMU Pre-integration

We adopted the same IMU pre-integration scheme as [3] which is also similar to [6]. The difference is that [3] chose to represent orientation in quaternions which leads to slightly different formulations. Quaternions are ideal for rotation representation not only because they are more compact, it is also much easier to correct for numerical problems through a simple re-normalization. In contrast, a rotation matrix requires re-orthogonalization to correct for numerical errors, which is a more involved process. To derive the equations for pre-integration, we start with the IMU measurement model. Note that in this model, we neglect the rotation of the earth since it is assumed the applications uses low-grade IMU that is unable to measure the small difference. $\hat{\mathbf{a}}_{v_k}^{v_k i}$ and $\hat{\boldsymbol{\omega}}_{v_k}^{v_k i}$ are the accelerometer and gyroscope measurements returned from the sensor, $\mathbf{b}_{a_t}$ and $\mathbf{b}_{\omega_t}$ are the biases, and $\mathbf{n}_a$ and $\mathbf{n}_\omega$ are the sensor noise.

$$\hat{\mathbf{a}}_{v_k}^{v_k i} = \mathbf{a}_{v_k}^{v_k i} + \mathbf{b}_{a_t} + \mathbf{C}_{v_k i}\mathbf{g}_i + \mathbf{n}_a \tag{3}$$

$$\hat{\boldsymbol{\omega}}_{v_k}^{v_k i} = \boldsymbol{\omega}_{v_k}^{v_k i} + \mathbf{b}_{\omega_t} + \mathbf{n}_\omega \tag{4}$$

The accelerometer and gyroscope additive noise and the bias random walk are modelled as zero mean Gaussian.

$$\mathbf{n}_a \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_a^2), \mathbf{n}_\omega \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_\omega^2) \tag{5}$$

$$\dot{\mathbf{b}}_{a_t} = \mathbf{n}_{b_a}, \dot{\mathbf{b}}_{\omega_t} = \mathbf{n}_{b_\omega} \tag{6}$$

$$\mathbf{n}_{b_a} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_{\mathbf{b}_a}^2), \mathbf{n}_{b_\omega} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_{\mathbf{b}_\omega}^2) \tag{7}$$

The IMU measurements between two images at frames v_k and v_{k+1} are integrated to obtain the change in position, velocity, and orientation. $\Delta t = t_{k+1} - t_k$ is the elapsed time between two image frames. Rearranging (3), integrating it once to get velocity and again to obtain the position. Similarly for the gyroscope, (4) is rearranged and integrated once for change in orientation. Note the operator $\otimes$ multiples two quaternions.

$$\begin{aligned} \mathbf{p}_i^{v_{k+1} i} = \mathbf{p}_i^{v_k i} + \mathbf{v}_i^{v_k i}\Delta t_k + \\ \iint_{t_k}^{t_{k+1}} (\mathbf{C}_{it}(\hat{\mathbf{a}}_{v_k}^{v_k i} - \mathbf{b}_{a_t} - \mathbf{n}_a) - \mathbf{g}_i) dt^2 \end{aligned} \tag{8}$$

$$\mathbf{v}_i^{v_{k+1} i} = \mathbf{v}_i^{v_k i} + \int_{t_k}^{t_{k+1}} (\mathbf{C}_{it}(\hat{\mathbf{a}}_{v_k}^{v_k i} - \mathbf{b}_{a_t} - \mathbf{n}_a) - \mathbf{g}_i) dt \tag{9}$$

$$\begin{aligned} \mathbf{q}_{iv_{k+1}} = \mathbf{q}_{iv_k} \otimes \\ \int_{t_k}^{t_{k+1}} \frac{1}{2}\mathbf{q}_{v_k t} \otimes \begin{bmatrix} 0 \\ \hat{\boldsymbol{\omega}}_{v_k}^{v_k i} - \mathbf{b}_{\omega_t} - \mathbf{n}_\omega \end{bmatrix} dt \end{aligned} \tag{10}$$

Using (8), (9) and (10) directly in optimization is not ideal. Within each integral is $\mathbf{C}_{it}$ which is global rotation, it contains the quantity $\mathbf{C}_{iv_k}$ which is the state we will be optimizing for. With every iteration of the optimizer, $\mathbf{C}_{iv_k}$ will change, this will force us to recompute all the integrals at every step of the optimization which will be very computationally expensive. To get around this, (8), (9) and (10) are rearranged such that the integrals only has terms with respect to frame v_k. The pre-integration terms are denoted as $\boldsymbol{\alpha}_{v_k v_{k+1}}$, $\boldsymbol{\beta}_{v_k v_{k+1}}$, and $\gamma_{v_k v_{k+1}}$.

