Towards End to End Learning of VIO using Algorithmic Prior

Chunshang Li

August 14, 2019

1 Motivation

Inertial navigation systems have been used widely to for estimating poses in 3D space. They are particularly useful for state estimation in GPS-denied situations such as indoor, underground, and urban jungles. A typical inertial navigation system uses a Inertial-Measurement Unit (IMU) that integrates high rate accelerometer and gyroscope data to obtain change in position, velocity, and rotation. The estimated position is the result of three nested integration, thus is prone to noise and biases which will cause the estimate to drift rapid. This makes pure inertial navigation useless but for application that can afford high-end IMU. In recent years, there has been a growing effort to integrate visual information from the camera with IMU's. These Visual Inertial Systems (VINS) enables observation of biases and able to correct for errors accumulated during the integration of IMU data. VINS systems has shown tremendous success due to its low cost nature and abundance of camera sensors and Micro-Electro-Mechanical Systems (MEMS) IMU's. VINS systems are widely used on mobiles robots, drones, and mobile phones.

Current VINS systems rely on state of the art visual SLAM methods with a typical pipeline consists of feature detection, feature matching, outlier rejection (e.g. RANSAC), 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 feature mismatches could easily cause the optimization process to diverge. In addition, VINS systems requires manual work for parameter tuning and calibration. Each stage of the visual pipeline has parameters that needs to be tuned which is a tedious process, and performance improve in one stage of the pipeline does not necessarily translate into overall performance improvement. Noise parameters such as IMU covariance and feature pixel location covariances can be in theory measured physically, but they are often treated as more parameters that needs to be tuned by hand. Also, VINS systems will not work without proper calibrations. Accurate camera intrinsic calibrations requires the use of fiducial to be captured across the image, and camera-IMU extrinsic calibration requires data collection with sufficient system excitation. Both of these require large amount of careful work to ensure the calibrations are correct.

The rise in deep learning provided the tools to process high dimensional data such as an image directly rather than reducing dimensionality through techniques such as feature extraction. This allows us to learn directly state estimation from data in an end-to-end manner as demonstrated from previous works [7], [41], and [42]. Instead of reducing the dimensionality of the inputs through techniques such as feature extraction, the deep network can be used to learn not only the local features but high level features as well. Additionally, the deep network may learn additional models that may improve state estimation such as rejecting dynamic objects and the scene, camera exposure changes, and scale of objects in the scene. On top of this, using learning methods enable the system to automatically gain knowledge during tests and deployment in the future. In contrast to images, IMU data are low dimensional and has well understood noise models that are grounded in physics. It is not necessary to use a full deep network to learn these physical models. Incorporating known state estimation processes has been shown to improve data efficiency and reduce over-fitting [25], [24], [20]. In this work, we explore these ideas further, designing a deep network for Visual Inertial Odometry (VIO) that incorporates domain specific knowledge from VINS system, constraining the network to learn the difficult hard-to-model parts of the pipeline.

2 Related Works

2.1 Model-based Visual SLAM

There has been extensive work on monocular visual SLAM and odometry in the past few decades. These methods uses physical models such as camera projection, rigid body motion, multi-view geometry, and photometric consistency. These vision-based methods generally fall into two categories: feature-based methods and direct methods. Feature-based methods detects features from images, perform 2D-to-2D, and 3D-to-2D matching to estimate the motion of the camera as well as the position of the features. On the other hand, direct methods works directly to minimize photometric errors between images without explicitly extracting feature points [37]. Notable feature-based visual SLAM includes MonoSLAM [9], PTAM [26], and ORB-SLAM [32] [33]. Notable direct methods include DTAM [34], semi-dense visual odometry [14], LSD-SLAM [13], and DSO [12]. Hybrid methods such as SVO [17] also exists, it combines both aspects of feature-based and direct methods.

2.2 Model-based Visual Inertial SLAM

Fusing IMU with monocular visual SLAM is also an extensively researched topic in recent years. Fusion resolves scale ambiguity and improve estimates. Angular velocity from the gyroscope can be integrated to obtain change in orientation, and accelerometer data can be integrated to obtain change in velocity and position. However IMUs are prone to drifting due to noise, and additional sensors are required for corrections. IMU has been integrated with both feature-based and direct visual SLAM methods in the past to form VINS. Visual inertial approaches are further categorized into loosely-coupled and tightly-coupled methods. Loosely coupled approaches are modular in that they process visual and IMU data separately then fusing them together, often using an Extended Kalman Filter (EKF). Tightly-coupled method combines visual and inertial measurements into a single estimation problem.

Notable methods that use a loosely-coupled approach are [30], [44], [45], and [43]. [44] and [45] treats the visual pipeline as a black box generic pose sensor, used an Error State Kalman Filter (ESKF) formulation that estimates the scale drift of position measurements provided by PTAM. The method also detects failures in the visual SLAM and allow the system to reinitialize. Furthermore, [45] performed an observability analysis showing the system is unobservable in global translation and yaw which is consistent with the observations made in [21] for a tightly-coupled method. Following up to [44], [43] formulated a method that updates the ESKF by converting the camera to a metric speed sensor through optical flow. The Modular Multi-Sensor Framework (MSF) proposed in [30] was used as the state estimation module in [15] fusing estimates from SVO and IMU to land a quadrotor on a moving platform.

Notable tightly-coupled feature-based methods include MSCKF [31], OKVIS [28], VINS-MONO [36], and [16]. MSCKF is a popular approach based on the ESKF. It maintains a window of poses to estimate, optimizes features using bundle adjustment, and performs measurement update through the geometric constraint between features and poses without the need to include features as part of the EKF state vector. OKVIS is the first to implement visual inertial fusion using nonlinear optimization. It performs optimization on a sliding window of keyframes and landmark features using reprojection and IMU errors. VINS-MONO and [16] improved computational on top of OKVIS by using IMU pre-integration in the visual inertial pipeline. Some examples of tightly-coupled direct methods are ROVIO [2], and VI-DSO [40]. ROVIO is based on an Iterated EKF, it tracks FAST corner features, and

performs measurement update using photometric error. VI-DSO minimizes photometric and IMU errors through nonlinear optimization.

Our work uses the loosely-coupled approach and have a neural network providing the visual measurements to correct for IMU drifts. This allows our method to be trained end to end in a data-driven fashion.

2.3 SLAM with Deep Learning

Deep Learning methods have demonstrated great success in field such as classification, object detection, and natural language processing. Recently, there had been a surge of interest in applying deep learning methods to SLAM tasks. [27] proposed a CNN architecture to estimate visual odometry. It achieves so by discretizing direction and velocity changes using Softmax functions, turning it into a classification problem. [8] takes optical flow as input and directly regress position and orientation displacement using CNN. This is further expanded on LS-VO where an auto-encoder is used to produce optical flow which is then used to estimate odometry. [46] proposed a visual odometry network that computes 3D optical flow computed from CNN-produced 2D optical flow and depth estimation. The depth and odometry estimation are further combined to produce dense 3D mapping. DeepVO [41] is the first to use Recurrent Convolutional Neural Network (RCNN) to regresss poses from a sequence of images. The method takes a pair of images, feed it through FlowNet [11], LSTM, and fully connected network to produce a 6DoF pose. [42] extended the work on DeepVO by incorporating uncertainty estimation through a loss function based on maximum likelihood. In addition, a SE(3) layer is proposed to compound relative poses in the neural network. [42] demonstrated deep learning methods are more resilient to rapid motion, motion blur, exposure change, and rolling shutter effects. This is further validated in Deep EndoVO [39] where the RCNN architecture is applied to localization of an endoscopic capsule robot in the gastrointestinal tract. VINet applied learning to visual inertial odometry in [7]. It uses LSTMs to process high-rate IMU data which are concatenated with resulting feature vector from images processed through CNN, which is then fed through a fully connected network producing relative poses. The relative poses are further concatenated through a SE(3) layer similar to [42]. VINet produced worse estimation results in comparison to OKVIS, however it was shown that the approach degrades more gracefully with inaccurate camera-IMU calibration. Although VINet demostrated promising results for a deep learning approach to visual inertial odometry, the application of LSTM to learn IMU dynamics is unnecessary. Unlike vision based method which is affected by hard-to-model environmental conditions, IMUs are much less sensitive to the environment and its kinematics are well known and modelled for. Our work combines model based IMU state estimation with deep learning visual processing in attempt to get best of both worlds.

