Mobile Robotics: Visual Inertial Navigation with Invariant EKF and Self-Supervised Visual Features

Jing-An Tzeng
University of Michigan
EECS
Ann Arbor, Michigan
robintzg@umich.edu

Joseph Lowman
University of Michigan
Robotics
Ann Arbor, Michigan
lowmanj@umich.edu

Nalin Bendapudi
University of Michigan
Robotics
Ann Arbor, Michigan
bnalin@umich.edu

Hsuan-Cheng Chen
University of Michigan
Robotics
Ann Arbor, Michigan
hsuanc@umich.edu

***Abstract*—This project aims to adapt an open source visual inertial navigation framework with two modifications. First, an alternative visual descriptor and extractor, SuperPoint, is tested in place of ORB features. Secondly, an invariant error state is implemented in order to more effectively track the IMU pose uncertainty while maintaining the visual feature correction step in a "dual track" system. Each of these changes is evaluated separately on three datasets obtained from the EuRoC MAV dataset. Our code repository is located here. See YouTube video for presentation here.**

***Index Terms*—VINS, inertial navigation, Invariant EKF, Kalman Filtering**

I. Introduction

Visual inertial navigation systems (VINS) attempt to localize a robot's motion using an inertial sensor such as an IMU and cameras. This is especially useful in GPS denied environments. Visual inertial navigation is a filtering based approach, employing a prediction step and a correction step. Prediction is accomplished with an IMU dynamics model, and visual features are used for the correction step.

The OpenVINS library is an implementation of a multi-state constraint Kalman filter (MSCKF) for use in research of VINS systems. [1]. Similar to typical SLAM system, the OpenVINS also consisted of front-end and back-end. In this project, we try to improve both the front-end and back-end system.

For front-end, the inputs to most real-world computer vision systems are raw images, not idealized point locations. Extracting interest points from images is one of the very first steps in geometric computer vision tasks such as Simultaneous Localization and Mapping (SLAM), camera calibration, and image matching. Interest points are locations in an image which are repeatable from different viewpoints and lighting conditions. The subfield of mathematics and computer vision known as Multiple View Geometry [2] consists of theorems and algorithms built on the assumption that interest points can be reliably extracted and matched across images. Image matching from feature use handcrafted features such as SIFT [3], SURF [4], and ORB [5] are still considered state-of-art techniques. These techniques do not take advantage of the deep learning techniques that have been exploited for many standard perception tasks. In the original VINS implementation, it used ORB features to mark interest points in real images. ORB features are considered one of the most prominent tracking features in frames, and the features estimate the position and orientation of the camera. The quality of feature is a key factor in the performance of visual odometry [6]. Robust and correct features must be used for optimal performance. Instead of using ORB features, we decided to use SuperPoint [7] to define interest points in real images. SuperPoint is a self-supervised learning technology using self-training, which means it can create a dataset of pseudo-ground truth interest point locations in real images without human annotation.

Instead of large-scale human annotation effort to define interest points in real images, Superpoint has two main advantage. First, it is possible to transfer knowledge from a synthetic dataset into real-world images. Second, sparse interest point detection and description can be cast as a single, efficient convolutional neural network in a real time system, see fig. I.

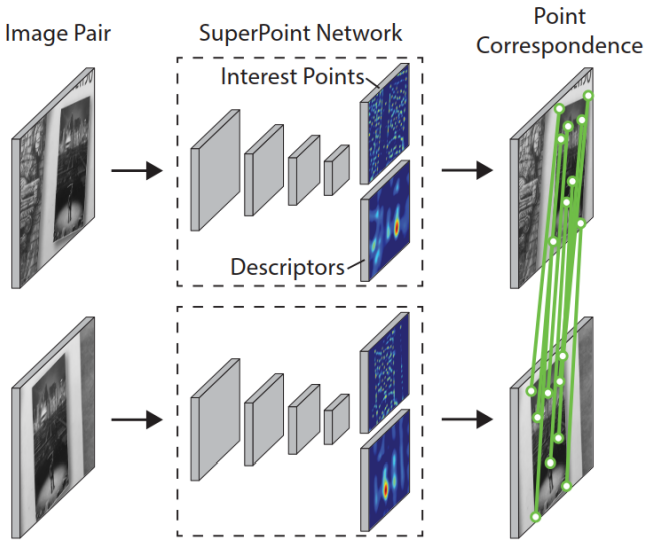


Fig. 1. SuperPoint for Geometric Correspondences. The CNN computes ORB-like 2D interest point locations and descriptors in a single forward pass. This picture is borrowed from [7].

For the back end, the Multi-State Constraint Kalman Filter

(MSCKF) implemented in the OpenVINS library is able to track visual features efficiently and achieve real-time operation [1]. In contrast to a typical EKF, which jointly estimate all visual landmarks, MSCKF augments the state of the system with camera poses. Once a visual feature has been observed by two or more camera poses, it can be triangulated. Also, the MSCKF expresses constraints between multiple camera pose to achieve higher estimation accuracy.

To summarize our work in the paper, we make three main contributions:

- We substitute Superpoint for ORB feature extractor and improve the performance for the whole system.
- We attempt to develop and combine the MSCKF and InEKF.
- We ran experiments and evaluated aspects of the Superpoint in order to prove that it can be a real-time system.