$$\begin{aligned} \mathbf{p}_i^{v_{k+1}i} &= (\mathbf{p}_i^{v_k i} + \mathbf{v}_i^{v_k i}\Delta t_k - \frac{1}{2}\mathbf{g}_i \Delta t_k^2) + \\ &\mathbf{C}_{iv_k} \underbrace{\iint_{t_k}^{t_{k+1}} \mathbf{C}_{v_k t}(\hat{\mathbf{a}}_{v_k}^{v_k i} - \mathbf{b}_{a_t} - \mathbf{n}_a) dt^2}_{\boldsymbol{\alpha}_{v_k v_{k+1}}} \qquad (11) \end{aligned}$$

$$\begin{aligned} \mathbf{v}_i^{v_{k+1}i} &= (\mathbf{v}_i^{v_k i} - \mathbf{g}_i \Delta t_k) + \\ &\mathbf{C}_{iv_k} \underbrace{\int_{t_k}^{t_{k+1}} \mathbf{C}_{v_k t}(\hat{\mathbf{a}}_{v_k}^{v_k i} - \mathbf{b}_{a_t} - \mathbf{n}_a) dt}_{\boldsymbol{\beta}_{v_k v_{k+1}}} \qquad (12) \end{aligned}$$

$$\begin{aligned} \mathbf{q}_{iv_{k+1}} &= \mathbf{q}_{iv_k} \otimes \\ &\underbrace{\int_{t_k}^{t_{k+1}} \frac{1}{2}\mathbf{q}_{v_k t} \otimes \begin{bmatrix} 0 \\ \hat{\boldsymbol{\omega}}_{v_k}^{v_k i} - \mathbf{b}_{\omega_t} - \mathbf{n}_\omega \end{bmatrix} dt}_{\gamma_{v_k v_{k+1}}} \qquad (13) \end{aligned}$$

$$\mathbf{z}_{v_k v_{k+1}} = \begin{bmatrix} \boldsymbol{\alpha}_{v_k v_{k+1}} \\ \boldsymbol{\beta}_{v_k v_{k+1}} \\ \gamma_{v_k v_{k+1}} \end{bmatrix} \qquad (14)$$

Even with the $\mathbf{C}_{iv_k}$ outside of the integral, the bias terms b_{a_t} and b_{w_t} at time t_k are also states we will be optimizing for, which will change at each iteration of optimization. To deal with the change in biases without recomputing the pre-integrals, a first-order approximation is used. This approximation is valid since the biases, by nature, are slowly changing quantities, optimization will change biases by a tiny amount. This approximation will be presented later.

The IMU pre-integration values are perturbed by noise. The perturbations are defined as follows. The pre-integration position $\boldsymbol{\alpha}_{v_k t}$, velocity $\boldsymbol{\beta}_{v_k t}$, and biases $\mathbf{b_{a_t}}$ and $\mathbf{b}_{\omega_t}$ are in the Euclidean space, thus the perturbation is simply added to the nominal value. Orientation is an element of the $SO(3)$ group, the perturbation is applied through a small angle approximation of the quaternion.

$$\boldsymbol{\alpha}_{v_k t} = \hat{\boldsymbol{\alpha}}_{v_k t} + \delta\boldsymbol{\alpha}_{v_k t} \qquad (15)$$

$$\boldsymbol{\beta}_{v_k t} = \hat{\boldsymbol{\beta}}_{v_k t} + \delta\boldsymbol{\beta}_{v_k t} \qquad (16)$$

$$\gamma_{v_k t} = \hat{\gamma}_{v_k t} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta}_{v_k t} \end{bmatrix} \qquad (17)$$

$$\mathbf{b_{a_t}} = \hat{\mathbf{b}}_{\mathbf{a_t}} + \delta\mathbf{b_{a_t}} \qquad (18)$$

$$\mathbf{b}_{\omega_\mathbf{t}} = \hat{\mathbf{b}}_{\omega_\mathbf{t}} + \delta\mathbf{b}_{\omega_\mathbf{t}} \qquad (19)$$

The perturbation and nominal kinematics are separated [7] and the continuous time perturbation kinematics is shown in (20).

$$\begin{bmatrix} \delta\dot{\boldsymbol{\alpha}}_{v_k t} \\ \delta\dot{\boldsymbol{\beta}}_{v_k t} \\ \delta\dot{\gamma}_{v_k t} \\ \delta\dot{\mathbf{b}}_{a_t} \\ \delta\dot{\mathbf{b}}_{\omega_t} \end{bmatrix} = \mathbf{F}_t \begin{bmatrix} \delta\boldsymbol{\alpha}_{v_k t} \\ \delta\boldsymbol{\beta}_{v_k t} \\ \delta\gamma_{v_k t} \\ \delta\mathbf{b}_{a_t} \\ \delta\mathbf{b}_{\omega_t} \end{bmatrix} + \mathbf{G}_t \begin{bmatrix} \mathbf{n}_a \\ \mathbf{n}_\omega \\ \mathbf{n}_{b_a} \\ \mathbf{n}_{b_\omega} \end{bmatrix} \qquad (20)$$