Data-driven deep learning approaches have shown considerable progress in SLAM tasks. There has been growing interest in using hybrid systems that combines elements of model-based and deep learning approaches. CNN-SLAM [38] used Convolutional Neural Networks (CNN) to predict depth of a scene with absolute scale, this is fused with monocular dense reconstruction to produce a scale accurate map. DPC-Net [35] used CNN to predict small, low-rate corrections to a stereo visual odometry pipeline, which is then used in pose graph relaxation to improve estimates. It is demonstrated DPC-Net can reduce the effects of estimator biases, poor calibrations, and challenging environmental conditions. CodeSLAM proposed in [3] learns a compact representation of scene depth that can be used in the optimization of dense visual SLAM. LS-Net [6] uses LSTM to learn nonlinear least square

optimizer updates. [20], [23], [25], and [24] introduces the use of algorithmic priors in deep learning for robotic tasks. [20] estimates visual odometry by formulating an EKF with 2D poses and velocities as states and velocity measurement obtained from images processed through an CNN. The EKF exists as part of the network and the network is trained end-to-end using ground truth labels. An end-to-end histogram filter is proposed in [23] which demonstrated its effectiveness under simple localization tasks. [25] and [24] concurrently proposed particle filter networks that enables the learning of motion and measurement model in an end-to-end manner. For experiments on visual odometry, [24] formulated the states in a similar way as [20] and showed it outperformed [20]. In these works, domain specific knowledge in robotics in the form of algorithmic priors is incorporated as part of deep network. Experiments 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 shown to improve performance as opposed to training each subsystems individually. Our work follows these inspirations and uses a robocentric ESKF as part of the network. ESKF is a well known variant of the Kalman Filter and has been applied extensively in IMU sensor fusion, and the robocentric formulation allows us to incorporate relative measurements. Our work also tests the idea in a more realistic setting, formulating a full 6DoF state estimation problem and aligning it to existing SLAM literature.

3 Background

3.1 Coordinate Frames and Transformations

There exists a wide range of notations for representing transformations in the 3D space. In this work, we adopt notation from [1]. A vector or matrices are expressed with **bold** fonts. All 3D quantities are expressed with respect to coordinate frames. All coordinate frames used in this work uses the right-handed convention. Superscripts and subscripts are used to decorate the vector and matrix quantities, giving us the knowledge of which coordinate frames these quantities relates to. As shown in Figure 1, displacement of frame $\mathcal{F}_c$ with respects to frame $\mathcal{F}_b$ expressed in $\mathcal{F}_a$, and $\mathbf{C}_{ab}$ expresses the rotation of frame $\mathcal{F}_b$ with respect to frame $\mathcal{F}_a$.

A rigid body transformation which is comprised of a translation and rotation in 3D space, it takes a point expressed in one 3D Cartesian coordinate frame into another. All coordinate systems presented here follows the right-hand convention. Both translation and translation in 3D space have 3 degrees of freedom, thus a transformation has a total of 6 degrees of freedom. In relation to Figure 2, Translation $\mathbf{r}_a^{ba}$ is a vector quantity, rotation $\mathbf{C}_{ab}$ is an element of the Special Orthogonal Lie Group, $\mathbb{SO}(3)$, and together they form $\mathbf{T}_{ab}$ which is an element of the Special Euclidean Lie group $\mathbb{SE}(3)$. $\mathbf{T}_{ab}$ describes a transformation or pose of frame $\mathcal{F}_b$ with respect to frame $\mathcal{F}_a$. This is described more precisely in Equation 1.

$$
\begin{aligned}
\mathbf{r}_a^{ba} &\in \mathbb{R}^3 \\
\mathbf{C}_{ab} \in \mathbb{SO}(3) &= \{\mathbf{C}_{ab} \in \mathbb{R}^{3\times 3} | \mathbf{C}_{ab}^T \mathbf{C}_{ab} = \mathbf{1}, \det \mathbf{C}_{ab} = 1\} \\
\mathbf{T}_{ab} \in \mathbb{SE}(3) &= \{\mathbf{T}_{ab} = \begin{bmatrix} \mathbf{C}_{ab} & \mathbf{r}_a^{ba} \\ \mathbf{0}_{3\times 1} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}\}
\end{aligned}
\tag{1}
$$

With translation, rotation, and transformations defined, we can now manipulate these quantities through standard matrix operations. For example, a vector $\mathbf{r}_b^{cb}$ expressed in frame

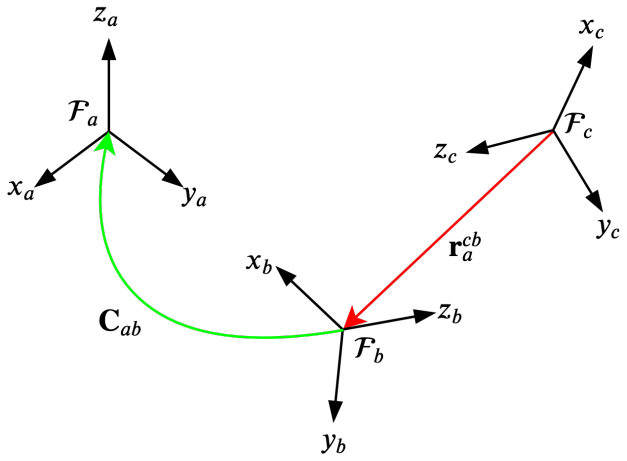


Figure 1: Rotation and translation of coordinate frames

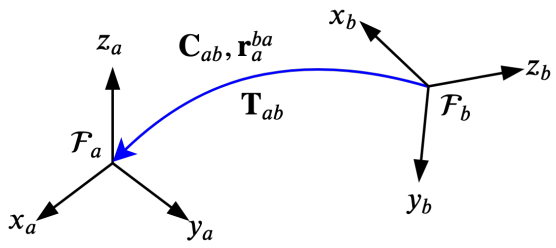


Figure 2: Transformation of coordinate frames

$\mathcal{F}_b$ can be changed to be expressed in frame $\mathcal{F}_a$ shown in Equation 2. A chain of rotations and transforms can be concatenated through simple matrix multiplications shown in 3. The subscripts are useful in this case to aid in keeping track of coordinates involved.

$$\mathbf{r}_a^{cb} = \mathbf{C}_{ab}\mathbf{r}_b^{cb} \tag{2}$$

$$\begin{aligned} \mathbf{C}_{ad} &= \mathbf{C}_{ab}\mathbf{C}_{bc}\mathbf{C}_{cd} \\ \mathbf{T}_{ad} &= \mathbf{T}_{ab}\mathbf{T}_{bc}\mathbf{T}_{cd} \end{aligned} \tag{3}$$

We can also transform a point in frame $\mathcal{F}_b$ into $\mathcal{F}_a$ shown in Equation 4. Note that since $\mathbf{T}_{ab}$ is an 4×4 matrix, we need to augment $\mathbf{r}_b^{cb} \in \mathbb{R}^3$ forming $\begin{bmatrix} \mathbf{r}_b^{cb} \\ 1 \end{bmatrix} \in \mathbb{R}^4$. The augmented vector is known as a homogeneous point representation. With some abuse of notation, $\mathbf{r}_b^{cb}$ could either be in homogeneous representation or not, whichever one makes the matrix dimension correct.

$$\mathbf{r}_a^{ca} = \mathbf{T}_{ab}\mathbf{r}_b^{cb} = \mathbf{R}_{ab}\mathbf{r}_b^{cb} + \mathbf{r}_a^{ba} \tag{4}$$

3.1.1 Optimizing on SO(3)

Rotations defined in 1 use a 3×3 matrix as representation. There exists a few other representations for rotations including quaternions (4 parameters), axis-angle (4 parameters), Euler angles (3 parameters), and rotation vectors (3 parameters). Rotation representation with 3 parameters are the most compact having a equal number of parameters as there are degrees of freedom. However, they suffer from singularity problems and combining these rotations in 3D isn't straight forward. Representation that are over-parameterized, with parameter counts of over 3, require additional constraints. For example, rotation matrices must be orthogonal and quaternions must have unit length. These constraints need to be enforced during the optimization process which is less than desirable. For over-parameterized rotation, we almost work exclusively with rotation matrices, but the same derivation results still holds for right-handed Hamiltonian unit quaternions.

A Lie group is a smooth manifold that with an operator that satisfies 4 properties: closure, associativity, identity, and invertibility as elaborated in [1]. Each Lie group is also associated with a Lie algebra which is a vector space over some field with a operator known as the Lie bracket, together they must satisfy the properties of closure, bilinearity, alternativity, and Jacobi identity further elaborated in [1]. Rotations, being an element of the $\mathbb{SO}(3)$ group, by definition there exist a Lie algebra $\mathfrak{so}(3)$ as defined in Equation 5, where $(\cdot)^\wedge$ is the skew-symmetric operator.

$$\begin{aligned} \boldsymbol{\phi}^\wedge \in \mathfrak{so}(3) &= \{\boldsymbol{\phi}^\wedge \in \mathbb{R}^{3\times 3} | \boldsymbol{\phi} \in \mathbb{R}^3\} \\ \boldsymbol{\phi}^\wedge = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} &= \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \end{aligned} \tag{5}$$

We can map an element of $\mathfrak{so}(3)$ to an element of $\mathbb{SO}(3)$ through the exponential mapping shown in Equation 6. In reverse, we can map an element of $\mathbb{SO}(3)$ to an element of $\mathfrak{so}(3)$ through the logarithmic mapping shown in Equation 7, where $\boldsymbol{\phi} = \phi \mathbf{a}$. Equation 6 and 7 will result in zero division errors when ϕ is near zero, approximations are required in those cases. In addition, 7 is discontinuous when $|\boldsymbol{\phi}| \to \pi$ due to angle wrapping. Furthermore, the mapping from $\mathfrak{so}(3)$ to $\mathbb{SO}(3)$ is a many-to-one mapping, the logarithmic mapping constrains the solution of $\mathfrak{so}(3)$ between $-\pi$ and π. Also note that exponential mapping and logarithmic mapping exists for $\mathbb{SE}(3)$ as well, however this is not further elaborated on since it is not used in this thesis.

