Towards End-to-end Learning of Visual Inertial Odometry with an EKF

Chunshang Li
Institute for Aerospace Studies
University of Toronto
Toronto, Canada
chunshang.li@robotics.utias.utoronto.ca

Steven L. Waslander
Institute for Aerospace Studies
University of Toronto
Toronto, Canada
steven.waslander@robotics.utias.utoronto.ca

***Abstract*—Classical visual-inertial fusion relies heavily on manually crafted image processing pipelines, which are prone to failure in situations with rapid motion and texture-less scenes. While end-to-end learning methods show promising results in addressing these limitations, embedding domain knowledge in the form of classical estimation processes within the end-to-end learning architecture has the potential of combining the best of both worlds. In this paper, we propose the first end-to-end trainable visual-inertial odometry (VIO) algorithm that leverages a robo-centric Extended Kalman Filter (EKF). The EKF propagates states through a known inertial measurement unit (IMU) kinematics model and accepts relative pose measurements and uncertainties from a deep network as updates. The system is fully differentiable and can be trained end-to-end through backpropagation. Our method achieves a translation error of 1.27% on the KITTI odometry dataset, which is competitive among classical and learning VIO methods. The implementation is publicly available on GitHub[1].**

***Keywords*-visual inertial odometry; localization; deep learning; robo-centric EKF; supervised deep learning**

I. Introduction

Ego-motion estimation is a key challenge in the field of autonomous mobile robotics. In the past few decades, there has been growing interest in VIO due to the low-cost nature and proliferation of cameras and Micro-Electro-Mechanical Systems (MEMS) IMUs on many robots and mobile electronic devices. Odometry estimation based on only IMU data is prone to large error accumulation in a relatively short amount of time. Combining exteroceptive measurements from cameras has proven to effectively reduce the amount of drift, with the added benefit of the IMU providing high rate state estimation and removing scale ambiguity in a monocular setting. Current VIO systems, such as [1] and [2], rely on techniques from the state-of-the-art Visual Odometry (VO) methods with a typical pipeline consisting of feature detection, feature matching, outlier rejection, initialization, and optimization. Although these methods have demonstrated impressive results, they are still prone to failure under rapid motion, motion blur, exposure changes, and low texture environments. The effects are often difficult to model, and loss of feature tracking could easily cause the optimization process to diverge. In addition, classical VIO systems require extensive parameter tuning and calibration to work. Each stage of the estimation pipeline has parameters that need to be adjusted, and performance improvement in one stage of the pipeline does not necessarily translate into performance improvement of the overall system.

[1]https://github.com/lichunshang/deep_ekf_vio

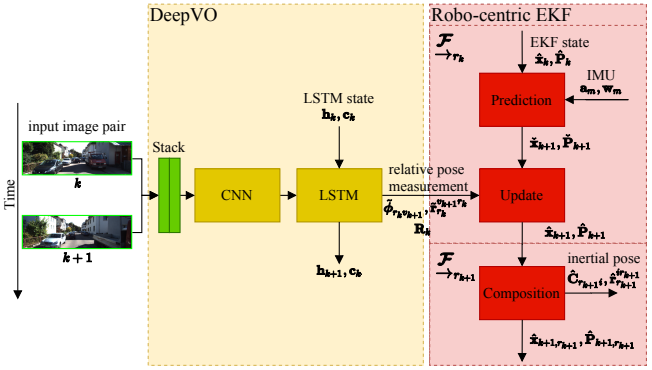

Figure 1. Overview of our system at a given timestep from k to $k+1$, the robo-centric EKF propagates states according to IMU kinematics model and accepts measurements from a deep network

The rise in deep learning provided the tools to directly process high dimensional data, such as images, rather than through hand-crafted feature extractors. This allows state estimation to be learned directly from data in an end-to-end manner, leveraging both local and higher level features, demonstrated in previous works, such as [3], [4], and [5]. Also, the deep network could learn additional underlying models that may improve state estimation, such as rejecting dynamic objects, camera exposure changes, and scale of objects in the scene. On top of this, learning methods open the door for systems to automatically gain knowledge during testing and large-scale deployment in the future. In contrast to images, IMU data is low dimensional and has well-understood models that are grounded in physics. We argue that, unlike the approach in [3], it is not always necessary to use a deep network to learn these physical models. Incorporating known state estimation processes has been shown to improve data efficiency and reduce over-fitting [6], [7], [8].

In this work, we explore these ideas further, designing a deep network for monocular VIO that incorporates elements of classical VIO algorithm, and constrain the learning to the portion of the pipeline that is composed of difficult-to-model processes. Our proposed method centers around a robo-centric EKF, which propagates its states based on IMU kinematics and fuses learned visual relative pose measurements and uncertainties from a deep network in its update step. The differentiability of the system is maintained, which allows the entire system to be trained end-to-end through backpropagation. This setup is more advantageous than learning the measurements individually as a sub-component since the overall system can be optimized according to the final objective. We tested and shown the effectiveness of our system on the KITTI odometry [9] and EuRoC drone [10] datasets. To the best of our knowledge, our method is the first to explore learning VIO end-to-end with the incorporation of a classical state estimator.

II. Related Work

A. Classical VIO Methods

There has been extensive work on classical model-driven monocular VIO methods. These methods can be divided into loosely-coupled and tightly-coupled methods. Loosely-coupled approaches such as Multi-Sensor Fusion (MSF) [11] accept generic up-to-scale inertial pose sensors, which can be obtained from a VO pipeline, and fuse it with IMU propagated states, usually through an EKF. In addition to estimating pose, velocities, and biases standard to other VIO methods, MSF keeps a scalar parameter to estimate the drifting scale from monocular VO. Tightly-coupled methods such as MSCKF [12], OKVIS [1], and VINS-Mono [2] extract, track, and triangulate features from images, and fuse the features with IMU propagated poses. In the case of MSCKF, it uses least-squares optimization to triangulate the features and fusion is performed in an EKF. Whereas OKVIS and VINS-Mono accomplish fusion through iterative non-linear least-squares optimizations. Direct VIO methods such as [13] and [14] have also been proposed, which perform estimation based on warped image patch intensities rather than using reprojected features, allowing them to perform more robustly in low texture scenes. Our proposed method is a loosely-coupled filter-based approach that accepts learned relative pose measurements with predicted scale. Instead of operating on image patches or features, the network computes relative pose using the entire image.