Where $\mathbf{F}_t$ and $\mathbf{G}_t$ are the state matrix and the noise matrix, they shown in (21) and (22).

$$\mathbf{F}_t = \begin{bmatrix} 0 & \mathbf{I} & 0 & 0 & 0 \\ 0 & 0 & -\mathbf{C}_{v_k t}(\hat{\mathbf{a}}_t^{ti} - \mathbf{b}_{a_k})^\times & -\mathbf{C}_{v_k t} & 0 \\ 0 & 0 & -(\hat{\boldsymbol{\omega}}_t^{ti} - \boldsymbol{\omega}_{a_k})^\times & 0 & -\mathbf{I} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (21)$$

$$\mathbf{G}_t = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\mathbf{C}_{v_k t} & 0 & 0 & 0 \\ 0 & -\mathbf{I} & 0 & 0 \\ 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & \mathbf{I} \end{bmatrix} \qquad (22)$$

The continuous time model is integrated to obtain the state transition matrix $\mathbf{\Phi}$, and $\mathbf{\Phi}$ is used to propagate the IMU uncertainties for a given IMU time step δt.

$$\mathbf{\Phi}_t = \mathbf{\Phi}(t, t+\delta t) = \exp(\int_t^{t+\delta t} \mathbf{F}(\tau) d\tau) \qquad (23)$$

$$\mathbf{Q}_t = \int_t^{t+\delta t} \mathbf{\Phi}(t,\tau)\mathbf{GQG\Phi}(t,\tau) d\tau \qquad (24)$$

$$\mathbf{Q} = \begin{bmatrix} \boldsymbol{\sigma}_a^2 & 0 & 0 & 0 \\ 0 & \boldsymbol{\sigma}_\omega^2 & 0 & 0 \\ 0 & 0 & \boldsymbol{\sigma}_{b_a}^2 & 0 \\ 0 & 0 & 0 & \boldsymbol{\sigma}_{b_\omega}^2 \end{bmatrix} \qquad (25)$$

Equation (26) is used to propagate covariance from t to $t+\delta t$. Starting at t_k with $\mathbf{P}_{v_k v_k} = \mathbf{0}$, the covariance is propagated for each IMU measurement between t_k and t_{k+1} to obtain $\mathbf{P}_{v_k v_{k+1}}$. $\mathbf{P}_{v_k v_{k+1}}$ will be used in the optimization.

$$\mathbf{P}_{v_k\, t+\delta t} = \mathbf{\Phi}_t \mathbf{P}_{v_k t} \mathbf{\Phi}_t^T + \mathbf{Q}_t \qquad (26)$$

Equation (27) is used to find the Jacobian of $\mathbf{z}_{v_k v_{k+1}}$ with respect to $\mathbf{z}_{v_k v_k}$. Start with $\mathbf{J}_{v_k v_k} = \mathbf{I}$, it is multiplied with $\mathbf{\Phi}$ recursively between t_k and t_{k+1} to approximate $\mathbf{J}_{v_k v_{k+1}} = \frac{\partial \delta \mathbf{z}_{v_k v_{k+1}}}{\partial \delta \mathbf{z}_{v_k v_k}}$.

$$\mathbf{J}_{t+\delta t} = \mathbf{\Phi}_t \mathbf{J}_t \qquad (27)$$

Using the sub-blocks of $\mathbf{J}_{v_k v_{k+1}}$: $\mathbf{J}_{b_a}^\alpha$, $\mathbf{J}_{b_\omega}^\alpha$, $\mathbf{J}_{b_a}^\beta$, $\mathbf{J}_{b_\omega}^\beta$, $\mathbf{J}_{b_\omega}^\gamma$ as shown in (28), the actual pre-integration values $\boldsymbol{\alpha}_{v_k v_{k+1}}$, $\boldsymbol{\beta}_{v_k v_{k+1}}$, and $\gamma_{v_k v_{k+1}}$ can be updated using (29). The mapping $\exp_q : \mathbf{R}^3 \mapsto S^3$ maps an axis angle vector to a quaternion in 3-Sphere. This differs from the [3] as it uses the small angle quaternion approximation shown in (17). Instead of using approximations, an exact mapping is used here as there is not much increase computational cost in practice. The $\exp_q$ mapping is defined in (30), where ϕ is the magnitude of the rotation and $\mathbf{u}$ is the unit vector indicating the direction of rotation.