This paper is organized as follows. In Section 2, we review related work on the visual SLAM and VINS system. We then present our method in Section 3 and the experiments in Section 4. Finally, Section 5 presents the discussion and in Section 6, we make our conclusion.

II. Related Work

A key difference between VINS and SLAM is that VINS does not attempt to build and maintain a map of the environment, whereas SLAM does. IMU measurements and monocular camera frames can also be used in the ORB-SLAM framework [8] in order to solve the SLAM problem and incorporate loop closures. Other proposed methods [9] applied the right invariant EKF to the visual inertial navigation problem (VINS). The authors proposed a method for applying the right invariant EKF to the multi-state constraint Kalman filter (MSCKF). This method was particularly exciting and relevant. The paper proved that the conventional EKF-VINS does not have expected invariance property and the output of RIEKF-VINS is invariant under any 'deterministic transformation'. The authors also note a drawback of the RIEKF-VINS technique, which is the cost of maintaining and propagating covariance matrices for a number of landmarks. So the authors introduce an algorithm which exploits the consistency of RI-EKF and also the linear time complexity of MSCKF. We found it extremely difficult to replicate this algorithm directly, as the notation was difficult to follow and many derivations were omitted from the paper.

In [10], the authors proposed an unscented Kalman filtering algorithm using the $SE_{2+p}(3)$ Lie group embedding. Additionally, they used IMU data fused with monocular vision data. The experimental results demonstrated that invariant Kalman filtering methods were significantly more accurate in estimating state than the traditional unscented Kalman filter. They reported only a minor improvement using the right invariant, unscented Kalman filter versus the right invariant, extended Kalman filter. However, the unscented filter does not require computation of Jacobians. The authors additionally observed that the invariant UKF took much more computation time than the right invariant EKF.

Another paper [11] derived formulations for an error-state Kalman filter suited for real applications using integration of signals from an IMU. However, this paper didn't use the invariant EKF, which is a special case of error-state EKF.

Some researchers have proposed methods for using stereo cameras in visual inertial odometry to improve state estimation [12]. However, this method used MSCKF with the linearized system in an extended Kalman filter. Presumably, the method could be improved by using the Lie group embedding of the state.

In the past, image matching from feature use handcrafted features such as SIFT [3], SURF [4], and ORB [5] and are still considered state-of-art techniques. These techniques do not take advantage of the deep learning techniques that have been exploited for many standard perception tasks. In this paper, we want to replace ORB feature with SuperPoint to analyze the performance of learning-based features.

III. Method

A. SuperPoint

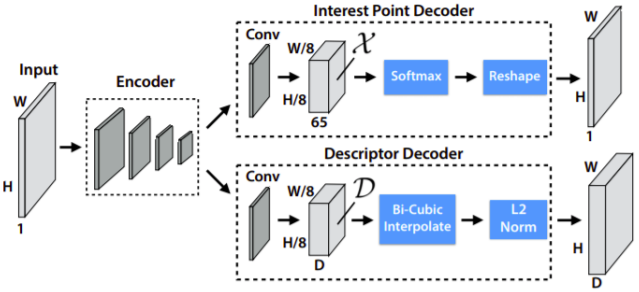


Fig. 2. The network architecture of SuperPoint. This picture is borrowed from [7].

SuperPoint is a fully-convolutional neural network architecture with a VGG-like encoder. The encoder consists of VGG blocks, spatial down-sampling via pooling and ReLU to reduce the dimensionality of the image. For the decoder, there are two kinds of decoders, the interest point decoder and the descriptor decoder. The output of the interest point decoder is a $H \times W \times 1$ output, with each pixel of the output corresponding to a probability of interest point for that pixel in the input. The output of the descriptor network is a tensor with size $H \times W \times D$ and each channel D is normalized to unit length and can be further used to conduct features matching problem. For our project, we use the pretrained model from here.

B. Multi-State Constraint Kalman Filter and InEKF Dual-track system

We introduce a "Dual-track" system which attempts to incorporate the benefits of invariant Kalman filtering, namely accurate uncertainty estimation, with the visual feature correction ability of the multi-state constraint Kalman Filter.

1) Multi-State Constraint Kalman Filter (MSCKF): The multi-state constraint Kalman Filter provides a measurement model for constraining features observed from multiple camera poses. Additionally, it has the advantage of only storing camera poses in the state, rather than 3D feature positions. Therefore, the complexity of the filter only grows linearly with respect to number of features.

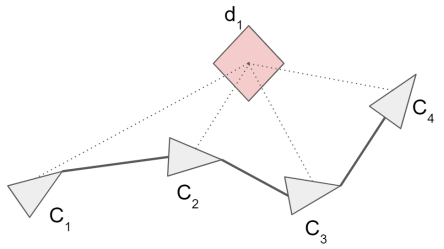


Fig. 3. MSCKF: Landmarks d_i are observed from a set of camera poses S_j. The set of camera poses are used to triangulate an estimate of the landmark's global position. Here, C_k is a camera pose in SE(3). A subset of camera poses are stored in the state, rather than feature positions directly.

a) Prediction: The state at time t is represented as X_t. It includes rotation of the IMU R_t, IMU velocity v_t, IMU position p_t, acceleration bias b_a, gyroscope bias b_g, and N camera poses. Each camera pose is composed of a quaternion rotation q_t^i and a position c_t^i.

$$X_t : (15 + 6N)$$

$$X_t = \begin{bmatrix} \mathbf{R_t} \\ \mathbf{v_t} \\ \mathbf{p_t} \\ \mathbf{b_a} \\ \mathbf{b_g} \\ \mathbf{q_t} \\ \mathbf{c_t} \end{bmatrix}$$

The error state is defined using the error quaternion for rotations $\delta q \approx \begin{bmatrix} \frac{1}{2}\delta\theta^T & 1 \end{bmatrix}^T$. All other errors are defined as the difference between the true value and the estimated value.