B. Learning-Based Methods

Deep learning methods have demonstrated great success in fields such as classification, object detection, and natural language processing. Recently, there had been a surge of work in applying deep learning methods to state estimation tasks. The first method to formulate VO as a sequential learning problem is DeepVO, which uses recurrent convolutional neural network (RCNN) to regress poses from a sequence of images. It takes a pair of images, feeds it through a FlowNet [15] encoder, Long Short-Term Memory (LSTM) layers, and a fully connected network to produce a 6-DoF pose. ESP-VO [5] extended this work by incorporating uncertainty estimation through a loss function based on maximum likelihood. Also, it uses a SE(3) layer to compose relative poses in the neural network. ESP-VO demonstrated that deep learning methods could be resilient to rapid motion, motion blur, exposure change, and rolling shutter effects. VINet [3] was the first to apply learning to visual-inertial odometry. It uses LSTMs to process high-rate IMU data which is concatenated with a feature vector from images processed through a convolutional neural network (CNN), then fed through another LSTM layer producing relative poses. The relative poses are composed through a SE(3) layer similar to [5]. Although VINet demonstrated promising results for a deep learning approach to VIO, the application of deep networks to learn IMU dynamics might be unnecessary. Unlike vision, which is affected by hard-to-model environmental conditions, IMUs are not affected by scene appearances and its physical model is well studied.

There has been growing interest in hybrid systems that incorporate classical state estimation as part of end-to-end training. LS-Net [16] uses an LSTM network to learn nonlinear least-squares optimization updates for dense reconstruction. Backprop KF [8] estimates odometry by formulating an end-to-end trainable EKF with 2D pose and velocity as states and updates using velocity measurements obtained from a deep network consisting of convolutional and fully connected layers. In this spirit, [17] formulated an IMU-only dead-reckoning system with learned covariances for pseudo-measurements EKF updates. An end-to-end trainable histogram filter is proposed in [18] and its effectiveness is demonstrated under simple localization tasks. DPF [6] and PF-Net [7] concurrently proposed particle filter networks that enable the learning of motion and measurement model of in an end-to-end manner. In these works, domain-specific knowledge in robotics in the form of classical estimators is incorporated as part of a deep network. Experiments in [7] show that in comparison to using full deep networks, algorithmic priors improve performance and explainability of the network as learning is constrained by the underlying problem structure. In addition, the ability to train the entire system end-to-end has also been shown to improve performance as opposed to training each subsystem individually. Our work follows these inspirations and applies the ideas to the task of VIO. Unlike previous works that formulate problems for simple localization tasks or 2D planar motion estimation [8][18][6][7], we formulate a full 6-DoF estimation problem in line with existing VIO literature.

III. Estimator Formulation

The goal of the VIO estimator is to estimate the vehicle frame with respect to an inertial frame of reference. The core of this estimator is the robo-centric EKF formulation first proposed in [19]. Robo-centric EKF formulation defines all its states with respect to a local reference frame. This includes the inertial states which are tracked as a "feature" with respect to the latest reference state. During EKF prediction, the vehicle states are propagated in the local reference frame. After performing an EKF update with the arrival of each new image, the robo-centric EKF shifts the reference frame from one timestep to the next in the composition step. The process then repeats in the new reference frame. The three step process of prediction, update, and composition is illustrated in Figure 1. The robo-centric formulation is advantageous since it allows for a natural way of incorporating relative measurements. The non-robocentric formulation requires relative measurements to be composed together with their uncertainties, where the uncertainty of the composed pose will grow unboundedly over time.

We now define the notations used throughout this paper. $\underrightarrow{\mathcal{F}}_i$ is the inertial frame which is fixed to the earth. $\underrightarrow{\mathcal{F}}_{r_k}$ is the reference frame at time k, where time $k, k+1, \dots$ corresponds to points in time when an image is received. $\underrightarrow{\mathcal{F}}_{v_\tau}$ is the vehicle frame at time τ, where $\tau, \tau+1, \dots$ corresponds to times when a IMU measurement is received between time k and $k+1$. Superscripts and subscripts are used to keep track the coordinate frames of physical quantities. For example, $\mathbf{v}_a^{bc}$ is the velocity of $\underrightarrow{\mathcal{F}}_b$ with respect to $\underrightarrow{\mathcal{F}}_c$ expressed in $\underrightarrow{\mathcal{F}}_a$, and $\mathbf{C}_{ab}$ is the rotation of $\underrightarrow{\mathcal{F}}_b$ with respect to $\underrightarrow{\mathcal{F}}_a$. Accents are also used to distinguish different types of quantities: $\bar{(\cdot)}$ is used to denote noisy quantities, $(\cdot)$ is used to denote propagated quantities throughout EKF prediction, and $\hat{(\cdot)}$ is used to denote corrected quantities after EKF update.

A. States

Nominal states are shown in (1), divided into inertial states $\mathbf{x}_{r_k i}$ and vehicle states $\mathbf{x}_{r_k v_\tau}$. For the inertial states, $(\mathbf{C}_{r_k i}, \mathbf{r}_{r_k}^{i r_k}) \in \mathbb{SE}(3)$ comprise the pose of $\underrightarrow{\mathcal{F}}_i$ in $\underrightarrow{\mathcal{F}}_{r_k}$, and $\mathbf{g}_{r_k} \in \mathbb{R}^3$ is the gravity vector expressed in $\underrightarrow{\mathcal{F}}_{r_k}$. For the vehicle states, $(\mathbf{C}_{r_k v_\tau}, \mathbf{r}_{r_k}^{v_\tau r_k}) \in \mathbb{SE}(3)$ comprise the pose of $\underrightarrow{\mathcal{F}}_{v_\tau}$ in $\underrightarrow{\mathcal{F}}_{r_k}$, and $\mathbf{v}_{v_\tau}^{v_\tau i}, \mathbf{b}_{\omega_\tau}, \mathbf{b}_{a_\tau} \in \mathbb{R}^3$ are vehicle velocity, gyroscope bias, and accelerometer bias, respectively, in the latest vehicle frame $\underrightarrow{\mathcal{F}}_{v_\tau}$.