$$\mathbf{C} = \exp(\boldsymbol{\phi}) = \sum_{n=0}^{\infty} \frac{1}{n!} (\boldsymbol{\phi}^\wedge)^n = \cos\phi \mathbf{1} + (1 - \cos\phi)\mathbf{a}\mathbf{a}^T + \sin\phi \mathbf{a}^\wedge \tag{6}$$

$$\begin{aligned} \boldsymbol{\phi} &= \ln(\mathbf{C})^\vee \\ \phi &= \arccos(\frac{\operatorname{trace}\mathbf{C} - 1}{2}) \\ \mathbf{a} &= \frac{(\mathbf{C} - \mathbf{C}^T)^\vee}{2\sin\phi} \end{aligned} \tag{7}$$

Elements of $\mathfrak{so}(3)$ defines a tangent plane at the identity element of $\mathbb{SO}(3)$. Instead of optimizing on the group directly, we optimize on the tangent space at the given group element in lie algebra. The is done through applying a perturbation on the $\mathfrak{so}(3)$ using $\mathfrak{so}(3)$ as shown in Equation 8. Note that applying the perturbation as a matrix multiplication on

the left or the right is chosen arbitrarily and does not affect the final result as long as the same perturbation is used consistently. Since elements of $\mathfrak{so}(3)$ have a minimum parameterization that exists in the vector space, we can optimize without enforcing additional constraints on the estimated quantities. When deriving the Jacobian with respect to the rotational quantities for optimization, the differentiation is performed with respect to the perturbation as opposed to the group elements.

$$\mathbf{C} = \bar{\mathbf{C}}\delta\mathbf{C} = \bar{\mathbf{C}}\exp(\boldsymbol{\phi}^{\wedge}) \approx \bar{\mathbf{C}}(\mathbf{1} + \delta\boldsymbol{\phi}^{\wedge}) \tag{8}$$

3.2 Robocentric Error State Kalman Filter

3.2.1 Overview

The Robo-centric Error State Kalman Filter (RC-KF) formulation is adopted from Robo-centric visual inertial odometry (VIO) [22]. In contrast to the other common formulation that have states in respect to earth referenced inertial frames, the states of RC-KF are with respect to a reference frame that moves along with the robot. This formulation allow us to fuse measurements that are relative in nature as opposed to using absolute measurements that are referenced in the inertial frame. This formulation is natural for the case of VIO since by definition of odometry, there will be not any measurements in the inertial frame of reference without accumulation of uncertainty. RC-KF is also an error state KF. An error state KF breaks states into nominal and error states. The nominal states keeps track of large signal dynamics. The error states keeps track of the errors caused by noise, and they are formulated as perturbations of the nominal states. This means the error states always operate at around zero which reduces errors which is far away from any singularities and reduces errors associated with linearization and discretization.

RC-KF uses a 3 references frames in its estimation process: inertial frame $\mathcal{F}_i$, reference frame at time k $\mathcal{F}_{r_k}$, and the vehicle frame at time τ $\mathcal{F}_{v_\tau}$ as shown in Figure 3. IMU measurements typically are received at a higher frequency than images. Each time step k indicates a point in time where an image is received. Time in between images at t_k and t_{k+1} are indicated with t_τ, $t_\tau + 1$, ... At every timestep t_τ a new piece of IMU data is received.

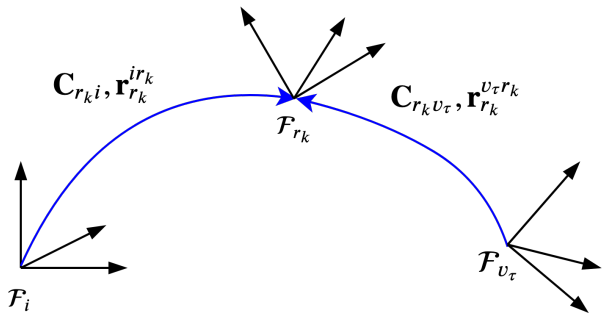


Figure 3: Robocentric VIO reference frames

During the prediction step of the RC-KF at time k, it integrates the IMU in frame $\mathcal{F}_{r_k}$

to obtain the nominal states and error state covariance (Figure 3). Then given a new image at time t_{k+1}, measurements are extracted, and the measurement model is used to produce a best estimation of error state in frame $\mathcal{F}_{r_k}$. The estimated error state is injected into the nominal state and reset to zero for the next update. Finally, an additional step is required to update the reference frame of the states from $\mathcal{F}_{r_k}$ to $\mathcal{F}_{r_{k+1}}$. The process then repeats for time t_{k+1}. The process is outlined in Figure 4.

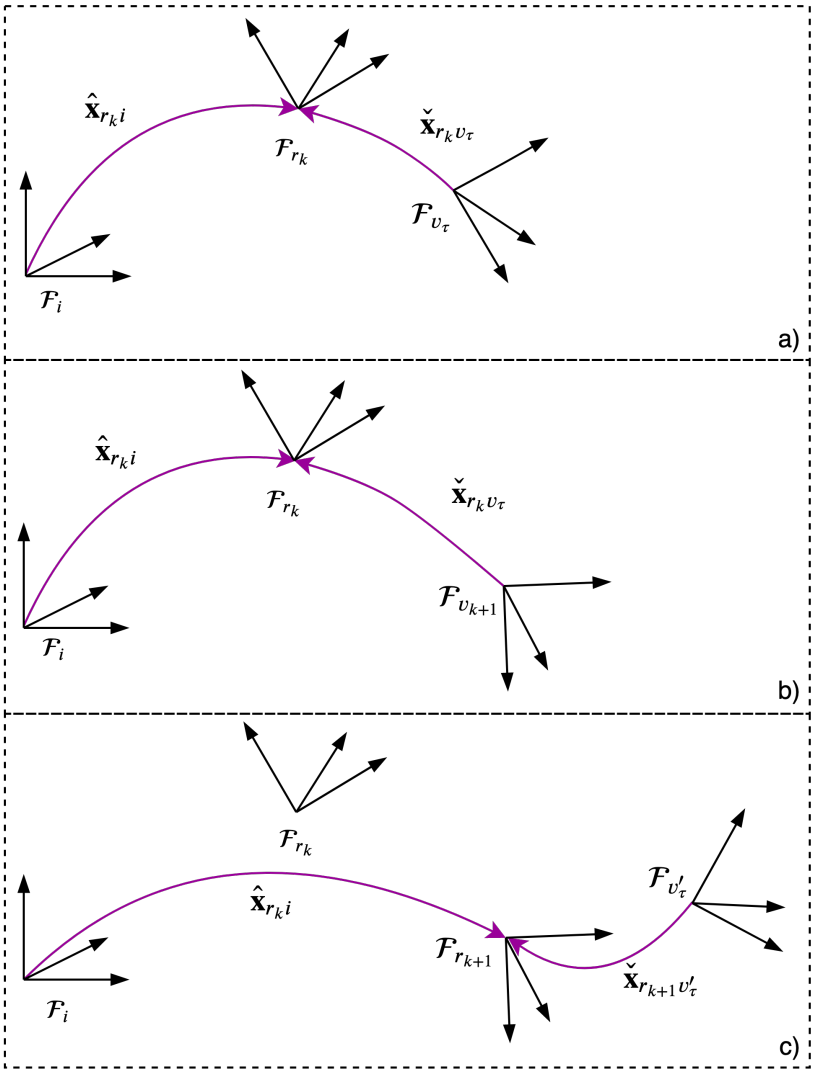


Figure 4: Robocentric VIO estimation process. a) shows the vehicle in the KF prediction step at time. b) shows the vehicle after the KF correction step given a new image. c) shows the the KF prediction step in the subsequent frame after the reference frame update is completed.

In [22], the JPL convention was used. This is a left-handed quaternion as opposed to

a right handed convention that is used more widely in the field of mobile robotics. In this work, we chose rotation matrices to represent rotations which is arguably easier to work with in absence of quaternion algebra. In addition, we present a simplified version of work from [22] since we do not keep track a sliding window of vehicle states, and we do not use a image features as our measurements. In terms of notations, $\bar{(\cdot)}$ is used to denote noisy quantities, $\check{(\cdot)}$ is used to denote predicted quantities in an KF, $\hat{(\cdot)}$ is used to denote corrected quantities in an KF.

3.2.2 States

Equation 9 shows the nominal states estimated by RC-KF. The states are separated into the global state $\mathbf{x}_{r_k i}$ and the vehicle frame $\mathbf{x}_{r_k v_\tau}$. We also assume the IMU frame is aligned with the vehicle frame. The states are are described as follows. Note that since frame r_k is not moving w.r.t. the inertial frame i at any time instant, $\mathbf{v}_{v_\tau}^{v_\tau i}$ is equivalent to $\mathbf{v}_{v_\tau}^{v_\tau r_k}$, for consistent we use $\mathbf{v}_{v_\tau}^{v_\tau i}$ in all of our derivations, same principle applies to acceleration and angular velocity.