$$\mathbf{J}_{v_k v_{k+1}} = \begin{bmatrix} & \mathbf{J}^{\alpha}_{b_a} & \mathbf{J}^{\alpha}_{b_\omega} \\ \cdots_{9\times9} & \mathbf{J}^{\beta}_{b_a} & \mathbf{J}^{\beta}_{b_\omega} \\ & \cdots_{3\times3} & \mathbf{J}^{\gamma}_{b_\omega} \\ \cdots_{3\times9} & \cdots_{3\times3} & \cdots_{3\times3} \end{bmatrix} \tag{28}$$

$$\begin{aligned} \boldsymbol{\alpha}_{v_k v_{k+1}} &= \hat{\boldsymbol{\alpha}}_{v_k t} + \mathbf{J}^{\alpha}_{b_a}\delta\mathbf{b}_{a_t} + \mathbf{J}^{\alpha}_{b_\omega}\delta\mathbf{b}_{\omega_t} \\ \boldsymbol{\beta}_{v_k v_{k+1}} &= \hat{\boldsymbol{\beta}}_{v_k t} + \mathbf{J}^{\beta}_{b_a}\delta\mathbf{b}_{a_t} + \mathbf{J}^{\beta}_{b_\omega}\delta\mathbf{b}_{\omega_t} \\ \boldsymbol{\gamma}_{v_k v_{k+1}} &= \hat{\boldsymbol{\gamma}}_{v_k t} \otimes \exp_q(\mathbf{J}^{\gamma}_{b_\omega}\delta\mathbf{b}_{\omega_t}) \end{aligned} \tag{29}$$

$$\mathbf{q} = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \exp_q(\phi\mathbf{u}) = \begin{bmatrix} \cos\phi \\ \mathbf{u}\sin\phi \end{bmatrix} \tag{30}$$

Finally, it is difficult to obtain the exact solution for the integrals from (8), (9), (10), (23), and (24), thus the discretization must be computed numerically. Many methods such as Euler's, mid-point integration, and Runge-Kutta are available to integrate numerically. The Euler's method is chosen here for simplicity. One of the more common method used is Runge-Kutta 4th order.

For computing the IMU pre-integration values numerically, it start with $\hat{\boldsymbol{\alpha}}_{v_k v_k} = 0$, $\hat{\boldsymbol{\beta}}_{v_k v_k} = 0$ and $\hat{\boldsymbol{\gamma}}_{v_k v_k} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$, and iteratively update the pre-integration with each IMU measurement.

$$\hat{\boldsymbol{\alpha}}_{v_k\, t+\delta t} \approx \hat{\boldsymbol{\alpha}}_{v_k t} + \hat{\boldsymbol{\beta}}_{v_k t}\delta t + \frac{1}{2}\mathbf{C}\{\hat{\boldsymbol{\gamma}}_{v_k t}\}(\hat{\mathbf{a}}^{ti}_t - \mathbf{b}_{a_k})\delta t^2 \tag{31}$$

$$\hat{\boldsymbol{\beta}}_{v_k\, t+\delta t} \approx \hat{\boldsymbol{\beta}}_{v_k t} + \mathbf{C}\{\hat{\boldsymbol{\gamma}}_{v_k t}\}(\hat{\mathbf{a}}^{ti}_t - \mathbf{b}_{a_k})\delta t \tag{32}$$

$$\hat{\boldsymbol{\gamma}}_{v_k\, t+\delta t} \approx \hat{\boldsymbol{\gamma}}_{v_k t} \otimes \exp_q(\hat{\boldsymbol{\omega}}^{ti}_t - \mathbf{b}_{\omega_k}) \tag{33}$$

Assuming constant values for $\mathbf{F}_t$ and $\mathbf{G}_t$ during the period of integration, $\boldsymbol{\Phi}$ is approximated by taking the first two terms of the matrix exponential. $\mathbf{Q}_t$ is approximated using the identity $\boldsymbol{\Phi}(t,t) = \mathbf{I}$.

$$\boldsymbol{\Phi}_t \approx \exp(\int_t^{t+\delta t} \mathbf{F}(t)d\tau) \approx \mathbf{I} + \mathbf{F}_t\delta t \tag{34}$$

$$\mathbf{Q}_t \approx \int_t^{t+\delta t} \boldsymbol{\Phi}(t,t)\mathbf{GQG}\boldsymbol{\Phi}(t,t)d\tau \approx \mathbf{GQG}\delta t \tag{35}$$

Later, the pre-integrations will be used as measurements during optimization.