$$\begin{aligned} \mathbf{x}_\tau &= [\mathbf{x}_{r_k i} \quad \mathbf{x}_{r_k v_\tau}] \\ \mathbf{x}_{r_k i} &= [\mathbf{C}_{r_k i} \quad \mathbf{r}_{r_k}^{i r_k} \quad \mathbf{g}_{r_k}] \\ \mathbf{x}_{r_k v_\tau} &= [\mathbf{C}_{r_k v_\tau} \quad \mathbf{r}_{r_k}^{v_\tau r_k} \quad \mathbf{v}_{v_\tau}^{v_\tau i} \quad \mathbf{b}_{\omega_\tau} \quad \mathbf{b}_{a_\tau}] \end{aligned} \tag{1}$$

Error state vectors corresponding to the nominal states are shown in (2). The error states are defined as perturbations on the nominal states, with all except rotations defined in the Euclidean space. The perturbations of rotation states are defined in (3), where $\mathbf{C} \in \mathbb{SO}(3)$ and $\boldsymbol{\phi} \in \mathbb{R}^3$, and $(\cdot)^\wedge$ is the skew-symmetric operator.

$$\begin{aligned} \delta\mathbf{x}_\tau &= [\delta\mathbf{x}_{r_k i}^\top \quad \delta\mathbf{x}_{r_k v_\tau}^\top]^\top \\ \delta\mathbf{x}_{r_k i} &= [\delta\boldsymbol{\phi}_{r_k i}^\top \quad \delta\mathbf{r}_{r_k}^{i r_k T} \quad \delta\mathbf{g}_{r_k}^\top]^\top \\ \delta\mathbf{x}_{r_k v_\tau} &= [\delta\boldsymbol{\phi}_{r_k v_\tau}^\top \quad \delta\mathbf{r}_{r_k}^{v_\tau r_k \top} \quad \delta\mathbf{v}_{v_\tau}^{v_\tau i \top} \quad \delta\mathbf{b}_{\omega_\tau}^\top \quad \delta\mathbf{b}_{a_\tau}^\top]^\top \end{aligned} \tag{2}$$

$$\mathbf{C} = \bar{\mathbf{C}} \exp(\boldsymbol{\phi}^\wedge) \tag{3}$$

B. Prediction

1) Continuous Time: We first start with the IMU model shown in (4). The measured accelerometer values, $\mathbf{a}_m$, and gyroscope values, $\boldsymbol{\omega}_m$, are defined in the vehicle frame at time τ, $\underrightarrow{\mathcal{F}}_{v_\tau}$, and are affected by noise $\mathbf{n}_a$ and $\mathbf{n}_\omega$, and biases $\mathbf{n}_{b_a}$ and $\mathbf{n}_{b_w}$, respectively. The accelerometer measurement is also combined with a fixed gravity vector, rotated to $\underrightarrow{\mathcal{F}}_{v_\tau}$. The random noise on the accelerometer and gyroscope, as well as the derivative of random walk on the biases are assumed to be zero mean Gaussian $\mathbf{n}_a \sim \mathcal{N}(\mathbf{0}, \sigma_a^2\mathbf{I})$, $\mathbf{n}_\omega \sim \mathcal{N}(\mathbf{0}, \sigma_\omega^2\mathbf{I})$, $\dot{\mathbf{b}}_{a_\tau} = \mathbf{n}_{b_a}$, $\dot{\mathbf{b}}_{\omega_\tau} = \mathbf{n}_{b_\omega}$, $\mathbf{n}_{b_a} \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{b}_a}^2\mathbf{I})$, $\mathbf{n}_{b_\omega} \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{b}_\omega}^2\mathbf{I})$.

$$\begin{aligned} \mathbf{a}_m &= \mathbf{a}_{v_\tau}^{v_\tau i} + \mathbf{C}_{v_\tau r_k}\mathbf{g}_{r_k} + \mathbf{b}_{a_\tau} + \mathbf{n}_a \\ \boldsymbol{\omega}_m &= \boldsymbol{\omega}_{v_\tau}^{v_\tau i} + \mathbf{b}_{\omega_\tau} + \mathbf{n}_\omega \end{aligned} \tag{4}$$

The continuous time evolution of true nominal vehicle states are described in (5), with the inertial states remaining constant.

$$\begin{aligned} \dot{\mathbf{C}}_{r_k v_\tau} &= \mathbf{C}_{r_k v_\tau}[\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge, \quad \dot{\mathbf{r}}_{r_k}^{v_\tau r_k} = \mathbf{C}_{r_k v_\tau}\mathbf{v}_{v_\tau}^{v_\tau i}, \\ \dot{\mathbf{v}}_{v_\tau}^{v_\tau i} &= \mathbf{a}_{v_\tau}^{v_\tau i} - [\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge \mathbf{v}_{v_\tau}^{v_\tau i}, \quad \dot{\mathbf{b}}_{\omega_\tau} = \mathbf{n}_{b_\omega}, \quad \dot{\mathbf{b}}_{a_\tau} = \mathbf{n}_{b_a} \end{aligned} \tag{5}$$

After applying the defined perturbations to all vehicle states, substituting in (4) and (5), solving for the error states, and removing all the second order terms, we arrive at (6) in matrix form with the linearized system matrix $\mathbf{F}$, linearized error matrix $\mathbf{G}$, and noise $\mathbf{n} = \begin{bmatrix}\mathbf{n}_\omega^\top & \mathbf{n}_{b_\omega}^\top & \mathbf{n}_a^\top & \mathbf{n}_{b_a}^\top\end{bmatrix}^\top$. We omit the derived values of $\mathbf{F}$ and $\mathbf{G}$ here for brevity, readers can refer to [19] for detailed derivations.

$$\delta\dot{\mathbf{x}}_\tau = \mathbf{F}\delta\mathbf{x}_\tau + \mathbf{G}\mathbf{n} \tag{6}$$

2) Discrete Time: In order to implement EKF prediction, we need to discretize the continuous model for the nominal and error states. For the nominal states we apply (7), where $\Delta t = t_{k+1} - t_k$. For simplicity, the velocity is integrated in $\underrightarrow{\mathcal{F}}_{r_k}$, and rotated to $\underrightarrow{\mathcal{F}}_{v_{k+1}}$ once all IMU measurements are received. We use Euler's method to approximate the integrals shown in (7). All other states remain constant throughout EKF prediction.