- $[\mathbf{C}_{r_k i} \;\; \mathbf{r}_{r_k}^{i r_k}]$: the pose of the inertial frame $\mathcal{F}_i$ with respect to the reference frame F_{r_k}, the inertial frame is tracked as a feature in the reference frame F_{r_k}
- $\mathbf{g}_{r_k}$: the local gravity vector expressed reference frame F_{r_k}, it also implicitly encodes the roll and pitch of F_{r_k} relative to the earth
- $[\mathbf{C}_{r_k v_\tau} \;\; \mathbf{r}_{r_k}^{v_\tau r_k}]$: the relative vehicle pose of F_{v_τ} with respect to F_{r_k}
- $\mathbf{v}_{v_\tau}^{v_\tau i}$: the vehicle velocity expressed in the vehicle frame F_{v_τ}, this is always vehicle-centric
- $\mathbf{v}_{v_\tau}^{v_\tau i}$: gyroscope bias
- $\mathbf{b}_{a_\tau}$: accelerometer bias

$$\begin{aligned}\mathbf{x}_\tau &= [\mathbf{x}_{r_k i} \mid \mathbf{x}_{r_k v_\tau}] \\ &= [\mathbf{C}_{r_k i} \;\; \mathbf{r}_{r_k}^{i r_k} \;\; \mathbf{g}_{r_k} \mid \mathbf{C}_{r_k v_\tau} \;\; \mathbf{r}_{r_k}^{v_\tau r_k} \;\; \mathbf{v}_{v_\tau}^{v_\tau i} \;\; \mathbf{b}_{\omega_\tau} \;\; \mathbf{b}_{a_\tau}]\end{aligned} \tag{9}$$

The error states are defined in Equation 10. The perturbations are defined in 11, other than rotational quantities with their perturbations defined in $\mathfrak{so}(3)$ as shown in Equation 8, all other perturbation are defined in vectorspace where simple additions can be used.

$$\begin{aligned}\delta\mathbf{x}_\tau &= [\delta\mathbf{x}_{r_k i}^T \mid \delta\mathbf{x}_{r_k v_\tau}^T]^T \\ &= [\delta\boldsymbol{\phi}_{r_k i}^T \;\; \delta\mathbf{r}_{r_k}^{i r_k T} \;\; \delta\mathbf{g}_{r_k}^T \mid \delta\boldsymbol{\phi}_{r_k v_\tau}^T \;\; \delta\mathbf{r}_{r_k}^{v_\tau r_k T} \;\; \delta\mathbf{v}_{v_\tau}^{v_\tau i T} \;\; \delta\mathbf{b}_{\omega_\tau}^T \;\; \delta\mathbf{b}_{a_\tau}^T]^T\end{aligned} \tag{10}$$

$$\begin{aligned}\mathbf{C}_{r_k i} &= \bar{\mathbf{C}}_{r_k i} \exp(\delta\boldsymbol{\phi}_{r_k i}) \\ \mathbf{r}_{r_k}^{i r_k} &= \bar{\mathbf{r}}_{r_k}^{i r_k} + \delta\mathbf{r}_{r_k}^{i r_k} \\ \mathbf{g}_{r_k} &= \bar{\mathbf{g}}_{r_k} + \delta\mathbf{g}_{r_k} \\ \mathbf{C}_{r_k v_\tau} &= \bar{\mathbf{C}}_{r_k v_\tau} \exp(\delta\boldsymbol{\phi}_{r_k v_\tau}) \\ \mathbf{r}_{r_k}^{v_\tau r_k} &= \bar{\mathbf{r}}_{r_k}^{v_\tau r_k} + \delta\mathbf{r}_{r_k}^{v_\tau r_k} \\ \mathbf{v}_{v_\tau}^{v_\tau i} &= \bar{\mathbf{v}}_{v_\tau}^{v_\tau i} + \delta\mathbf{v}_{v_\tau}^{v_\tau i} \\ \mathbf{b}_{\omega_\tau} &= \bar{\mathbf{b}}_{\omega_\tau} + \delta\mathbf{b}_{\omega_\tau} \\ \mathbf{b}_{a_\tau} &= \bar{\mathbf{b}}_{a_\tau} + \delta\mathbf{b}_{a_\tau}\end{aligned} \tag{11}$$

3.2.3 IMU Measurement Model

$\mathbf{a}_m$ and $\boldsymbol{\omega}_m$ are the accelerometer and gyroscope measurements we would obtain directly from the IMU hardware. For the accelerometer measurement, it is the sum of the true acceleration $\mathbf{a}_{v_\tau}^{v_\tau i}$, the gravity vector rotated to the vehicle frame $\mathbf{C}_{v_\tau r_k}\mathbf{g}_{r_k}$, biases $\mathbf{b}_{a_\tau}$, and noise $\mathbf{n}_a$. For the gyroscope, it is the sum of the true angular velocity $\boldsymbol{\omega}_{v_\tau}^{v_\tau i}$, the bias $\mathbf{b}_{\omega_t}$, and noise $\mathbf{n}_\omega$. The equation in shown in 12. The accelerometer and gyroscope biases changes over time and are modeled as a random walk process with noise $\mathbf{n}_{b_a}$ and $\mathbf{n}_{b_w}$ respectively. We assume the noise are zero mean Gaussian as required by the KF.

$$\begin{aligned}\mathbf{a}_m &= \mathbf{a}_{v_\tau}^{v_\tau i} + \mathbf{C}_{v_\tau r_k}\mathbf{g}_{r_k} + \mathbf{b}_{a_\tau} + \mathbf{n}_a \\ \boldsymbol{\omega}_m &= \boldsymbol{\omega}_{v_\tau}^{v_\tau i} + \mathbf{b}_{\omega_t} + \mathbf{n}_\omega\end{aligned} \tag{12}$$

$$\begin{aligned}\mathbf{n}_a \sim \mathcal{N}(\mathbf{0}, \sigma_a^2\mathbf{I}), \mathbf{n}_\omega \sim \mathcal{N}(\mathbf{0}, \sigma_\omega^2\mathbf{I}) \\ \dot{\mathbf{b}}_{a_t} = \mathbf{n}_{b_a}, \dot{\mathbf{b}}_{\omega_t} = \mathbf{n}_{b_\omega} \\ \mathbf{n}_{b_a} \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{b}_a}^2\mathbf{I}), \mathbf{n}_{b_\omega} \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{b}_\omega}^2\mathbf{I})\end{aligned} \tag{13}$$

Rearranging 12 into 14 leads to the true acceleration and angular velocity which are the quantities that will be integrated to produce relative motion predictions.

$$\begin{aligned}\mathbf{a}_{v_\tau}^{v_\tau i} &= \mathbf{a}_m - \mathbf{C}_{v_\tau r_k}\mathbf{g}_{r_k} - \mathbf{b}_{a_\tau} - \mathbf{n}_a \\ \boldsymbol{\omega}_{v_\tau}^{v_\tau i} &= \boldsymbol{\omega}_m - \mathbf{b}_{\omega_t} - \mathbf{n}_\omega\end{aligned} \tag{14}$$

Substituting Equation 11 into Equation 14, we obtain Equation 15, we can describe acceleration and angular velocity in terms of their perturbations with respect to to the nominal values. This will be used later on to derive the continuous time error state process model.

$$\begin{aligned}\underbrace{\mathbf{a}_m - \bar{\mathbf{C}}_{r_k v_\tau}^T \bar{\mathbf{g}}_{r_k} - \bar{\mathbf{b}}_{a_\tau}}_{\bar{\mathbf{a}}_{v_\tau}^{v_\tau i}} &= \mathbf{a}_{v_\tau}^{v_\tau i} + \underbrace{\delta\mathbf{b}_{a_\tau} + \mathbf{n}_a + \bar{\mathbf{C}}_{r_k v_\tau}^T \delta\mathbf{g}_{r_k} - \delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge \bar{\mathbf{C}}_{r_k v_\tau}^T \bar{\mathbf{g}}_{r_k} - \delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge \bar{\mathbf{C}}_{r_k v_\tau}^T \delta\mathbf{g}_{r_k}}_{-\delta\mathbf{a}_{v_\tau}^{v_\tau i}} \\ \underbrace{\boldsymbol{\omega}_m - \bar{\mathbf{b}}_{\omega_t}}_{\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}} &= \boldsymbol{\omega}_{v_\tau}^{v_\tau i} + \underbrace{\delta\mathbf{b}_{\omega_\tau} + \mathbf{n}_w}_{-\delta\boldsymbol{\omega}_{v_\tau}^{v_\tau i}}\end{aligned} \tag{15}$$

3.2.4 IMU True State Kinematics

The true IMU kinematics in continuous time with respect to frame $\mathcal{F}_{r_k}$ is shown in Equation 16. The derivations for equation for $\dot{\mathbf{C}}_{r_k v_\tau}$, $\dot{\mathbf{r}}_{r_k}^{v_\tau r_k}$ and the biases are trivial, since by definition, $\dot{\mathbf{C}} = \mathbf{C}\boldsymbol{\omega}^\wedge$ and $\dot{\mathbf{r}} = \mathbf{v}$. The velocity must be expressed in the vehicle frame and the derivation for $\dot{\mathbf{v}}_{v_\tau}^{v_\tau r_k}$ is shown in Equation 17.