III. Initialization

For initialization, we assume that the vehicle is stationary for a period of time as in [2], which simplifies the initialization problem. Since the scale is directly observable from a stereo camera, it does not need to be recovered in the initialization stage. We follow [3] and ignore accelerometer bias terms in the initial step, because the accelerometer bias is coupled with the gravity vector, and due to the large magnitude of the gravity vector, the accelerometer bias is difficult to determine [3]

Regarding the gyroscope bias, because the vehicle is assumed to be stationary, the bias can be calculated by simply taking the average of all the gyroscope data during initialization as follows for N number of IMU measurements when the vehicle is stationary.

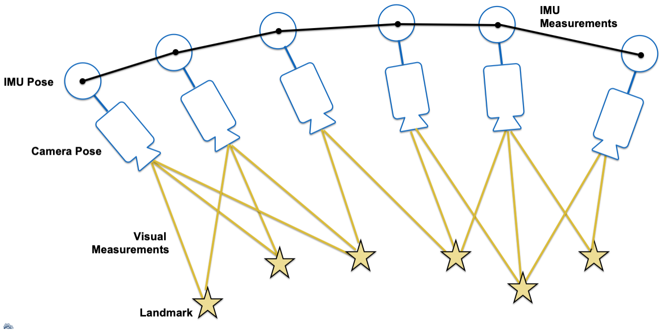

Fig. 1. Visual Inertial Bundle Adjustment

$$\mathbf{b}_{w_0} = \frac{1}{N}\sum_{k=1}^{N}\boldsymbol{\omega}^{v_k i}_{v_k} \tag{36}$$

The initial gravity vector $\mathbf{g}$ is also determined by taking the average of the accelerometer reading during initialization.

$$\mathbf{g}_0 = \frac{1}{N}\sum_{k=1}^{N}\hat{\mathbf{a}}^{v_k i}_{v_k} \tag{37}$$

After obtaining the initial gravity vector, we determine the rotation $\mathbf{C}_{iv_0}$ between the inertial frame and the current vehicle frame v_0 by rotating the gravity to the z-axis [3].

IV. Back-end Optimization

After completing the initialization step, we perform a sliding window-based tightly-coupled Stereo Visual Inertial Odometry Optimization to solve the camera pose in the world frame.

A. Formulation

We follow the state formulation from [3] except for each landmark the states are the x, y, and z coordinates in the world frame rather than the inverse depth formulation used in [3].

$$\text{vehicle state :} \quad \mathbf{x}_k = [\mathbf{p}_i^{v_k i}, \mathbf{v}_i^{v_k i}, \mathbf{q}_i^{v_k i}, \mathbf{b}_{a_k}, \mathbf{b}_{\omega_k}] \tag{38}$$

$$\text{full state :} \quad \boldsymbol{\chi} = [\underbrace{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_m,}_{\text{vehicle}} \underbrace{\mathbf{p}_i^{p_1 i}, \mathbf{p}_i^{p_2 i}, \ldots, \mathbf{p}_i^{p_n i}}_{\text{landmark}}] \tag{39}$$

where $\mathbf{x}_k$ is the IMU state at the time k, which includes translation, velocity, and orientation of the IMU in the world frame, and acceleration and gyroscope bias. n represents the number of frames in the sliding window, and m is the total number of landmarks in the sliding window.

We form this problem into a visual inertial bundle adjustment as shown in Figure 1. At each frame, the IMU pose and the camera pose is connected by calibration. The visual measurements provide constraints between the camera poses and the landmarks. The IMU measurements connect IMU poses from frame to frame.

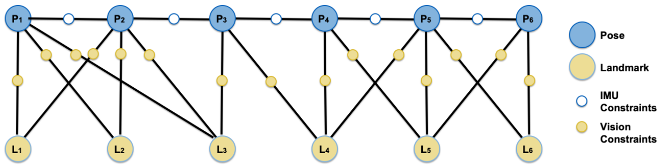

Fig. 2. Factor graph

The graph in Figure 1 can be represented using a factor graph which is shown in Figure 2. The factor graph has two types of variables, which are THE pose of the IMU and the 3D position of the landmarks. There are also two factors which are the IMU measurement constraints and the visual measurements constraints. The objective of the factor graph is to minimize the sum of all measurement residuals which include IMU measurement residuals and visual measurement residuals obtain a maximum posterior estimation:

$$\min_{\chi}\Big\{\sum_{k}\rho(\mathbf{e}_{s,k}\mathbf{Q_k}^{-1}\mathbf{e}_{s,k}^T)+ \sum_{j,k}\rho(\mathbf{e}_{l,jk}\mathbf{R}_{jk}^{-1}\mathbf{e}_{l,jk}^T)\Big\} \tag{40}$$

where the Huber norm is defined as follows. A robust loss function is used to minimize the effects of outliers.

$$\rho(s)=\begin{cases}1 & s\geq a\\ 2\sqrt{s}-1 & s<a\end{cases} \tag{41}$$

$\mathbf{e}_{s,k}$, and $\mathbf{e}_{l,jk}$ are the residuals for IMU measurements and visual measurements respectively, and they are defined in section IV-B and section IV-C. The entire factor graph optimization is implemented in C++ using g2o library.