$$\begin{aligned} \check{\mathbf{C}}_{r_k v_{k+1}} &= \int_{t\in[t_k, t_{k+1}]} \check{\mathbf{C}}_{r_k v_s}(\boldsymbol{\omega}_m - \check{\mathbf{b}}_{\omega_s})^\wedge dt \\ \check{\mathbf{v}}_{r_k}^{v_{k+1} i} &= \hat{\mathbf{v}}_{r_k}^{v_k i} - \mathbf{g}_{r_k}\Delta t + \int_{t\in[t_k, t_{k+1}]} \check{\mathbf{C}}_{r_k v_s}(\mathbf{a}_m - \check{\mathbf{b}}_{a_s})dt \end{aligned} \tag{7}$$

$$\check{\mathbf{r}}_{r_k}^{v_{k+1}r_k} = \hat{\mathbf{v}}_{r_k}^{v_k i}\Delta t - \frac{1}{2}\check{\mathbf{g}}_{r_k}\Delta t^2 + \iint_{t\in[t_k,t_{k+1}]} \check{\mathbf{C}}_{r_k v_s}(\mathbf{a}_m - \check{\mathbf{b}}_{a_s})dt^2$$

The error state vector itself remains zero throughout EKF prediction, and (6) is only used to propagate error state covariances. From time t_τ to $t_{\tau+1}$, (8) is used to find the state transition matrix $\mathbf{\Phi}_{\tau+1,\tau}$ using first order approximation, where $\delta t = t_{\tau+1} - t_\tau$.

$$\mathbf{\Phi}_{\tau+1,\tau} = \exp\left(\int_{t_\tau}^{t_{\tau+1}} \mathbf{F}(s)ds\right) \approx \mathbf{I} + \mathbf{F}_\tau \delta t \tag{8}$$

We then propagate state covariance with each new IMU measurement to t_{k+1} according to (10), where $\mathbf{Q}$ is the IMU covariance matrix described in (9).

$$\mathbf{Q} = \mathrm{diag}(\sigma_w^2\mathbf{I}, \sigma_{\mathbf{b}_w}^2\mathbf{I}, \sigma_a^2\mathbf{I}, \sigma_{\mathbf{b}_a}^2\mathbf{I}) \tag{9}$$

$$\check{\mathbf{P}}_{\tau+1} = \mathbf{\Phi}_{\tau+1,\tau}\check{\mathbf{P}}_\tau\mathbf{\Phi}_{\tau+1,\tau}^\top + \mathbf{G}\mathbf{Q}\mathbf{G}^\top\delta t \tag{10}$$

C. Update

1) Measurement Model: Given relative pose measurements $\tilde{\mathbf{z}}_k = \begin{bmatrix}\tilde{\boldsymbol{\phi}}_{r_k v_{k+1}}^\top & \tilde{\mathbf{r}}_{r_k}^{v_{k+1}r_k\top}\end{bmatrix}^\top$ described in detail in Section IV, with corresponding covariances $\mathbf{R}_k$, the measurement residual $\boldsymbol{\epsilon}_{k+1} = \begin{bmatrix}\boldsymbol{\epsilon}_\theta^\top & \boldsymbol{\epsilon}_r^\top\end{bmatrix}^\top$ is shown in (11).

$$\begin{bmatrix}\boldsymbol{\epsilon}_\theta \\ \boldsymbol{\epsilon}_r\end{bmatrix} = \begin{bmatrix}\ln(\exp(\tilde{\boldsymbol{\phi}}_{r_k v_{k+1}}^\wedge)\mathbf{C}_{r_k v_{k+1}}^\top)^\vee \\ \tilde{\mathbf{r}}_{r_k}^{v_{k+1}r_k} - \mathbf{r}_{r_k}^{v_{k+1}r_k}\end{bmatrix} \tag{11}$$

Since $\mathbf{z}_k$ is produced by a deep network, the residual (11) cannot be used directly in an end-to-end training scheme as $\tilde{\boldsymbol{\phi}}_{r_k v_{k+1}}$ could be far from the ground truth at the start of the training process, causing opportunities for $\boldsymbol{\epsilon}_\theta$ to be close to π. This is undesirable for training stability because $\boldsymbol{\phi} = \ln(\mathbf{C})^\vee$ is not differentiable as $\boldsymbol{\phi} \to \pi$. To circumvent this issue, we applied first order Baker-Campbell-Hausdorff (BCH) formula shown in (12) which approximates the log mapping as subtraction of two rotational vectors. This approximation is valid since both rotations will be small.

$$\begin{aligned}\boldsymbol{\epsilon}_\theta &= \ln(\exp(\tilde{\boldsymbol{\phi}}_{r_k v_{k+1}}^\wedge)\exp(\boldsymbol{\phi}_{r_k v_{k+1}}^\wedge)^\top)^\vee \\ &\approx \tilde{\boldsymbol{\phi}}_{r_k v_{k+1}} - \boldsymbol{\phi}_{r_k v_{k+1}}\end{aligned} \tag{12}$$

We then differentiate (11) with respect to the error states to find the measurement Jacobian $\mathbf{H}_{k+1} = \frac{\partial \boldsymbol{\epsilon}_{k+1}}{\partial \delta\mathbf{x}_{k+1}}$. The derivations are shown in (13), and the final $\mathbf{H}_{k+1}$ is shown in (14), where $\mathbf{J}$ is the right Jacobian of $\mathbb{SO}(3)$.

$$\begin{aligned}\boldsymbol{\epsilon}_\theta &\approx \tilde{\boldsymbol{\phi}}_{r_k v_{k+1}} - \ln(\mathbf{C}_{r_k v_{k+1}}) \\ &\approx \underbrace{\tilde{\boldsymbol{\phi}}_{r_k v_{k+1}} - \hat{\boldsymbol{\phi}}_{r_k v_{k+1}}}_{\bar{\epsilon}_\theta} - \mathbf{J}_r(\boldsymbol{\phi}_{r_k v_{k+1}})^{-1}\delta\boldsymbol{\phi}_{r_k v_{k+1}} \\ \boldsymbol{\epsilon}_r &= \underbrace{\tilde{\mathbf{r}}_{r_k}^{v_{k+1}r_k} - \hat{\mathbf{r}}_{r_k}^{v_{k+1}r_k}}_{\bar{\epsilon}_r} - \delta\mathbf{r}_{r_k}^{v_{k+1}r_k}\end{aligned} \tag{13}$$

$$\mathbf{H}_{k+1} = \begin{bmatrix}\mathbf{0}_{9\times3} & -\mathbf{J}(-\check{\boldsymbol{\phi}}_{r_k v_{k+1}})^{-1} & \mathbf{0} & \mathbf{0}_{9\times3} \\ \mathbf{0}_{9\times3} & \mathbf{0} & -\mathbf{I} & \mathbf{0}_{9\times3}\end{bmatrix} \tag{14}$$