$$\begin{aligned}\dot{\mathbf{C}}_{r_k v_\tau} &= \mathbf{C}_{r_k v_\tau}[\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge \\ \dot{\mathbf{r}}_{r_k}^{v_\tau r_k} &= \mathbf{C}_{r_k v_\tau}\mathbf{v}_{v_\tau}^{v_\tau i} \\ \dot{\mathbf{v}}_{v_\tau}^{v_\tau i} &= \mathbf{a}_{v_\tau}^{v_\tau i} - [\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge \mathbf{v}_{v_\tau}^{v_\tau i} \\ \dot{\mathbf{b}}_{\omega_\tau} &= \mathbf{n}_{b_\omega} \\ \dot{\mathbf{b}}_{a_\tau} &= \mathbf{n}_{b_a}\end{aligned} \tag{16}$$

$$
\begin{aligned}
\mathbf{v}_{v_\tau}^{v_\tau i} &= \mathbf{C}_{r_k v_\tau}^T \mathbf{v}_{r_k}^{v_\tau i} \\
\dot{\mathbf{v}}_{v_\tau}^{v_\tau i} &= \dot{\mathbf{C}}_{r_k v_\tau}^T \mathbf{v}_{r_k}^{v_\tau i} + \mathbf{C}_{r_k v_\tau}^T \dot{\mathbf{v}}_{r_k}^{v_\tau i} \\
&= -[\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge \mathbf{C}_{r_k v_\tau}^T \mathbf{v}_{r_k}^{v_\tau r_k} + \dot{\mathbf{v}}_{v_\tau}^{v_\tau r_k} \\
&= -[\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge \mathbf{v}_{v_\tau}^{v_\tau i} + \mathbf{a}_{v_\tau}^{v_\tau i}
\end{aligned}
\tag{17}
$$

Substituting Equation 14 into Equation 16 yields Equation 18 which is the final true IMU continuous time kinematics.

$$
\begin{aligned}
\dot{\mathbf{C}}_{r_k v_\tau} &= \mathbf{C}_{r_k v_\tau} (\boldsymbol{\omega}_m - \mathbf{b}_{\omega_t} - \mathbf{n}_\omega)^\wedge \\
\dot{\mathbf{r}}_{r_k}^{v_\tau r_k} &= \mathbf{C}_{r_k v_\tau} \mathbf{v}_{v_\tau}^{v_\tau i} \\
\dot{\mathbf{v}}_{v_\tau}^{v_\tau i} &= (\mathbf{a}_m - \mathbf{C}_{v_\tau r_k} \mathbf{g}_{r_k} - \mathbf{b}_{a_\tau} - \mathbf{n}_a) - (\boldsymbol{\omega}_m - \mathbf{b}_{\omega_t} - \mathbf{n}_\omega)^\wedge \mathbf{v}_{v_\tau}^{v_\tau i} \\
\dot{\mathbf{b}}_{\omega_\tau} &= \mathbf{n}_{b_\omega} \\
\dot{\mathbf{b}}_{a_\tau} &= \mathbf{n}_{b_a}
\end{aligned}
\tag{18}
$$

3.2.5 IMU Nominal State Kinematics

The nominal state kinematics is the true state kinematics with all noise terms removed. This is the model we are using to propagate the imperfect states we are estimating. Removing all noise terms in Equation 18 yields Equation 19.

$$
\begin{aligned}
\dot{\bar{\mathbf{C}}}_{r_k v_\tau} &= \bar{\mathbf{C}}_{r_k v_\tau} (\boldsymbol{\omega}_m - \bar{\mathbf{b}}_{\omega_t})^\wedge \\
\dot{\bar{\mathbf{r}}}_{r_k}^{v_\tau r_k} &= \bar{\mathbf{C}}_{r_k v_\tau} \bar{\mathbf{v}}_{v_\tau}^{v_\tau i} \\
\dot{\bar{\mathbf{v}}}_{v_\tau}^{v_\tau i} &= (\mathbf{a}_m - \bar{\mathbf{C}}_{v_\tau r_k} \bar{\mathbf{g}}_{r_k} - \bar{\mathbf{b}}_{a_\tau}) - [\boldsymbol{\omega}_m - \bar{\mathbf{b}}_{\omega_t}]^\wedge \bar{\mathbf{v}}_{v_\tau}^{v_\tau i} \\
\dot{\bar{\mathbf{b}}}_{\omega_\tau} &= \mathbf{0} \\
\dot{\bar{\mathbf{b}}}_{a_\tau} &= \mathbf{0}
\end{aligned}
\tag{19}
$$

3.2.6 IMU Error State Kinematics

For the error states we are concerned with the propagation of the relative states' errors $\delta\mathbf{x}_{r_k v_\tau}$: relative pose error $(\delta\boldsymbol{\phi}_{r_k v_\tau}, \mathbf{r}_{r_k}^{v_\tau r_k})$, vehicle-centric velocity error, and biases' errors $\delta\mathbf{b}_{\omega_\tau}$, and $\delta\mathbf{b}_{a_\tau}$ $\mathbf{v}_{v_\tau}^{v_\tau i}$. The global state errors $\delta\mathbf{x}_{r_k i}$: global pose error $(\delta\boldsymbol{\phi}_{r_k i}, \delta\mathbf{r}_{r_k}^{i r_k T})$ and local gravity error $\delta\mathbf{g}_{r_k}^T$ don't change as they do not gain any information from the new IMU measurements. The propagation of the global states are set to zero shown in Equation 20. The derivation of IMU biases propagation is trivial as they are only affected by noise from random walk, shown in Equation 21.

$$
\begin{aligned}
\delta\dot{\boldsymbol{\phi}}_{r_k i} &= \mathbf{0} \\
\delta\dot{\mathbf{r}}_{r_k}^{i r_k} &= \mathbf{0} \\
\delta\dot{\mathbf{g}}_{r_k} &= \mathbf{0}
\end{aligned}
\tag{20}
$$

$$
\begin{aligned}
\delta\dot{\mathbf{b}}_{\omega_\tau} &= \dot{\mathbf{b}}_{\omega_\tau} - \dot{\bar{\mathbf{b}}}_{\omega_\tau} = \mathbf{n}_{b_\omega} \\
\delta\dot{\mathbf{b}}_{a_\tau} &= \dot{\mathbf{b}}_{a_\tau} - \dot{\bar{\mathbf{b}}}_{a_\tau} = \mathbf{n}_{b_a}
\end{aligned}
\tag{21}
$$

The full error state Equations are shown in Equation 22. In the reset of this subsection, we show the derivation for $\delta\dot{\boldsymbol{\phi}}_{r_k v_\tau}$, $\delta\dot{\mathbf{r}}_{r_k}^{v_\tau r_k}$, and $\delta\dot{\mathbf{v}}_{v_\tau}^{v_\tau i}$.

$$\begin{aligned}
\delta\dot{\boldsymbol{\phi}}_{r_k v_\tau} &= -[\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge \delta\boldsymbol{\phi}_{r_k v_\tau} - \delta\mathbf{b}_{\omega_\tau} - \mathbf{n}_w \\
\delta\dot{\mathbf{r}}_{r_k}^{v_\tau r_k} &= \bar{\mathbf{C}}_{r_k v_\tau}\delta\mathbf{v}_{v_\tau}^{v_\tau i} - \bar{\mathbf{C}}_{r_k v_\tau}[\bar{\mathbf{v}}_{v_\tau}^{v_\tau i}]^\wedge \delta\boldsymbol{\phi}_{r_k v_\tau} \\
\delta\dot{\mathbf{v}}_{v_\tau}^{v_\tau i} &= -\bar{\mathbf{C}}_{r_k v_\tau}^T \delta\mathbf{g}_{r_k} - (\bar{\mathbf{C}}_{r_k v_\tau}^T \bar{\mathbf{g}}_{r_k})^\wedge \delta\boldsymbol{\phi}_{r_k v_\tau} - [\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge \delta\mathbf{v}_{v_\tau}^{v_\tau i} \\
&\quad - [\bar{\mathbf{v}}_{v_\tau}^{v_\tau i}]^\wedge (\delta\mathbf{b}_{\omega_\tau} + \mathbf{n}_w) - (\delta\mathbf{b}_{a_\tau} + \mathbf{n}_a) \\
\delta\dot{\mathbf{b}}_{\omega_\tau} &= \mathbf{n}_{b_\omega} \\
\delta\dot{\mathbf{b}}_{a_\tau} &= \mathbf{n}_{b_a}
\end{aligned} \tag{22}$$

Derivation of Orientation Error Kinematics To derive the continuous time error state kinematics for rotation, we start with a applying perturbation on the nominal state $\mathbf{C}_{r_k v_\tau}$ and take the derivative of the left hand side and the right hand side.