The error dynamics are described by

$$\dot{\tilde{X}} = F\tilde{X} + Gn$$

where F and G are the Jacobians computed in [13] and n is the noise vector.

During propagation, the IMU mean state estimate is propagated using the discrete time IMU nonlinear dynamics.

The covariance of the state is propagated by

$$\dot{P} = FP + PF^T + GQ_{IMU}G^T$$

We view the covariance matrix as a block matrix, composed of the covariance of the IMU P_I, the covariance of the camera poses P_C, and the cross-covariances between them P_{CI} and P_{IC}.

$$P = \begin{bmatrix} P_{II} & P_{IC} \\ P_{CI} & P_{CC} \end{bmatrix}$$

b) Correction: Assume a set S_j of the most recent camera poses in the state at time j and a 2 by 1 image noise vector $\mathbf{n}_i^{(j)}$.

The observation is obtained from the camera frame position as

$$\mathbf{z}_i^{(j)} = \frac{1}{{}^{C_i}Z_j} \begin{bmatrix} {}^{C_i}X_j \\ {}^{C_i}Y_j \end{bmatrix} + \mathbf{n}_i^{(j)}, \quad i \in \mathcal{S}_j$$

Equations (18)-(26) of [13] express the residual computed between the feature observation in pixel coordinates and the estimated global position.

2) Invariant Extended Kalman filter (InEKF):

a) Prediction: In conventional error-state EKF, propagation and correction of the error state depend on Jacobian linearizations. The Jacobians depend on the error-state. An advantage of the invariant Kalman filter is the definition of the error state. It is defined such that the Jacobians do not depend on the error state and therefore the Jacobians stay constant.

In the invariant filter, we maintain a separate state. Here, we keep $\frac{1}{5}$ of the visual features in order to limit computation time. Each feature is represented by its global position as a landmark $d_t^i \in \mathbb{R}^3$.

$$X_t : (15 + 3N) \times (15 + 3N)$$

$$X_t = \begin{bmatrix} R_t & v_t & p_t & b_a & b_g & d_t \\ 0_{1x3} & 1 & 0 & 0 & 0 & 0 \\ 0_{1x3} & 0 & 1 & 0 & 0 & 0 \\ 0_{1x3} & 0 & 0 & 1 & 0 & 0 \\ 0_{1x3} & 0 & 0 & 0 & 1 & 0 \\ 0_{1x3} & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

b) Error Dynamics: The error dynamics can be defined as

$$\frac{d}{dt}(\bar{R}_t R_t^T) \approx (\bar{R}_t(w_t - b_g))_\times \frac{d}{dt}(\bar{v}_t - \bar{R}_t R_t^T v_t) \approx$$

$$(\bar{g}_t)_\times \xi_t^R + (\bar{v}_t)_\times \bar{R}_t(w_t - b_g) + \bar{R}_t(w_t - b_a)$$

$$\frac{d}{dt}(\bar{p}_t - \bar{R}_t R_t^T p_t) \approx \xi_t^v + (\bar{p}_t)_\times \bar{R}_t(w_t - b_g)$$

$$\frac{d}{dt}(\bar{d}_t - \bar{R}_t R_t^T d_t) \approx (\bar{d}_t)_\times \bar{R}_t(w_t - b_g) + \bar{R}_t h(z) w_t$$

c) Prediction Step: The IMU signal is used as an input $u = \begin{bmatrix} \omega & a \end{bmatrix}^T$. The standard IMU discrete time dynamics are used for IMU state propagation (see [14]). Denote the predicted state as $\overline{\mathbf{X}}_t$.

Uncertainty is propagated using the formula $FP^F + P^F F^T + Q$. Here, F represents the dynamics of the uncertainty, while Q represents the uncertainty in measurements. In the case of a standard EKF, F is a Jacobian that depends on the state estimate. Therefore, uncertainty can diverge when the state estimate is poor. In the invariant case, the mapping of the error in the Lie algebra to the Lie group is exact and independent of the state estimate. Let A_t represent the uncertainty dynamics in the Lie algebra. Then F is the mapping of the change in uncertainty to the Lie group, using the exponential map. Let the operator $[\ ]_\times$ denote the skew symmetric matrix form of a vector.