B. IMU Measurement Residual

The IMU residuals are obtained by rearranging (11), (12), and (13). Here we used rotation matrix representation to make Jacobian derivation easier. For rotation, the optimization is performed on the perturbation on SO(3), use quaternion or rotation matrix to derive the Jacobian will lead to the same result under the same assumptions. In the Jacobian derivation of [3], it uses small quaternion approximations as well as the assumption that the error is small. We adopted the Jacobian derivation of [6], which is formulated using rotation matrices and it does not make the small error assumption. The increase in computational cost should be minimal and more accurate Jacobian will lead to better convergence.

$$\mathbf{e}_{s,k}(\hat{\mathbf{z}}_{v_k v_{k+1}},\mathbf{x}_k,\mathbf{x}_{k+1}) = \begin{bmatrix}\mathbf{C}_{v_k i}(\mathbf{p}_i^{v_{k+1}i}-\mathbf{p}_i^{v_k i}-\mathbf{v}_i^{v_k i}\Delta t_k+\frac{1}{2}\mathbf{g}_i\Delta t_k^2)-\hat{\boldsymbol{\alpha}}_{v_k v_{k+1}}\\ \mathbf{C}_{v_k i}(\mathbf{v}_i^{v_{k+1}i}-\mathbf{v}_i^{v_k i}+\mathbf{g}_i\Delta t_k)-\hat{\boldsymbol{\beta}}_{v_k v_{k+1}}\\ \ln(\mathbf{C}^T\{\hat{\boldsymbol{\gamma}}_{v_k v_{k+1}}\}\mathbf{R}_{iv_k}^T\mathbf{R}_{iv_{k+1}})^{\vee}\\ \mathbf{b}_{a_{k+1}}-\mathbf{b}_{a_k}\\ \mathbf{b}_{\omega_{k+1}}-\mathbf{b}_{\omega_k}\end{bmatrix} \tag{42}$$

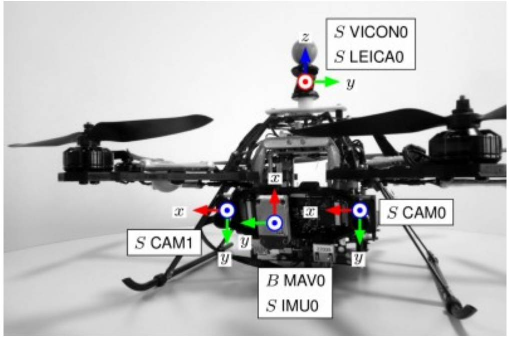

Fig. 3. UAV Configuration

C. Visual Measurement Residual

A traditional pinhole camera model is assumed, which defines the reprojection errors on a generalized image plane. The visual residuals are shown as follow:

$$\mathbf{e}_{l,jk}(\hat{\mathbf{z}}_{c_k l},\mathbf{x}_k,\mathbf{p}_i^{p_j i}) = \mathbf{y}_k^{p_j}-\pi(\mathbf{C}_{cv}(\mathbf{C}_{iv_k}^T(\mathbf{p}_i^{p_j i}-\mathbf{r}_i^{v_k i})-\boldsymbol{\rho}_v^{cv})) \tag{43}$$

where $\mathbf{y}_k^{p_j}$ is the observation of the p_j landmark in the kth image, which includes the x and y coordinate of the left image and x coordinate of the right image. π is the camera projection model which projects the landmark from the camera frame to the image plane.

D. Sliding Window without Marginalization

In order to reduce the computational complexity of the optimization, we apply a sliding window approach. However, we do not perform a proper marginalization to propagate the information from the marginalized out pose. Whenever a new pose is added, the oldest pose will be discarded. We keep a window size to be 25.