$$\begin{aligned}
\mathbf{C}_{r_k v_\tau} &= \bar{\mathbf{C}}_{r_k v_\tau} \exp(\delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge) \\
\dot{\mathbf{C}}_{r_k v_\tau} &= \dot{\bar{\mathbf{C}}}_{r_k v_\tau} \exp(\delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge) + \bar{\mathbf{C}}_{r_k v_\tau} \dot{\exp}(\delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge) \\
\mathbf{C}_{r_k v_\tau}[\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge &= \bar{\mathbf{C}}_{r_k v_\tau}[\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge \exp(\delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge) + \bar{\mathbf{C}}_{r_k v_\tau} \dot{\exp}(\delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge)
\end{aligned} \tag{23}$$

Then, we substitute perturbation from equation 11, then simply by canceling out the nominal rotation from both sides of the equation.

$$\begin{aligned}
\bar{\mathbf{C}}_{r_k v_\tau} \exp(\delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge)[\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge &= \bar{\mathbf{C}}_{r_k v_\tau}[\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge \exp(\delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge) + \bar{\mathbf{C}}_{r_k v_\tau} \dot{\exp}(\delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge) \\
\exp(\delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge)[\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge &= [\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge \exp(\delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge) + \dot{\exp}(\delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge)
\end{aligned} \tag{24}$$

We now apply the approximation $\exp(\delta\boldsymbol{\phi}^\wedge) \approx \mathbf{1} + \delta\boldsymbol{\phi}^\wedge$ to the results from the previous step and expand the terms in the brackets. This approximation is valid since it is assumed the perturbations have a value always close to zero.

$$\begin{aligned}
(\mathbf{1} + \delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge)[\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i} + \delta\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge &= [\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge (\mathbf{1} + \delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge) + \delta\dot{\boldsymbol{\phi}}_{r_k v_\tau}^\wedge \\
[\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge + [\delta\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge + \delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge[\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge + \delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge[\delta\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge &= [\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge + [\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge \delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge + \delta\dot{\boldsymbol{\phi}}_{r_k v_\tau}^\wedge
\end{aligned} \tag{25}$$

After cancellation of like terms and some rearrangement, we have as follows.

$$\delta\dot{\boldsymbol{\phi}}_{r_k v_\tau}^\wedge = \delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge[\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge - [\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge \delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge + [\delta\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge \tag{26}$$

Then, the identity of $(\mathbf{u}^\wedge\mathbf{v})^\wedge = \mathbf{u}^\wedge\mathbf{v}^\wedge - \mathbf{v}^\wedge\mathbf{u}^\wedge$ is applied. Finally, the kinematics of the orientation error is obtained by substituting in $\delta\boldsymbol{\omega}_{v_\tau}^{v_\tau i}$ from Equation 15 and applying the identity $\mathbf{u}^\wedge\mathbf{v} = -\mathbf{v}^\wedge\mathbf{u}$.

$$\begin{aligned}
\delta\dot{\boldsymbol{\phi}}_{r_k v_\tau} &= \delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge[\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}] + \delta\boldsymbol{\omega}_{v_\tau}^{v_\tau i} \\
&= -[\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge \delta\boldsymbol{\phi}_{r_k v_\tau} - \delta\mathbf{b}_{\omega_\tau} - \mathbf{n}_w
\end{aligned} \tag{27}$$

Derivation of Translation Error Kinematics We again start with the perturbation model for relative translation shown in Equation 11. We then take the derivative and

rearrange the equation, and substitute $\dot{\mathbf{r}}_{r_k}^{v_\tau r_k}$ from Equation 16 into the resulting derivation.

$$\begin{aligned}\mathbf{r}_{r_k}^{ir_k} &= \bar{\mathbf{r}}_{r_k}^{ir_k} + \delta\mathbf{r}_{r_k}^{ir_k} \\ \delta\dot{\mathbf{r}}_{r_k}^{v_\tau r_k} &= \dot{\mathbf{r}}_{r_k}^{v_\tau r_k} - \dot{\bar{\mathbf{r}}}_{r_k}^{v_\tau r_k} \\ &= \mathbf{C}_{r_k v_\tau}\mathbf{v}_{v_\tau}^{v_\tau i} - \bar{\mathbf{C}}_{r_k v_\tau}\bar{\mathbf{v}}_{v_\tau}^{v_\tau i}\end{aligned} \tag{28}$$

We further expand $\mathbf{C}_{r_k v_\tau}\mathbf{v}_{v_\tau}^{v_\tau i}$ into their perturbed form using Equation 11 again, and expand the terms in the brackets after using the approximation for $\exp(\boldsymbol{\phi}^\wedge)$

$$\begin{aligned}\delta\dot{\mathbf{r}}_{r_k}^{v_\tau r_k} &= \bar{\mathbf{C}}_{r_k v_\tau}\exp(\delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge)(\bar{\mathbf{v}}_{v_\tau}^{v_\tau i} + \delta\mathbf{v}_{v_\tau}^{v_\tau i}) - \bar{\mathbf{C}}_{r_k v_\tau}\bar{\mathbf{v}}_{v_\tau}^{v_\tau i} \\ &= \bar{\mathbf{C}}_{r_k v_\tau}(\mathbf{1} + \delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge)(\bar{\mathbf{v}}_{v_\tau}^{v_\tau i} + \delta\mathbf{v}_{v_\tau}^{v_\tau i}) - \bar{\mathbf{C}}_{r_k v_\tau}\bar{\mathbf{v}}_{v_\tau}^{v_\tau i} \\ &= \bar{\mathbf{C}}_{r_k v_\tau}\bar{\mathbf{v}}_{v_\tau}^{v_\tau i} + \bar{\mathbf{C}}_{r_k v_\tau}\delta\mathbf{v}_{v_\tau}^{v_\tau i} + \bar{\mathbf{C}}_{r_k v_\tau}\delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge\bar{\mathbf{v}}_{v_\tau}^{v_\tau i} + \bar{\mathbf{C}}_{r_k v_\tau}\delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge\delta\mathbf{v}_{v_\tau}^{v_\tau i} - \bar{\mathbf{C}}_{r_k v_\tau}\bar{\mathbf{v}}_{v_\tau}^{v_\tau i}\end{aligned} \tag{29}$$

Finally, after canceling out like terms and removing second order terms, we apply the identity $\mathbf{u}^\wedge\mathbf{v} = -\mathbf{v}^\wedge\mathbf{u}$ to arrive at the final perturbation kinematics for translation.

$$\begin{aligned}\delta\dot{\mathbf{r}}_{r_k}^{v_\tau r_k} &= \bar{\mathbf{C}}_{r_k v_\tau}\delta\mathbf{v}_{v_\tau}^{v_\tau i} + \bar{\mathbf{C}}_{r_k v_\tau}\delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge\bar{\mathbf{v}}_{v_\tau}^{v_\tau i} \\ &= \bar{\mathbf{C}}_{r_k v_\tau}\delta\mathbf{v}_{v_\tau}^{v_\tau i} - \bar{\mathbf{C}}_{r_k v_\tau}[\bar{\mathbf{v}}_{v_\tau}^{v_\tau i}]^\wedge\delta\boldsymbol{\phi}_{r_k v_\tau}\end{aligned} \tag{30}$$

Derivation of Translation Error Kinematics We again start with the perturbation model for velocity shown in Equation 11. We again take the derivative and rearrange the equation, and substitute $\dot{\mathbf{v}}_{v_\tau}^{v_\tau i}$ from Equation 16 into the resulting derivation

$$\begin{aligned}\mathbf{v}_{v_\tau}^{v_\tau i} &= \bar{\mathbf{v}}_{v_\tau}^{v_\tau i} + \delta\mathbf{v}_{v_\tau}^{v_\tau i} \\ \delta\dot{\mathbf{v}}_{v_\tau}^{v_\tau i} &= \dot{\mathbf{v}}_{v_\tau}^{v_\tau i} - \dot{\bar{\mathbf{v}}}_{v_\tau}^{v_\tau i} \\ &= \mathbf{a}_{v_\tau}^{v_\tau i} - [\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge\mathbf{v}_{v_\tau}^{v_\tau i} - \bar{\mathbf{a}}_{v_\tau}^{v_\tau i} + [\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge\bar{\mathbf{v}}_{v_\tau}^{v_\tau i}\end{aligned} \tag{31}$$

We further expand the terms with their into their perturbed forms using Equation 11. Then, after substituting the definition of $\delta\mathbf{a}_{v_\tau}^{v_\tau i}$ from Equation 15 and fully expand the terms, the equation is simplified through canceling out like terms and removing second order terms.