2) EKF Update: A regular EKF update is performed as shown in (15) to obtain a estimate of the error $\delta\hat{\mathbf{x}}_{k+1}$.

$$\begin{aligned}\mathbf{K}_{k+1} &= \check{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T(\mathbf{H}_{k+1}\check{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T + \mathbf{R}_{k+1})^{-1} \\ \hat{\mathbf{P}}_{k+1} &= (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1})\check{\mathbf{P}}_{k+1} \\ \delta\hat{\mathbf{x}}_{k+1} &= \mathbf{K}_{k+1}\bar{\boldsymbol{\epsilon}}_{k+1}\end{aligned} \tag{15}$$

Then, $\delta\hat{\mathbf{x}}_{k+1}$ is injected into the predicted nominal states according to their defined perturbations in Section III-A, yielding $\hat{\mathbf{x}}_{k+1}$. Note that up to this point, the estimate for the rotation and position in the reference frame $\mathbf{C}_{r_k v_\tau}$ and $\mathbf{r}_{r_k}^{v_\tau r_k}$ have not changed.

D. Composition

Finally, the reference frame for all states are shifted forward from frame $\underrightarrow{\mathcal{F}}_{r_k}$ to frame $\underrightarrow{\mathcal{F}}_{r_{k+1}}$, using (16), obtaining $\hat{\mathbf{x}}_{k+1,r_{k+1}}$. We reset the local vehicle pose for the next EKF iteration after it has been composed with the inertial pose. The velocity and biases are not changed since they are already in $\underrightarrow{\mathcal{F}}_{r_{k+1}}$.

$$\begin{aligned}\hat{\mathbf{C}}_{r_{k+1}i} = \hat{\mathbf{C}}_{r_k v_{k+1}}^T\hat{\mathbf{C}}_{r_k i}, \quad \hat{\mathbf{g}}_{r_{k+1}} = \hat{\mathbf{C}}_{r_k v_{k+1}}^T\hat{\mathbf{g}}_{r_k}, \\ \hat{\mathbf{r}}_{r_{k+1}}^{ir_{k+1}} = \check{\mathbf{C}}_{r_k v_{k+1}}^T(\hat{\mathbf{r}}_{r_k}^{ir_k} - \hat{\mathbf{r}}_{r_k}^{v_{k+1}r_k}), \\ \check{\mathbf{C}}_{r_{k+1}v_{k+1}} = \mathbf{I}, \quad \check{\mathbf{r}}_{r_{k+1}}^{v_{k+1}r_{k+1}} = \mathbf{0}\end{aligned} \tag{16}$$

The state covariances must also be propagated to reflect the operation in (16), according to (17). Again for brevity, we omit the contents of $\mathbf{U}_{k+1}$, and readers can refer to [19] for details.

$$k\hat{\mathbf{P}}_{k+1,r_{k+1}} = \mathbf{U}_{k+1}\hat{\mathbf{P}}_{k+1}\mathbf{U}_{k+1}^T,\ \mathbf{U}_{k+1} = \frac{\partial\delta\hat{\mathbf{x}}_{k+1,r_{k+1}}}{\partial\delta\hat{\mathbf{x}}_{k+1}} \tag{17}$$

IV. Learning VIO End-to-End

In this section, we describe how the proposed system integrates relative visual measurements with the estimator formulation described in Section III to produce an end to end trainable system. The overall approach is summarized in Figure 1. The network produces relative pose estimation with uncertainties to update the EKF according to Section III-C.

A. Relative Pose Measurements

We adopted the DeepVO [4] architecture to produce relative pose measurements for EKF updates. DeepVO and its extension ESP-VO [5] are deep neural networks that learns monocular visual odometry in a supervised fashion. The network combines a convolutional neural network (CNN) with a recurrent neural network (RNN) to learn visual odometry from raw RGB images in an end-to-end manner. The CNN learns feature extraction from images and the RNN provides the system with the opportunity to learn complex motion dynamics over a sequence of data and maintain a consistent scale output over time.

The input to the network at each timestep is a pair of images at time t_k and t_{k+1} stacked along the RGB channel. The images are then processed through a series of CNN layers. The architecture of the CNN layers is based on the encoder section of FlowNetS [15]. To learn features at different scales, 9 CNN layers are used with varying kernel sizes, padding and strides. The features extracted by the CNN are flattened into a vector and passed into the RNN. A Long Short-Term Memory (LSTM) RNN is used to alleviate the vanishing and exploding gradient problem for learning long sequences. Finally, the LSTM output is passed into a Fully Connected Network (FCN) to produce a 12 dimensional vector $\mathbf{y}_k = \begin{bmatrix} \tilde{\boldsymbol{\phi}}_{r_k v_{k+1}}^\top & \tilde{\mathbf{r}}_{r_k}^{v_{k+1} r_k \top} & \mathbf{w}_{\boldsymbol{\phi}}^\top & \mathbf{w}_{\mathbf{r}}^\top \end{bmatrix}^\top$ representing the 6-DoF relative motion and the corresponding uncertainties. We assume the measurement uncertainties are uncorrelated between the 6-DoF motion, hence the covariance $\mathbf{R}_k$ can be simplified as a diagonal matrix. Similar to [17], we apply (18) elementwise to $\mathbf{w}_{\boldsymbol{\phi}_k}$ and $\mathbf{w}_{\mathbf{r}_k}$ to obtain the diagonals of $\mathbf{R}_k$, where σ_0^2 serves as a initial guess of uncertainty and $\beta \in \mathbb{R}_{>0}$ is used to bound σ^2.

$$\sigma^2 = \sigma_0^2 10^{\beta \tanh(w)} \tag{18}$$

B. Training

To train the proposed system end-to-end, we employ the sum of two loss functions, C_1 and C_2, that compute errors over a sequence of length N, with κ_1 and κ_2 serving as scalar factors to weigh the rotation and translation quantities. The loss function C_1, shown in (19), guides the network to learn the relative pose measurements. We found that applying this intermediary training signal is crucial to convergence of the network early in the training process, as it is difficult to train recurrent networks with random initialization of weights.