V. Experimental Results

We performed our experiments on the well-known EuRoC dataset. The EuRoC dataset is collected on a quadrotor indoors equipped with a Visual Inertial Sensor [9], the UAV platform is shown in Figure 3. The Visual Inertial Sensor module provides hardware synchronization between the two cameras and the IMU, this ensures all measurements are collected at the same instance in time. The stereo cameras images arrive at 20Hz, and the IMU measurements arrive at 200Hz. Due to the IMU measurements arriving at a much higher rate than stereo images, IMU pre-integration is critical to be computationally efficient. Ground truth is also provided, it is either collected by a VICON or Leica system depending on the sequence. However, the ground truth provided is not in the same frame as the estimation. Thus, we need to align the estimated poses with the ground truth, this is done in the same fashion as [4]. The EuRoC dataset is challenging due to rapid motions and illumination changes throughout the sequences. To demonstrate the effect of combining vision with IMU, we compared our results to ProSLAM which are vision only. The results are summarized in table I.

TABLE I
EXPERIMENTAL RESULTS RMS ERRORS

Sequences	ProSLAM	Ours
MH_01_easy	0.07986	**0.07247**
MH_02_easy	0.06326	**0.06125**
MH_03_medium	0.4279	**0.3434**
V1_01_easy	0.1202	**0.1018**
V1_02_medium	0.2333	**0.1798**

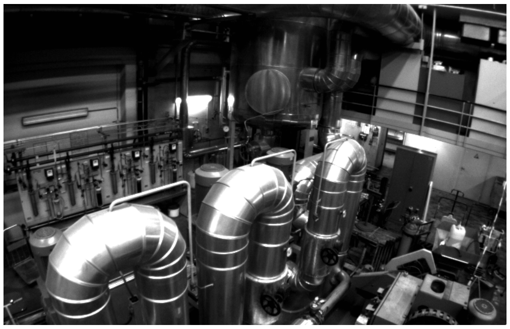

Fig. 4. Sample frame from MH_01_easy

There are a total of 10 sequences available in EuRoC, we weren't able to finish the unlisted sequences. ProSLAM, with loop closing disabled, uses only relative transformation between to images, and compound these relative transforms to estimate the trajectory. The front-end of ProSLAM is designed for this relative transformation estimation. It extracts as many features as possible between two frames, and coarsely match the features without performing outlier rejection with methods such as RANSAC. As a result, the features matched over a few frames are not stable, there are many mismatches and outliers. Although this front-end has proven to work well for ProSLAM, our experience suggests visual inertial fusion needs high-quality stable features over multiple frames. ProSLAM's front-end was not robust enough for the more difficult sequences.

For the EuRoC sequence we were able to successfully finish, there is a slight improvement with the easy sequence such as MH_01_easy, MH_02_easy, and V1_01_easy. In these sequences, the quadrotor was moving at a slow to moderate velocity, and the scenes do not have too much illumination change. The Machine Hall (MH) sequences are particularly feature-rich. A sample from MH_01_easy is shown in Figure 4. This shows in environments with good texture and illumination without rapid motion, visual inertial fusion does not offer much of an advantage. The trajectory plots for MH_01_easy shown in Figure 5. It is clear that the estimated trajectory follows the groundtruth very well.

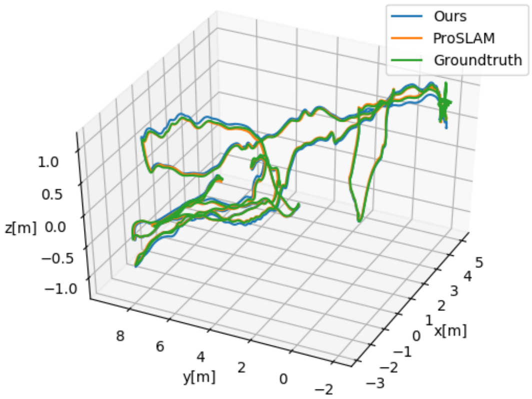

Fig. 5. MH_01_easy trajectories comparison

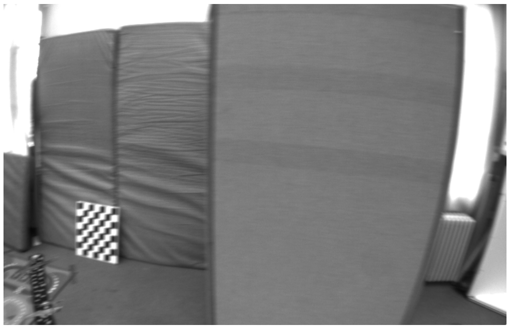

Fig. 6. Sample frame from V1_02_medium

The more difficult trajectories such as MH_03_medium and V1_02_medium, our result is significantly better. The quadrotor in these sequences moved much more aggressively in comparison to the easy sequences. Rapid motion creates would create less accurate feature matches due to motion blur. Figure 6 shows a frame from V1_02_medium without good features and has a good amount of motion blur. The addition IMU motion constraint proved valuable, as inaccurate matches have less of an effect on the estimate obtained from optimization. This is apparent when looking at trajectory plots of V1_02_medium shown in Figure 8 and 7. In Figure 7, the ProSLAM trajectory is not smooth due to the inaccurate feature matching. Comparing this to Figure 8, where we can see the trajectory is much more smooth. This suggests visual inertial fusion increases accuracy in situations with rapid motion and degraded feature matching accuracy. We also spent minimal effort tuning the IMU parameters, the results could be better with additional parameter tuning.