The matrix A_t, considering only the IMU and IMU biases, is constant and is computed as

$$A = \begin{bmatrix} \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} \\ [g]_\times & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & \mathbf{I}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} \end{bmatrix}$$

The adjoint is obtained as

$$\mathrm{Ad}_{\mathbf{X}_t} = \begin{bmatrix} \mathbf{R} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ (\mathbf{v}_t)_\times \mathbf{R}_t & \mathbf{R}_t & \mathbf{0} & \mathbf{0} \\ (\mathbf{p}_t)_\times \mathbf{R}_t & \mathbf{0} & \mathbf{R}_t & \mathbf{0} \\ (\mathbf{d}_t)_\times \mathbf{R}_t & \mathbf{0} & \mathbf{0} & \mathbf{R}_t \end{bmatrix}$$

The discrete time form of the right invariant covariance update equations are as follows:

$$\Phi = exp_m(A\Delta t) \approx I + A\Delta t \tag{1}$$

$$\hat{Q}_k = Ad_{X_t} Q_k Ad_{X_t}^T \Delta t \tag{2}$$

$$\hat{P}_t^F = \Phi P_{t-1}^F \Phi^T + \hat{Q}_k \tag{3}$$

The covariance transition matrix for the IMU is computed as

$$\Phi = \begin{bmatrix} I & 0 & 0 & 0 & -R\Delta t & 0 \\ [g]_\times \Delta t & I & 0 & 0 & -R[v]_\times \Delta t & -R\Delta t \\ 0 & I\Delta t & I & 0 & -R[p]_\times \Delta t & 0 \\ 0 & 0 & 0 & I & -R[d]_\times \Delta t & 0 \\ 0 & 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{bmatrix}$$

d) Correction Step: In the correction step, we rely on the OpenVINS MSCKF triangulation formulation, using Gauss Newton optimization, to approximate 3D positions for each feature $\begin{bmatrix} x_{tri} & y_{tri} & z_{tri} \end{bmatrix}^T$. Then, we perform landmark corrections using the right invariant observation model.

The right invariant observation model is formulated as in [14].

$$Y_t = (X_t)^{-1} b + V_t$$

$$Y_t = \begin{bmatrix} x_{tri} \\ y_{tri} \\ z_{tri} \\ 0 \\ 1 \\ -1 \end{bmatrix}$$

$$(X_t)^{-1} = \begin{bmatrix} R_t & -R_t^T & -R_t^T p_t & -R_t^T q_t \\ \mathbf{0}_{3x3} & 1 & 0 & 0 \\ \mathbf{0}_{3x3} & 0 & 1 & 0 \\ \mathbf{0}_{3x3} & 0 & 0 & 1 \end{bmatrix}$$

$$b_t = \begin{bmatrix} 0_{3x1} \\ 1 \\ -1 \end{bmatrix}, V_t = \begin{bmatrix} w \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The correction is done with the right invariant form and auxiliary selection matrix $\mathbf{\Pi} \triangleq \begin{bmatrix} \mathbf{I} & \mathbf{0}_{3,3} \end{bmatrix}$. This formulation was presented in [14] and is repeated here for clarity.

$$\mathbf{S}_t = \mathbf{H}_t \mathbf{P}_t \mathbf{H}_t^\top + \overline{\mathbf{N}}_t, \quad \mathbf{K}_t = \mathbf{P}_t \mathbf{H}_t^\top \mathbf{S}_t^{-1}$$

$$\mathbf{H}_t = \begin{bmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & -\mathbf{I} & \mathbf{I} \end{bmatrix}$$

$$\mathbf{N}_t = \overline{\mathbf{R}}_t \mathbf{J}_p \left(\tilde{\boldsymbol{\alpha}}_t\right) \mathrm{Cov}\left(\mathbf{w}_t^\alpha\right) \mathbf{J}_p^\top \left(\tilde{\boldsymbol{\alpha}}_t\right) \overline{\mathbf{R}}_t^\top$$

$$\overline{\mathbf{X}}_t^+ = \exp\left(\mathbf{K}_t \mathbf{\Pi} \overline{\mathbf{X}}_t \mathbf{Y}_t\right)$$

$$\overline{\mathbf{X}}_t \mathbf{P}_t^+ = \left(\mathbf{I} - \mathbf{K}_t \mathbf{H}_t\right) \mathbf{P}_t \left(\mathbf{I} - \mathbf{K}_t \mathbf{H}_t\right)^\top + \mathbf{K}_t \overline{\mathbf{N}}_t \mathbf{K}_t^\top$$

3) Invariant EKF With Muti-State Constraint: We tried to combine the merits from both MSCKF and invariant EKF, and we attempted to develop a RI-MSCKF by ourselves.

Our method for the dual-track system is shown in figure 4. The invariant EKF is used for propagation of the IMU state covariance, and the MSCKF is used for propagation of the camera pose covariance. In the correction step, the MSCKF is used to triangulate 3D position estimates for each feature. Then the invariant EKF landmark correction step attempts to correct the IMU state using the global landmark position estimates.

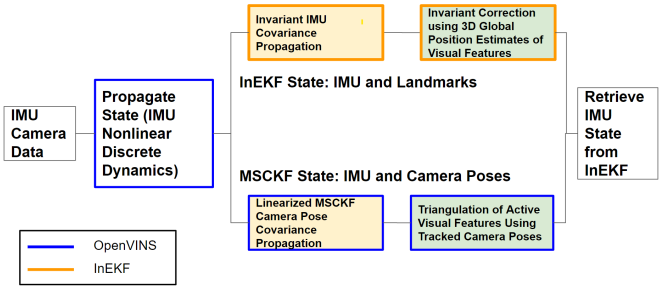


Fig. 4. The combined pipeline for MSCKF and InEKF. MSCKF blocks are outlined in blue and InEKF blocks are outlined in orange. Corresponding class methods are shown in appendix figure 7.