$$\begin{gathered} C_1 = \sum_{k=1}^{N} \kappa_1 \mathbf{e}_{\boldsymbol{\phi}}^\top \mathbf{e}_{\boldsymbol{\phi}} + \mathbf{e}_{\mathbf{r}}^\top \mathbf{e}_{\mathbf{r}} \\ \mathbf{e}_{\boldsymbol{\phi}} = \tilde{\boldsymbol{\phi}}_{r_k v_{k+1}} - \boldsymbol{\phi}_{r_k v_{k+1}} \quad \mathbf{e}_{\mathbf{r}} = \tilde{\mathbf{r}}_{r_k}^{v_{k+1} r_k} - \mathbf{r}_{r_k}^{v_{k+1} r_k} \end{gathered} \tag{19}$$

The loss function C_2 is applied to minimize the errors on estimated inertial poses $\hat{\mathbf{C}}_{r_k i}$ and $\hat{\mathbf{r}}_{r_k}^{i r_k}$. The inertial pose estimates are affected by the measurement covariance $\mathbf{R}_k$ in addition to the quality of the measurements, thus C_2 encourages the network to learn a covariance that will produce the optimal estimates. In contrast to [5] and [20], which applied Mean Squared Error (MSE) on quaternions, we used Frobenius norm on the difference between $\hat{\mathbf{C}}_{v_k i}^T \mathbf{C}_{v_k i}$ and identity matrix as the rotation error metric. This has the advantage of preventing quaternion flips in early stages of training, when the errors tend to be large. Moreover, $\mathbf{I} - \hat{\mathbf{C}}_{v_k i}^T \mathbf{C}_{v_k i}$ approximates $\ln(\hat{\mathbf{C}}_{v_k i}^T \mathbf{C}_{v_k i})$ when $\hat{\mathbf{C}}_{v_k i}$ is close to $\mathbf{C}_{v_k i}$, providing a valid distance measure while circumventing the differentiability issue of the log mapping near π. We also implemented the option to train IMU covariance $\mathbf{Q}$ from (9) through backpropagation, but we found the covariances easily overfit on training data, producing worse results on evaluation.

$$C_2 = \sum_{k=1}^{N} \left\| \hat{\mathbf{r}}_{v_k}^{i v_k} - \mathbf{r}_{v_k}^{i v_k} \right\|_2^2 + \kappa_2 \left\| \mathbf{I} - \hat{\mathbf{C}}_{v_k i}^T \mathbf{C}_{v_k i} \right\|_F^2 \tag{20}$$

During training, we break down long sequences into short sub-sequences of 32 timesteps with ten timesteps separating them from each other. We randomly sample sub-sequences and combine them into a single batch of size 16. Instead of initializing all LSTM states at zero for each sub-sequence, the final LSTM states at the end of each sub-sequence are kept after each forward pass of the network and used for initialization in the following temporally adjacent sub-sequence. To combat overfitting, we applied data augmentation techniques such as random noise in brightness, contrast, saturation and hue. In addition, we also found that using left-right flips and travelling backwards in time helps considerably with training DeepVO. However, simulating flipped IMU data requires a sequence of transformations and corrections for gravity, which leads to error accumulation that reduces odometry accuracy. Thus, C_2 is only applied for non-flipped sub-sequences, while C_1 is applied on all augmented sub-sequences.

V. Experiments

We tested the proposed method on the KITTI dataset [9] and the EuRoC drone dataset [10], and compared to other relevant state-of-the-art methods with metrics frequently used on the specific dataset. KITTI and EuRoC datasets are small relative to typical deep learning datasets. To obtain the best possible results with the available training data, we evaluate on one sequence and use all other available sequences for training. We trained the model until overfitting occurred and selected the model with the lowest evaluation error.

A. KITTI Dataset

The KITTI odometry dataset [9] features an outdoor driving environment on urban roads with mostly planar 3-DoF motion. It provides camera data at 10Hz, and IMU data at 100Hz. The odometry dataset has 22 sequences, however only sequences 00 to 10 (03 not available) have corresponding raw data with IMU measurements. We also excluded sequences 00, 02, and 05 from evaluation due to missing IMU data for a few seconds at multiple points in time, but we kept the valid portions of the sequence for training. The standard KITTI evaluation metric is used, which computes the average translation error (%) and rotation error (deg/100m) per unit of distance travel, evaluated at 100, 200, ..., 800 meters. We compare the four methods:

- **Vision-Only**: sub-component of the network using just vision, training for the relative pose measurements and composing the relative poses outputs;

Table I
COMPARISON OF ACCURACY ON THE KITTI DATASET

Seq.	IMU-Only		Vision-Only		ORB+MSF		Proposed	
	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}
01 (2.5km)	4.46	**0.078**	32.02	1.914	**2.03**	0.199	25.24	0.159
04 (0.4km)	1.51	**0.004**	2.29	1.168	0.90	0.094	**0.34**	0.007
06 (1.2km)	6.38	**0.147**	7.86	3.556	**1.51**	0.269	2.14	0.156
07 (0.7km)	14.13	**0.263**	3.72	3.986	1.73	0.478	**0.93**	0.305
08 (3.0km)	68.64	**0.225**	5.33	1.511	1.30	0.319	**1.23**	0.255
09 (1.7km)	29.96	**0.213**	6.78	2.232	1.38	0.276	**1.09**	0.253
10 (0.9km)	14.05	**0.203**	7.23	2.408	1.40	0.316	**1.17**	0.248
Ave.	44.47	**0.212**	5.95	2.124	1.37	0.313	**1.27**	0.244

[1] Unit for t_{err} is [%], and unit for r_{err} is [°/100m]
[2] Seq. 01 is excluded from average since it is not well represented by other sequences during training, explained further in Section V-C

- **IMU-Only**: sub-component of the network using just IMU data, integrating IMU measurements using the EKF prediction step described in Section III-B;
- **ORB+MSF**: loosely-coupled fusion of stereo ORB-SLAM [21] keyframe poses with IMU using the MSF [11] framework. We attempted to use monocular ORB-SLAM poses, but the results were significantly worse due to rapid scale drift of monocular VO on KITTI;
- **VINet** [3]: End-to-end learning of VIO described in Section II-B. It is only tested on seq. 10 and trained on 0-9, and we reconstructed the boxplot from [3] to the best of our ability for comparison, shown in Figure 4;
- **Proposed**: our proposed method of training VIO with a robo-centric EKF end-to-end;

We also sought to compare with tightly-coupled VIO methods such as [12], and [2], however, their open-sourced code do not work well with KITTI.