VI. CONCLUSION AND FUTURE WORK

In this report, we present a visual inertial odometry that is based on [4] and follows [3] IMU pre-integration method. We evaluate our algorithm in EuRoc dataset and demonstrate that our approach outperforms the original ProSLAM, especially when the features in the images are not rich or the motion of the UAV is aggressive.

There are a number of possible improvements that can be implemented in the future. We could add keyframe selection to further reduce the computational complexity of the optimization. For the sliding window, a proper marginalization can

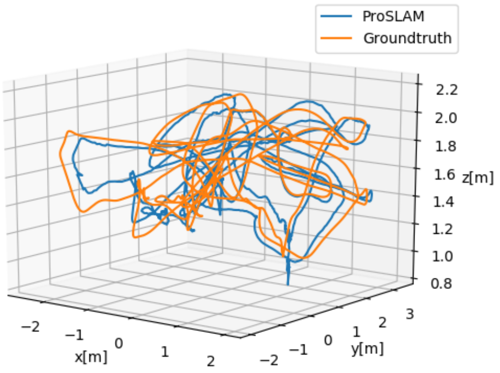

Fig. 7. V1_02_Medium ProSLAM result compared to groundtruth

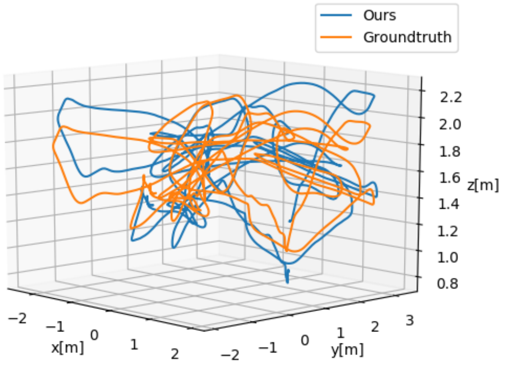

Fig. 8. V1_02_Medium our result compared to groundtruth

be applied to propagate the information from the discarded poses and features. A loop closure module can be added to detect whether the UAV has visited the same location before and further reduce the drift of the poses over time. In addition, the current front-end of the odometry is not robust enough to work on some of the difficult sequences in EuRoc dataset. As a result, improving the robustness of the visual front-end is also necessary. Furthermore, tuning parameters is critical to get VIO working, this process is tedious, and these parameters must be manually tuned in a careful way for different environments. An additional piece of future work could be applying learning methods to help in implementing VIO.

REFERENCES

[1] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, “Keyframe-based visualinertial odometry using nonlinear optimization,” *The International Journal of Robotics Research,* 34(3), 314334.

[2] K. Sun et al., “Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight,” *IEEE Robotics and Automation Letters,* vol. 3, no. 2, pp. 965-972, April 2018.

[3] T. Qin, P. Li and S. Shen, “VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator,” *IEEE Transactions on Robotics,* vol. 34, no. 4, pp. 1004-1020, Aug. 2018. doi: 10.1109/TRO.2018.2853729

[4] D. Schlegel, et al. “ProSLAM: Graph SLAM from a Programmer’s Perspective,” *2018 IEEE International Conference on Robotics and Automation (ICRA),* (2018): 1-9.

[5] T.D Barfoot, “State Estimation for Aerospace Vehicles,” Cambridge, U.K.: Cambridge Univ. Press, 2015.

[6] C. Forster, L. Carlone, F. Dellaert and D. Scaramuzza, “On-Manifold Preintegration for Real-Time Visual–Inertial Odometry,” in IEEE Transactions on Robotics, vol. 33, no. 1, pp. 1-21, Feb. 2017.

[7] J. Sol, “Quaternion kinematics for the error-state Kalman filter,” CoRRabs/1711.02508 (2017): n. pag.

[8] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, “Keyframe-based visual-inertial odometry using nonlinear optimization,” Int. J. Robot. Research, vol. 34, no. 3, pp. 314334, Mar. 2014.

[9] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale and R. Siegwart, A Synchronized Visual-Inertial Sensor System with FPGA Pre-Processing for Accurate Real-Time SLAM in IEEE International Conference on Robotics and Automation (ICRA), 2014