Additionally, we attempted to derive the RI-MSCKF from scratch.

a) State:

$$X_t : (9 + 6N) \times (9 + 6N)$$

$$X_t = \begin{bmatrix} R_c & v_c & p_c & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & R_1 & p_1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where R_1, p_1 is the first camera pose. The process and measurement model for each camera are identical, therefore, without loss of generality, we will derive all further equations assuming only one camera pose constraint.

b) Propagation: We define the deterministic system dynamics

$$f_{u_t}(\cdot) = \begin{bmatrix} \mathbf{R}_t\left(\tilde{\omega}_t\right)_\times & \mathbf{R}_t\tilde{\mathbf{a}}_t + \mathbf{g} & \mathbf{v}_t & \mathbf{0}_{3,1} & \mathbf{0}_{3,1} \\ \mathbf{0}_{1,3} & 0 & 0 & 0 & 0 \\ \mathbf{0}_{1,3} & 0 & 0 & 0 & 0 \\ \mathbf{0}_{1,3} & 0 & 0 & 0 & 0 \end{bmatrix}$$

It can be shown to satisfy the group affine property.

$$f_{u_t}\left(\boldsymbol{X}_1\boldsymbol{X}_2\right) = f_{u_t}\left(\boldsymbol{X}_1\right)\boldsymbol{X}_2 + \boldsymbol{X}_1 f_{u_t}\left(\boldsymbol{X}_2\right) - \boldsymbol{X}_1 f_{u_t}\left(\boldsymbol{I}_d\right)\boldsymbol{X}_2$$

Thus, following the autonomous error dynamics theorem, the left and right invariant error will be independent of the system's state.

Also, by [14], we know that the invariant error ξ satisfies the linear system.

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\xi}_t &= \mathbf{A}_t\boldsymbol{\xi}_t + \hat{\mathbf{w}}_t = \mathbf{A}_t\boldsymbol{\xi}_t + \mathrm{Ad}_{\hat{\mathbf{X}}_t}\mathbf{w}_t \\ \boldsymbol{\eta}_t^r &= \exp\left(\boldsymbol{\xi}_t\right) \end{aligned}$$

where A_t can be computed by linearizing the invariant invariant error dynamics using the first-ordered Taylor expansion.

$$\boldsymbol{\eta}_t^r = \exp\left(\boldsymbol{\xi}_t\right) \approx \mathbf{I}_d + \mathcal{L}_{\mathfrak{g}}\left(\boldsymbol{\xi}_t\right) \text{ to yield}$$

$$g_{u_t}\left(\mathbf{I}_d + \mathcal{L}_{\mathfrak{g}}(\boldsymbol{\xi})\right) =$$

$$\begin{bmatrix} \left(\mathbf{I} + \left(\boldsymbol{\xi}_t^R\right)_\times\right)\left(\tilde{\omega}_t\right)_\times & \left(\mathbf{I} + \left(\boldsymbol{\xi}_t^R\right)_\times\right)\tilde{\mathbf{a}}_t + \mathbf{g} & \boldsymbol{\xi}_t^v & \mathbf{0_{2,3}} \\ \mathbf{0}_{1,3} & 0 & 0 & 0_{2,3} \\ \mathbf{0}_{1,3} & 0 & 0 & 0_{2,3} \\ \mathbf{0}_{1,3} & 0 & 0 & 0_{2,3} \end{bmatrix}$$

$$- \begin{bmatrix} \mathbf{I} + \left(\boldsymbol{\xi}_t^R\right)_\times & \boldsymbol{\xi}_t^v & \boldsymbol{\xi}_t^p & 0 & 0 \\ \mathbf{0}_{3,1} & 1 & 0 & 0 & 0 \\ \mathbf{0}_{3,1} & 0 & 1 & 0 & 0 \\ \mathbf{0}_{3,1} & 0 & 0 & \mathbf{I} + \left(\boldsymbol{\xi}_t^{R_1}\right)_\times & \boldsymbol{\xi}_t^{p_1} \\ \mathbf{0}_{3,1} & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \left(\tilde{\omega}_t\right)_\times & \tilde{\mathbf{a}} + \mathbf{g} & \mathbf{0}_{3,1} & \mathbf{0}_{3,1} & \mathbf{0}_{3,1} \\ \mathbf{0}_{1,3} & 0 & 0 & 0 & 0 \\ \mathbf{0}_{1,3} & 0 & 0 & 0 & 0 \\ \mathbf{0}_{1,3} & 0 & 0 & 0 & 0 \\ \mathbf{0}_{1,3} & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{0}_{3,3} & (\mathbf{g})_\times\boldsymbol{\xi}_t^R & \boldsymbol{\xi}_t^v & \mathbf{0}_{3,1} & \mathbf{0}_{3,1} \\ \mathbf{0}_{1,3} & 0 & 0 & 0 & 0 \\ \mathbf{0}_{1,3} & 0 & 0 & 0 & 0 \\ \mathbf{0}_{1,3} & 0 & 0 & 0 & 0 \\ \mathbf{0}_{1,3} & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \mathcal{L}_{\mathfrak{g}}\left(\begin{bmatrix} \mathbf{0}_{3,1} \\ (\mathbf{g})_\times\boldsymbol{\xi}_t^R \\ \boldsymbol{\xi}_t^v \\ \mathbf{0}_{3,1} \\ \mathbf{0}_{3,1} \end{bmatrix}\right)$$