$$\begin{aligned}\delta\dot{\mathbf{v}}_{v_\tau}^{v_\tau i} &= \delta\mathbf{a}_{v_\tau}^{v_\tau i} - [\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i} + \delta\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge(\bar{\mathbf{v}}_{v_\tau}^{v_\tau i} + \delta\mathbf{v}_{v_\tau}^{v_\tau i}) + [\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge\bar{\mathbf{v}}_{v_\tau}^{v_\tau i} \\ &= -\delta\mathbf{b}_{a_\tau} - \mathbf{n}_a - \bar{\mathbf{C}}_{r_k v_\tau}^T\delta\mathbf{g}_{r_k} + \delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge\bar{\mathbf{C}}_{r_k v_\tau}^T\bar{\mathbf{g}}_{r_k} + \delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge\bar{\mathbf{C}}_{r_k v_\tau}^T\delta\mathbf{g}_{r_k} - \\ &\quad ([\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge\mathbf{v}_{v_\tau}^{v_\tau i} + [\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge\delta\mathbf{v}_{v_\tau}^{v_\tau i} + [\delta\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge\bar{\mathbf{v}}_{v_\tau}^{v_\tau i} + [\delta\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge\delta\mathbf{v}_{v_\tau}^{v_\tau i}) + [\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge\bar{\mathbf{v}}_{v_\tau}^{v_\tau i} \\ &= -\bar{\mathbf{C}}_{r_k v_\tau}^T\delta\mathbf{g}_{r_k} + \delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge\bar{\mathbf{C}}_{r_k v_\tau}^T\bar{\mathbf{g}}_{r_k} - ([\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge\delta\mathbf{v}_{v_\tau}^{v_\tau i} + [\delta\boldsymbol{\omega}_{v_\tau}^{v_\tau i}]^\wedge\mathbf{v}_{v_\tau}^{v_\tau i}) - \delta\mathbf{b}_{a_\tau} - \mathbf{n}_a\end{aligned} \tag{32}$$

Finally, we substitute the definition of $\delta\omega_{v_\tau}^{v_\tau i}$ and arrive at the velocity perturbation kinematics after rearranging the terms.

$$\begin{aligned}\delta\dot{\mathbf{v}}_{v_\tau}^{v_\tau i} &= -\bar{\mathbf{C}}_{r_k v_\tau}^T\delta\mathbf{g}_{r_k} + \delta\boldsymbol{\phi}_{r_k v_\tau}^\wedge\bar{\mathbf{C}}_{r_k v_\tau}^T\bar{\mathbf{g}}_{r_k} - [\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge\delta\mathbf{v}_{v_\tau}^{v_\tau i} + (\delta\mathbf{b}_{\omega_\tau} + \mathbf{n}_w)^\wedge\bar{\mathbf{v}}_{v_\tau}^{v_\tau i} - \delta\mathbf{b}_{a_\tau} - \mathbf{n}_a \\ &= -\bar{\mathbf{C}}_{r_k v_\tau}^T\delta\mathbf{g}_{r_k} - (\bar{\mathbf{C}}_{r_k v_\tau}^T\bar{\mathbf{g}}_{r_k})^\wedge\delta\boldsymbol{\phi}_{r_k v_\tau} - [\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge\delta\mathbf{v}_{v_\tau}^{v_\tau i} - [\bar{\mathbf{v}}_{v_\tau}^{v_\tau i}]^\wedge(\delta\mathbf{b}_{\omega_\tau} + \mathbf{n}_w) - (\delta\mathbf{b}_{a_\tau} + \mathbf{n}_a)\end{aligned} \tag{33}$$

3.2.7 RC-KF Prediction

The nominal state model shown in Equation 19 and Equation 22 are in continuous time. However, IMU data are sampled at some time intervals δt. In order to use sampled data,

we need to integrate over the δt amount of time between samples. In this subsection, the equations for integration and discretization are presented. Note that propagating the error states are not required since they reset to zero after each KF update. Thus, we only propagate the nominal states and the error state covariances.

Nominal States The only states that requires propagation are the rotation $\mathbf{C}_{r_k v_\tau}$, translation $\mathbf{r}_{r_k}^{v_\tau r_k}$, and velocity $\mathbf{v}_{r_k}^{v_\tau i}$ states. The global states and the biases remains constant during the period of propagation.

$$\begin{aligned} \check{\mathbf{x}}_{r_k i} &= \hat{\mathbf{x}}_{r_k i} \\ \check{\mathbf{b}}_{\omega_\tau} &= \check{\mathbf{b}}_{\omega_k} \\ \check{\mathbf{b}}_{a_\tau} &= \check{\mathbf{b}}_{a_k} \end{aligned} \tag{34}$$

Integration of the states is shown in Equation 35, where $\Delta t = t_\tau - t_k$. The propagation of all the states are performed with respect to the reference frame $\mathbb{F}_{r_k}$. To obtain $\check{\mathbf{v}}_{r_k}^{v_k i}$ at the end of the propagation, it needs to be rotated with $\check{\mathbf{v}}_{r_k}^{v_{k+1} i} = \check{\mathbf{C}}_{r_k v_{k+1}}^T \check{\mathbf{v}}_{r_k}^{v_{k+1} i}$.

$$\begin{aligned} \check{\mathbf{C}}_{r_k v_\tau} &= \int_{t_k}^{t_\tau} \check{\mathbf{C}}_{r_k v_s} (\boldsymbol{\omega}_m - \check{\mathbf{b}}_{\omega_s})^\wedge ds \\ \check{\mathbf{r}}_{r_k}^{v_\tau r_k} &= \hat{\mathbf{v}}_{r_k}^{v_k i} \Delta t - \frac{1}{2} \check{\mathbf{g}}_{r_k} \Delta t^2 + \underbrace{\iint_{t_k}^{t_\tau} \check{\mathbf{C}}_{r_k v_s} (\mathbf{a}_m - \check{\mathbf{b}}_{a_s}) ds^2}_{\check{\boldsymbol{\alpha}}_{r_k}^{v_\tau r_k}} \\ \check{\mathbf{v}}_{r_k}^{v_\tau i} &= \hat{\mathbf{v}}_{r_k}^{v_k i} - \check{\mathbf{g}}_{r_k} \Delta t + \underbrace{\int_{t_k}^{t_\tau} \check{\mathbf{C}}_{r_k v_s} (\mathbf{a}_m - \check{\mathbf{b}}_{a_s}) ds}_{\check{\boldsymbol{\beta}}_{r_k}^{v_\tau i}} \end{aligned} \tag{35}$$

We then approximate the integrals from Equation 35 using Euler's method, the discrete integration occurs with each new IMU measurement. Higher order methods such as Runge-Kutta 4 can also be used, however using Euler's method simpler and it is sufficient due to the high-rate nature of IMU's.

$$\begin{aligned} \check{\mathbf{C}}_{r_k v_{\tau+1}} &\approx \check{\mathbf{C}}_{r_k v_\tau} \exp((\boldsymbol{\omega}_m - \check{\mathbf{b}}_{\omega_\tau})^\wedge \delta t) \\ \check{\boldsymbol{\alpha}}_{r_k}^{v_{\tau+1} r_k} &\approx \check{\boldsymbol{\alpha}}_{r_k}^{v_\tau r_k} + \check{\boldsymbol{\beta}}_{r_k}^{v_\tau i} \delta t + \frac{1}{2} \check{\mathbf{C}}_{r_k v_\tau} (\mathbf{a}_m - \check{\mathbf{b}}_{a_\tau}) \delta t^2 \\ \check{\boldsymbol{\beta}}_{r_k}^{v_{\tau+1} i} &\approx \check{\boldsymbol{\beta}}_{r_k}^{v_\tau i} + \check{\mathbf{C}}_{r_k v_\tau} (\mathbf{a}_m - \check{\mathbf{b}}_{a_\tau}) \delta t \end{aligned} \tag{36}$$

Error States Covariances For KF prediction, we rewrite Equation 22 into a matrix form shown in 37, where $\mathbf{F}$ is the linearized motion model, $\mathbf{G}$ is the linearized noise model, and $\mathbf{n} = [\mathbf{n}_w^T, \mathbf{n}_{b_w}^T, \mathbf{n}_a^T, \mathbf{n}_{b_a}]^T$ is the IMU noise.

$$\delta \dot{\mathbf{x}}_\tau = \mathbf{F} \delta \mathbf{x}_\tau + \mathbf{G} \mathbf{n} \tag{37}$$

$$\mathbf{F} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -[\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge & \mathbf{0} & \mathbf{0} & -\mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\bar{\mathbf{C}}_{r_k v_\tau}[\bar{\mathbf{v}}_{v_\tau}^{v_\tau i}]^\wedge & \mathbf{0} & \bar{\mathbf{C}}_{r_k v_\tau} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\bar{\mathbf{C}}_{r_k v_\tau}^T & -(\bar{\mathbf{C}}_{r_k v_\tau}^T \bar{\mathbf{g}}_{r_k})^\wedge & \mathbf{0} & -[\bar{\boldsymbol{\omega}}_{v_\tau}^{v_\tau i}]^\wedge & -[\bar{\mathbf{v}}_{v_\tau}^{v_\tau i}]^\wedge & -\mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -[\bar{\mathbf{v}}_{v_\tau}^{v_\tau i}]^\wedge & \mathbf{0} & -\mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{38}$$

The state transition matrix $\boldsymbol{\Phi}_{\tau+1,\tau}$ can be computed using Equation 39. Here the $exp(\cdot)$ is the matrix exponential, and we chose to use a second order approximation for computing $\boldsymbol{\Phi}_{\tau+1,\tau}$.

$$\begin{aligned} \boldsymbol{\Phi}_{\tau+1,\tau} &= \exp(\int_{t_\tau}^{t_{\tau+1}} \mathbf{F}(s)ds) \\ &\approx \mathbf{1} + \mathbf{F}_t \delta t + \frac{1}{2}\mathbf{F}_t^2 \delta t^2 \end{aligned} \tag{39}$$