For all methods, ground truth velocity and gravity are used for initialization with all initial biases set to zero. We resized images to 320×96 pixels to fit onto a single Titan Xp GPU. The compiled numerical values are shown in Table I with Figure 2, 3 showing the trajectory in the XY plane. From these results, we can summarize that our method:

- significantly improves upon the translation error of Vision-Only and IMU-Only implementations on almost all sequences, while IMU-Only has the lowest orientation error of all methods;
- outperforms ORB+MSF on all sequences for orientation and most sequences for translation;
- has overall lower error than VINet, especially when evaluated on longer distances;

B. EuRoC Dataset

We tested the proposed system on the EuRoC drone dataset [10], which features a drone flying in indoor environments with full 6-DoF motion. It provides hardware synchronized camera and IMU data at 20Hz and 200 Hz, respectively. The dataset contains a total of 11 sequences at three locations. We omit the sequence V2_03 due to issues with image data not arriving at the expected rates. The image data is downsampled to 10Hz and the IMU

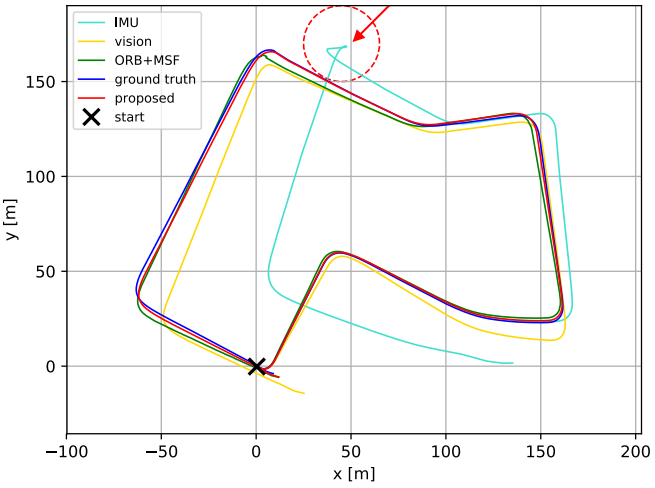

Figure 2. KITTI seq. 07 trajectory

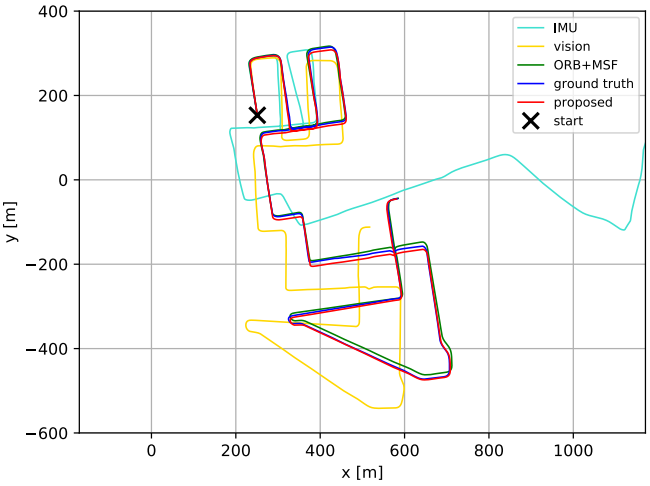

Figure 3. KITTI seq. 08 trajectory

data to 100Hz. For error metrics, we adopt the widely used RMSE absolute translation error (meters) after aligning the trajectories. In addition to Vision-Only and our proposed method, we compare two more methods:

- **SVO+MSF**: loosely-coupled fusion of SVO [22] outputs with IMU using the MSF [11] framework, results are retrieved from [23];
- **VINS-Mono**: a tightly-coupled VIO method, we disabled loop closure for VIO only comparison;

Comparison to IMU-only is omitted since EuRoC IMUs are much noisier than KITTI and contain significant biases, thus results would diverge immediately without corrections. Since not all EuRoC sequences have ground truth available when the drone is stationary, we retrieved the initial gravity by solving a linear least-squares problem using ground truth and accelerometer measurements. We used ground truth to initialize velocity, averaged gyro measurements in stationary frames for initial gyroscope bias, and set initial accelerometer bias to zero. Images are resized to 235×150

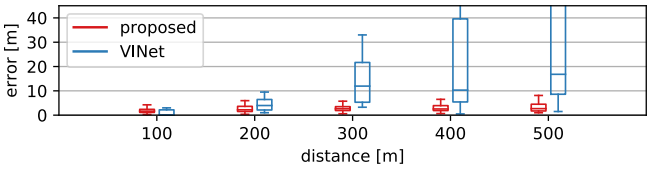

Figure 4. Trans. error on KITTI seq. 10 of our method compared to VINet

Table II
COMPARISON OF ACCURACY ON THE EUROC DATASET

Seq.	Vision-Only	SVO+MSF	VINS-Mono	Proposed
MH_01	1.25	**0.14**	0.16	1.17
MH_02	1.83	0.20	**0.18**	1.56
MH_03	2.47	0.48	**0.20**	1.89
MH_04	2.25	1.38	**0.35**	2.12
MH_05	2.62	0.51	**0.30**	1.96
V1_01	1.66	0.40	**0.09**	2.07
V1_02	1.17	0.63	**0.11**	2.20
V1_03	1.33	×	**0.19**	2.83
V2_01	1.16	0.20	**0.09**	1.49
V2_02	1.84	0.37	**0.16**	2.22

[1] Units in meters
[2] × indicates sequence not completed

pixels with the same batch size and sub-sequence length as used for the KITTI dataset. We summarize the results in Table II. We observe the state of the art classical methods have higher performance than our proposed method, and the causes are further elaborated in Section V-C.