, where we know $-g[\boldsymbol{\xi}_\mathbf{t}^\mathbf{R}]_\times = \mathbf{g}_\times\boldsymbol{\xi}_t^R$ and get

$$\mathbf{A}_t = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 \\ (\mathbf{g})\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 \end{bmatrix}$$

Also, we can compute the adjoint operator

$$\mathrm{Ad}_{\mathbf{X}_t} = \begin{bmatrix} \mathbf{R} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ (\mathbf{v})_\times\mathbf{R} & \mathbf{R} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ (\mathbf{p})_\times\mathbf{R} & \mathbf{0} & \mathbf{R} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{R}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & (\mathbf{p_1})_\times\mathbf{R}_1 & \mathbf{R}_1 \end{bmatrix}$$

After compute the equations above, we can do our state propagation.

$$\frac{\mathrm{d}}{\mathrm{d}t}\hat{\mathbf{X}}_t = f_{u_t}\left(\hat{\mathbf{X}}_t\right) \text{ and } \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{P}_t = \mathbf{A}_t\mathbf{P}_t + \mathbf{P}_t\mathbf{A}_t^\top + \hat{\mathbf{Q}}_t$$

where $\hat{\mathbf{Q}}_t = \mathrm{Ad}_{\hat{\mathbf{x}}_t}\,\mathrm{Cov}\left(\mathbf{w}_t\right)\mathrm{Ad}_{\hat{\mathbf{x}}}^\top$

For the correction step, the MSCKF model uses the reprojection error as the measurement model. However, we still struggled to develop the Jacobian matrix and to write down the right-invariant observation equation $Y_t = X_t^{-1}b + V_t$.

IV. Experiments

A. Evaluation

We evaluated our modifications using four datasets from the EuRoC MAV dataset [15]. The datasets were "Vicon Room 1 01" (referred to as "easy"), "Vicon Room 1 02" (referred to as "medium"), and "Vicon Room 1 03" (referred to as "difficult"). Each of these datasets is collected with stereo camera images from a micro aerial vehicle as it flies around a room. Ground truth position and orientation are provided by a motion capture system as well, for comparison.

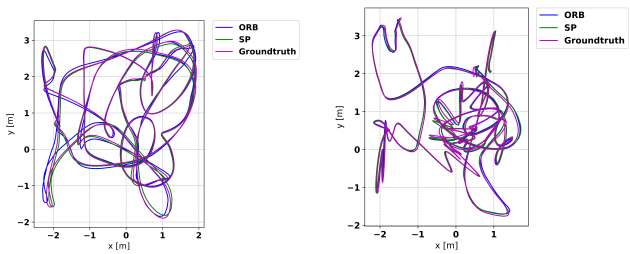
B. SuperPoint

We evaluated ORB and SuperPoint under the same environments. We used this repo to conduct the experiment. The repository implements commonly used trajectory evaluation methods for visual(-inertial) odometry [16]. The first two plots are 2D trajectories follow by two box plots. We also give the Root Mean Squared Error (RMSE) of ORB and SuperPoint on three datasets. Then, we report the run time of ORB and SuperPoint. See Fig. 5, Fig. 6, Table I and Table II

C. Invariant EKF

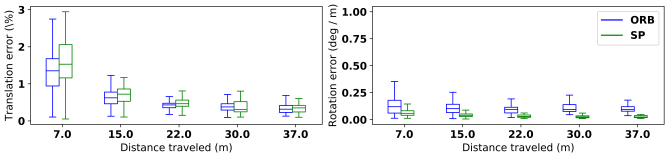
Unfortunately, two of our invariant modifications did not succeed. We struggled to formulate the observation model in a way to successfully correct the uncertainty. Results are shown in Table III.

We found that using all available visual features in the correction step resulted in an intractably slow algorithm. To compensate, we limited the number of visual features to $\frac{1}{5}$ of the total number of visual features. Clearly, we should expect some performance tradeoff here. However, we also expected

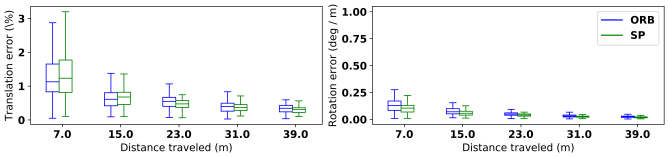


(a) Trajectory of Medium Data (b) Trajectory of Difficult Data

Fig. 5. Trajectory plots of medium and difficult data. ORB is the original method proposed in OpenVINS. SP stands for SuperPoint, which is our method.



(a) Error on Medium Data



(b) Error on Difficult Data

Fig. 6. Box plot of data from experiments

Data	Error (RMSE)	ORB (Baseline)	SP (Ours)
Easy	Position (m)	0.058	**0.051**
	Orientation (deg)	1.243	**0.763**
Medium	Position (m)	0.095	**0.060**
	Orientation (deg)	3.354	**1.980**
Difficult	Position (m)	0.051	**0.049**
	Orientation (deg)	2.393	**2.318**

TABLE I
THE TABLE COMPARES THE ERROR OF ORB AND SUPERPOINT. THE PERFORMANCE OF SUPERPOINT IS BETTER THAN ORB IN EACH OF THE EASY, MEDIUM, AND DIFFICULT DATASETS.

Data	Feature	Tracking Time (s)	Total Run Time (s)
Easy	ORB	0.0095	0.0305
	SP	0.0112	0.0321
Medium	ORB	0.0107	0.0289
	SP	0.0113	0.0290
Difficult	ORB	0.0118	0.0282
	SP	0.0141	0.0295

TABLE II
THE TABLE COMPARES THE RUN TIME OF ORB IN THE THIRD COLUMN AND SUPERPOINT IN THE FOURTH COLUMN. THE TRACKING TIME CORRESPONDS TO THE TIME CONSUMED BY EXTRACTING FEATURES. THE TOTAL RUN TIME CORRESPONDS TO THE TIME TO FINISH ONE ITERATION OF THE SLAM ALGORITHM. THE ENTIRE PROGRAM WAS RUN ON AN INTEL I7-8750H PROCESSOR.

to see a reasonable state estimate even with limited features used in each correction step.

Data	Error (RMSE)	MSCKF (Baseline)	InEKF (Ours)
Easy	Position (m)	0.107	> 100
	Orientation (deg)	2.308	> 90
Medium	Position (m)	0.114	> 100
	Orientation (deg)	3.128	> 90
Difficult	Position (m)	0.162	> 100
	Orientation (deg)	6.235	> 90

TABLE III
COMPARISON OF EUROC MAV DATASET PERFORMANCE. UNFORTUNATELY, THE INEKF IMPLEMENTATION DIVERGED FROM THE GROUND TRUTH SIGNIFICANTLY IN EACH DATASET, RESULTING IN A POSITION ERROR GREATER THAN 100 METERS AND ORIENTATION ERRORS GREATER THAN 90 DEGREES.

V. DISCUSSION

A. SuperPoint

For the Superpoint features, we evaluated both performance and run time. We analyze the trade off between them. In the performance part, you should notice that the Superpoint's performance beat the ORB features in all the tasks, especially in the medium level task. Also, when we observe the Fig. 6, notice that Superpoint can also improve the precision of the whole system. We think this is because Superpoint is a learning-based feature extractor which captures better quality features than the handcrafted feature extractor. We also found supporting evidence for this observation in the Superpoint paper [7]. The authors showed that Superpoint feature had the highest matching score among the state-of-the-art feature extactors, which means it is good at being recovered by the whole pipeline over the number of features proposed by the pipeline in the shared viewpoint region.

The running time of Superpoint is 5 to 15 % slower than the ORB feature depending on the task. Nevertheless, it is fast enough to be called real-time. It is impressive because we run our model for our project on CPU rather than GPU. By implementing parallel computing on GPU, the whole system can be sped up.

B. Invariant EKF

The invariant EKF requires additional thought. Our attempts did not succeed in improving the performance. Future work should examine the best method for propagating covariance of the augmented camera poses in the state estimate. This was the main area we struggled with. The IMU state covariance can be predicted, but we had difficulty finding a method for computing the camera pose covariance prediction. The correction step may also be accomplished using a different method.

We attempted to use the estimated 3D global positions of each visual features for a right invariant correction step. We found that this method was quite slow when using all available visual features. Thus, we limited the number of visual features used in each correction step. We did not use a selection criteria for deciding which visual features to include. Future work may

find a way to quantify the quality of each visual feature. Then the most promising visual features can be used for correction. Various metrics could be used, such as using the features seen from the most camera poses, or using the features with the lowest residual error during optimization.

VI. Conclusion

We evaluated two changes to the OpenVINS library. The first change, SuperPoint visual features, demonstrated a significant improvement in performance over the baseline visual features. Also, the running time is acceptable. The worst case is about 70Hz, which can be seen as a real-time system. The second change, invariance error state estimation, resulted in very poor performance. We attempted to combine both the MSCKF and InEKF using a dual-track form. When this formulation failed, we attempted to derive the RI-MSCKF from scratch. We hypothesize that a different form of observation model and correction step could enable the invariant EKF to achieve similar or better performance to the MSCKF.

Acknowledgment

Thanks to the Mobile Robotics course staff for guidance and advice throughout the course of the project.

References

[1] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "Openvins: A research platform for visual-inertial estimation," in *Proc. of the IEEE International Conference on Robotics and Automation*, Paris, France, 2020. [Online]. Available: https://github.com/rpng/open_vins

[2] B. P. Wrobel, "Multiple view geometry in computer vision," *KI*, vol. 15, p. 41, 2001.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, p. 91–110, Nov. 2004. [Online]. Available: https://doi.org/10.1023/B:VISI.0000029664.99615.94

[4] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, p. 346–359, Jun. 2008. [Online]. Available: https://doi.org/10.1016/j.cviu.2007.09.014

[5] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, Nov 2011, pp. 2564–2571.

[6] D. Nistér, O. Naroditsky, and J. R. Bergen, "Visual odometry," *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1, pp. I–I, 2004.

[7] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 337–33 712, 2017.

[8] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular slam with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017.

[9] T. Zhang, K. Wu, D. Su, S. Huang, and G. Dissanayake, "An invariant-ekf vins algorithm for improving consistency," 2017.