C. Discussion

Our proposed method offers competitive performance on the KITTI dataset. From the results, we can observe our method works significantly better than using just the Vision-Only or IMU-Only sub-components of our method. Vision-Only struggles with accurately estimating large rotations, which can be seen from the less-than-perfect right angles turns in Figure 3, but it is relatively consistent on overall translation. With IMU-Only, the position estimation from triple integration of accelerometer and gyroscope measurements drifts rapidly over time, observed in Figure 2 and Figure 3. In particular, as circled in Figure 2, we can see a dramatic position drift when the vehicles stop for a few seconds, which our method does not exhibit. Also, the IMU-Only results have the lowest orientation error among all methods, which can be attributed to the fact that the gyroscope on the KITTI IMU has low noise and biases. This implies the majority of IMU-Only drifts are the result of integrating noisier accelerometer measurements, leading to increasingly inaccurate velocity estimates, which then, in turn, produce diverging position estimates. Fusing relative pose measurements with consistent displacements effectively corrects the accumulated errors in velocity, significantly improving position estimates.

Comparing to ORB+MSF, our method obtained lower orientation error on all sequences and lower translation error on sequences 04, 07, 08, 09, and 10. The performance improvements can be partly attributed to our system's ability

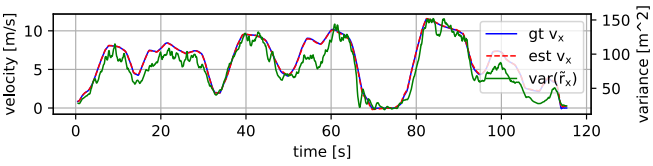

Figure 5. The forward velocity ($\mathbf{v}_{v_\tau}^{v_\tau i}$) and learned variance of forward displacement ($\tilde{\mathbf{r}}_{r_k}^{v_{k+1} r_k}$) on KITTI seq. 07

to learn relative pose measurements in the vehicle frame directly from ground truth, implicitly learning the underlying sensor calibrations, which circumvents any errors that might accumulate from intrinsic and extrinsic calibrations. Furthermore, while C_1 guides the network to learn accurate measurements, C_2 encourages the learning of covariances that allow the EKF to fuse these measurements effectively. This is shown in Figure 5, where we observe the network has learned to output the uncertainty of forward displacement (x component of $\tilde{\mathbf{r}}_{r_k}^{v_{k+1} r_k}$) in proportion to the forward velocity, and the resulting velocity estimate closely follows the ground truth. For sequence 01, our translation error is much higher than ORB+MSF. This is because sequence 01 is a highway scene with the vehicle travelling up to 95 km/h. Unfortunately, the vehicle in all other sequences travelled much slower, which is not representative of 01. We saw the network consistently underestimate the travelled distance during the high-speed portion, but was able to produce better estimates for the slower off-ramp section. The imperfect estimation of scale can be observed on all sequences to varying degrees, resulting in varying performance. The problem could be alleviated through including scale as an additional EKF state as well as having more representative data.

On the EuRoC dataset, classical VIO methods worked significantly better than our method. This occurred for a few reasons. First, unlike the KITTI IMU which is quite precise, the IMU used in the EuRoC dataset is much nosier and contains significant accelerometer and gyroscope bias. Gyroscope bias is relatively easy to recover under stationary circumstances, but accelerometer bias is difficult to estimate on initialization due to the coupling of gravity vector, which means it must rely on additional visual measurements for correction. However, the DeepVO architecture in combination with limited amount of training data was unable to produce the accurate relative pose measurements needed to update the EKF in a drone scenario with 6-DoF motion. This is evident from Vision-Only results which is up to an order of magnitude worse than the results reported in monocular SVO [22] when properly scaled. Thus, it becomes difficult for our method effectively learn and perform VIO estimation when sensing from both IMU and vision are of insufficient quality and are unable to complement each other.

VI. CONCLUSION

This paper proposed an end-to-end trainable method for Visual Inertial Odometry. Our approach used a robo-centric

EKF formulation which estimates vehicle states in a local frame of reference and updates inertial states through a composition step. The EKF propagates local pose using IMU, and performs EKF correction using relative pose measurements obtained from a deep network. The proposed system is fully differentiable which allows end-to-end training of our system for overall improved performance. Our method was tested on the KITTI and EuRoc dataset. We demonstrated competitive performance on the KITTI dataset, while the effectiveness on the EuRoC dataset could improve with more training data, better system initialization, and IMU calibration. For future work, we would like to experiment with more effective measurement models, more efficient deep architectures for visual measurements, and incorporating IMU data directly in the learning process alongside the EKF.

REFERENCES

[1] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[2] 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.

[3] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI, 2017.

[4] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation*, May 2017, pp. 2043–2050.

[5] S. Wang, R. Clark, H. Wen, and A. Trigoni, "End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks," *The International Journal of Robotics Research*, vol. 37, pp. 513 – 542, 2018.

[6] P. Karkus, D. Hsu, and W. S. Lee, "Particle filter networks with application to visual localization," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 169–178.

[7] R. Jonschkowski, D. Rastogi, and O. Brock, "Differentiable Particle Filters: End-to-End Learning with Algorithmic Priors," in *Proceedings of Robotics: Science and Systems*, 2018.

[8] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, "Backprop kf: Learning discriminative deterministic state estimators," in *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., 2016, pp. 4376–4384.

[9] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition*, 2012.

[10] 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.

[11] 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 *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 3923–3929.

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

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

[14] L. von Stumberg, V. C. Usenko, and D. Cremers, "Direct sparse visual-inertial odometry using dynamic marginalization," *2018 IEEE International Conference on Robotics and Automation*, pp. 2510–2517, 2018.

[15] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *2015 IEEE International Conference on Computer Vision*, Dec 2015, pp. 2758–2766.

[16] R. Clark, M. Bloesch, J. Czarnowski, S. Leutenegger, and A. J. Davison, "Learning to solve nonlinear least squares for monocular stereo," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[17] M. Brossard, A. Barrau, and S. Bonnabel, "AI-IMU dead-reckoning," *CoRR*, vol. abs/1904.06064, 2019.

[18] R. Jonschkowski and O. Brock, "End-to-end learnable histogram filters," in *Workshop on Deep Learning for Action and Interaction at NIPS*, December 2016.

[19] Z. Huai and G. Huang, "Robocentric visual-inertial odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2018, pp. 6319–6326.

[20] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *2015 IEEE International Conference on Computer Vision*, Dec 2015, pp. 2938–2946.

[21] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct 2017.

[22] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "Svo: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, April 2017.

[23] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *2018 IEEE International Conference on Robotics and Automation*, May 2018, pp. 2502–2509.