[10] M. Brossard, S. Bonnabel, and A. Barrau, "Invariant kalman filtering for visual inertial slam," in *2018 21st International Conference on Information Fusion (FUSION)*. IEEE, 2018, pp. 2021–2028.

[11] J. Solà, "Quaternion kinematics for the error-state kalman filter," *CoRR*, vol. abs/1711.02508, 2017. [Online]. Available: http://arxiv.org/abs/1711.02508

[12] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust stereo visual inertial odometry for fast autonomous flight," 2017.

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

[14] R. Hartley, M. G. Jadidi, J. W. Grizzle, and R. M. Eustice, "Contact-aided invariant extended kalman filtering for legged robot state estimation," *arXiv preprint arXiv:1805.10410*, 2018.

[15] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract

[16] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.

[17] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, pp. 1147–1163, 2015.

[18] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[19] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, 2012, pp. 37–49.

Appendix

A. Discrete IMU Nonlinear Dynamics

$$
\begin{aligned}
\mathbf{p} &= \mathbf{p} + \mathbf{v}\Delta t + \tfrac{1}{2}\left(\mathbf{R}\left(\mathbf{a}_m - \mathbf{a}_b\right) + \mathbf{g}\right)\Delta t^2 \\
\mathbf{v} &= \mathbf{v} + \left(\mathbf{R}\left(\mathbf{a}_m - \mathbf{a}_b\right) + \mathbf{g}\right)\Delta t \\
\mathbf{q} &= \mathbf{q} \otimes \mathbf{q}\left\{\left(\boldsymbol{\omega}_m - \boldsymbol{\omega}_b\right)\Delta t\right\}
\end{aligned}
$$

B. MSCKF Observation Model

MSCKF observation model presented in [9].

The camera frame coordinates from global coordinates.

$$
{}^{C_i}\mathbf{p}_{f_j} = \begin{bmatrix} {}^{C_i}X_j \\ {}^{C_i}Y_j \\ {}^{C_i}Z_j \end{bmatrix} = \mathbf{C}\left(\ {}_{iG}\bar{q}\right)\left({}^{G}\mathbf{p}_{f_j} - {}^{G}\mathbf{p}_{C_i}\right)
$$

Residual

$$
\mathbf{r}_i^{(j)} = \mathbf{z}_i^{(j)} - \hat{\mathbf{z}}_i^{(j)}
$$

Linearization using Jacobians with respect to state and with respect to feature

$$
\mathbf{H}_{\mathbf{X}_i}^{(j)} = \begin{bmatrix} \mathbf{0}_{2\times 15} & \mathbf{0}_{2\times 6} & \cdots & \mathbf{J}_i^{(j)}\left[{}^{C_i}\hat{\mathbf{X}}_{f_j\times}\right] & -\mathbf{J}_i^{(j)}\mathbf{C}\left({}^{C_i}_{G}\hat{q}\right) \\ & & & \text{Jacobian wrt pose i} & \end{bmatrix}
$$

$$
\mathbf{H}_{f_i}^{(j)} = \mathbf{J}_i^{(j)}\mathbf{C}\left({}^{C_i}_{G}\hat{q}\right)
$$

Jacobian of the feature image coordinates with respect to the feature camera frame position

$$
\mathbf{J}_i^{(j)} = \nabla_{{}^{C_i}\hat{\mathbf{p}}_{f_j}}\mathbf{z}_i^{(j)} = \frac{1}{\frac{1}{{}^{C_i}\hat{Z}_j}}\begin{bmatrix} 1 & 0 & -\frac{{}^{C_i}\hat{X}_j}{{}^{C_i}\hat{Z}_j} \\ 0 & 1 & -\frac{{}^{C_i}\hat{Y}_j}{{}^{C_i}\hat{Z}_j} \end{bmatrix}
$$

Stacking all residuals and Jacobians:

$$
\mathbf{r}^{(j)} \simeq \mathbf{H}_{\mathbf{X}}^{(j)}\widetilde{\mathbf{X}} + \mathbf{H}_f^{(j)G}\widetilde{\mathbf{p}}_{f_j} + \mathbf{n}^{(j)}
$$

Using A to represent the left nullspace of H_f

$$
\begin{aligned}
\mathbf{r}_o^{(j)} &= \mathbf{A}^T\left(\mathbf{z}^{(j)} - \hat{\mathbf{z}}^{(j)}\right) \simeq \mathbf{A}^T\mathbf{H}_{\mathbf{X}}^{(j)}\widetilde{\mathbf{X}} + \mathbf{A}^T\mathbf{n}^{(j)} \\
&= \mathbf{H}_o^{(j)}\widetilde{\mathbf{X}}^{(j)} + \mathbf{n}_o^{(j)}
\end{aligned}
$$

C. Dual Track Code Class Reference

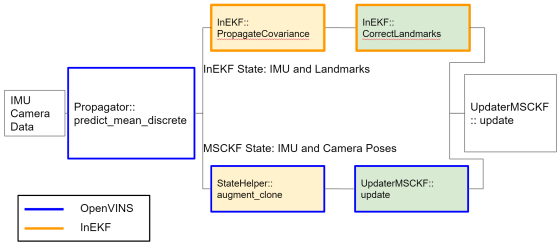


Fig. 7. The combined pipeline for MSCKF and InEKF. MSCKF blocks are outlined in blue and InEKF blocks are outlined in orange. Classes and class methods are documented