With $\boldsymbol{\Phi}_{\tau+1,\tau}$ computed, we can calculate how IMU noise represented by the covariance matrix $\mathbf{Q}$ affects the states at time τ using Equation 40. Again, the integral is approximated as closed form solution isn't available.

$$\begin{aligned} \mathbf{Q}_\tau &\approx \int_{t_\tau}^{t_{\tau+1}} \boldsymbol{\Phi}(t_\tau, s)\mathbf{G}\mathbf{Q}\mathbf{G}^T\boldsymbol{\Phi}(t_\tau, s)^T ds \\ &\approx \boldsymbol{\Phi}_{\tau+1,\tau}\mathbf{G}\mathbf{Q}\mathbf{G}^T\boldsymbol{\Phi}_{\tau+1,\tau}^T \delta t \end{aligned} \tag{40}$$

$$\mathbf{Q} = \begin{bmatrix} \sigma_w^2\mathbf{I} & 0 & 0 & 0 \\ 0 & \sigma_{\mathbf{b}_w}^2\mathbf{I} & 0 & 0 \\ 0 & 0 & \sigma_a^2\mathbf{I} & 0 \\ 0 & 0 & 0 & \sigma_{\mathbf{b}_a}^2\mathbf{I} \end{bmatrix} \tag{41}$$

Covariance is propagated recursively, starting from $\check{\mathbf{P}}_k$, all the way to $\check{\mathbf{P}}_{k+1}$ Finally, the state covariance is propagated recursively according to Equation 42.

$$\check{\mathbf{P}}_\tau = \boldsymbol{\Phi}_{\tau+1,\tau}\check{\mathbf{P}}_\tau\boldsymbol{\Phi}_{\tau+1,\tau}^T + \mathbf{Q}_\tau \tag{42}$$

3.2.8 RC-KF Update

Given a measurement model that is a function of the states as shown in Equation 43, where $h(\mathbf{x}_{k+1})$ is the measurement function which maps some sensor data to the nominal states at time $k+1$, and $\mathbf{n}_r$ is the noise associated with the measurement that is zero-mean Gaussian with covariance $\mathbf{R}_{k+1}$.

$$\begin{aligned} \mathbf{y}_k &= h(\mathbf{x_k}) + \mathbf{n}_r \\ \mathbf{n}_r &\sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k) \end{aligned} \tag{43}$$

We linearize $h(\mathbf{x}_{k+1})$ with respect to the error states $\delta\mathbf{x}_{k+1}$ to obtain the measurement Jacobian $\mathbf{H}_{k+1}$.

$$\mathbf{H}_{k+1} = \frac{\partial h(\mathbf{x}_{k+1})}{\partial \delta\mathbf{x}_{k+1}} \tag{44}$$

Then, the Kalman update is performed as shown in Equation 45 to obtain the best estimate of errors at time $k+1$.

$$\begin{aligned}
\mathbf{K}_{k+1} &= \check{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T(\mathbf{H}_{k+1}\check{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T + \mathbf{R}_{k+1})^{-1} \\
\hat{\mathbf{P}}_{k+1} &= (\mathbf{1} - \mathbf{K}_{k+1}\mathbf{H}_{k+1})\check{\mathbf{P}}_{k+1} \\
\delta\hat{\mathbf{x}}_{k+1} &= \mathbf{K}_{k+1}(\mathbf{y}_{k+1} - h(\check{\mathbf{x}}_{k+1}))
\end{aligned} \tag{45}$$

After obtaining an estimate of the error $\delta\hat{\mathbf{x}}_{k+1}$, it is injected back into the nominal states shown in Equation 46. This is performed according to the perturbations defined in Equation 11. Note that the global pose states $\mathbf{C}_{r_k i}$ and $\mathbf{r}_{r_k}^{ir_k}$ are not observable by the filter, so that their error estimates are zero and not affected by the update step.

$$\hat{\mathbf{g}}_{r_k} = \check{\mathbf{g}}_{r_k} + \delta\hat{\mathbf{g}}_{r_k} \tag{46}$$

$$\hat{\mathbf{C}}_{r_k v_{k+1}} = \check{\mathbf{C}}_{r_k v_{k+1}} \exp(\delta\hat{\boldsymbol{\phi}}_{r_k v_{k+1}}) \tag{47}$$

$$\hat{\mathbf{r}}_{r_k}^{v_{k+1} r_k} = \check{\mathbf{r}}_{r_k}^{v_{k+1} r_k} + \delta\hat{\mathbf{r}}_{r_k}^{v_{k+1} r_k} \tag{48}$$

$$\hat{\mathbf{v}}_{v_{k+1}}^{v_{k+1} i} = \check{\mathbf{v}}_{v_{k+1}}^{v_{k+1} i} + \delta\hat{\mathbf{v}}_{v_{k+1}}^{v_{k+1} i} \tag{49}$$

$$\hat{\mathbf{b}}_{a_{k+1}} = \check{\mathbf{b}}_{a_{k+1}} + \delta\hat{\mathbf{b}}_{a_{k+1}} \tag{50}$$

$$\hat{\mathbf{b}}_{\omega_{k+1}} = \check{\mathbf{b}}_{\omega_{k+1}} + \delta\hat{\mathbf{b}}_{\omega_{k+1}} \tag{51}$$

3.2.9 RC-KF Reference Frame Update

Finally, the reference frame is shifted forward by from frame $\mathcal{F}_k$ to image frame $\mathcal{F}_{k+1}$ shown in Equation 52. This is referred to in [22] as the composition step. The relative pose is concatenated to the global pose, then it is set to identity. The velocity and biases are already in the vehicle frame and they are carried forward.

$$\begin{aligned}
\hat{\mathbf{C}}_{r_{k+1} i} &= \hat{\mathbf{C}}_{r_k v_{k+1}}^T \hat{\mathbf{C}}_{r_k i} \\
\hat{\mathbf{r}}_{r_{k+1}}^{i r_{k+1}} &= \hat{\mathbf{C}}_{r_k v_{k+1}}^T (\hat{\mathbf{r}}_{r_k}^{i r_k} - \hat{\mathbf{r}}_{r_k}^{v_{k+1} r_k}) \\
\hat{\mathbf{g}}_{r_{k+1}} &= \hat{\mathbf{C}}_{r_k v_{k+1}}^T \hat{\mathbf{g}}_{r_k} \\
\hat{\mathbf{C}}_{r_{k+1} v_{k+1}} &= \mathbf{1} \\
\hat{\mathbf{r}}_{r_{k+1}}^{v_{k+1} r_{k+1}} &= \mathbf{0} \\
\hat{\mathbf{v}}_{v_{k+1}}^{v_{k+1} i} &= \hat{\mathbf{v}}_{v_{k+1}}^{v_{k+1} i} \\
\hat{\mathbf{b}}_{a_{k+1}} &= \hat{\mathbf{b}}_{a_{k+1}} \\
\hat{\mathbf{b}}_{\omega_{k+1}} &= \hat{\mathbf{b}}_{\omega_{k+1}}
\end{aligned} \tag{52}$$

In addition to updating the reference frame of the states, the uncertainties associated with the error states must also be propagated. This is shown in Equation 53. The covariance of relative pose are propagated to the global pose due to the concatenating in Equation 52.

Then, the relative pose covariance is set to zero for the next iteration. The uncertainty of velocity and biases are not changed.

$$\hat{\mathbf{P}}'_{k+1} = \mathbf{U}_{k+1}\hat{\mathbf{P}}_{k+1}\mathbf{U}_{k+1}^T \tag{53}$$

$$\mathbf{U}_{k+1} = \frac{\partial \delta \check{\mathbf{x}}_{k+1}}{\partial \delta \hat{\mathbf{x}}_{k+1}} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & -\hat{\mathbf{C}}^T_{r_{k+1}i} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{C}}^T_{r_k v_{k+1}} & \mathbf{0} & [\hat{\mathbf{r}}^{ir_{k+1}}_{r_{k+1}}]^\wedge & -\hat{\mathbf{C}}^T_{r_k v_{k+1}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \hat{\mathbf{C}}^T_{r_k v_{k+1}} & \check{\mathbf{g}}^\wedge_{r_{k+1}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{54}$$

3.3 Deep Learning

3.3.1 Feedforward Neural Networks

An feedforward network (Figure 5) is a nonlinear function $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$ that approximates a function $\mathbf{y}^* = f^*(\mathbf{x})$, where $\mathbf{x}$ is an input tensor and $\boldsymbol{\theta}$ is a set of parameters that best approximates f to f^*. $f(\mathbf{x}; \boldsymbol{\theta})$ is composed of a chain of nested nonlinear functions, with each function in the chain f_i known as a layer, with its own set of parameters $\boldsymbol{\theta}_i$. The first layer is also called the input layer, the last layer called the output layer, and the middle layers are known as the hidden layers. For example, a feedforward network with three layers would be: $f(\mathbf{x}; \boldsymbol{\theta}) = f_3(f_2(f_1(\mathbf{x}; \boldsymbol{\theta}_1); \boldsymbol{\theta}_2); \boldsymbol{\theta}_3)$. A common hidden layer is composed of a linear function followed by a nonlinear activation function that is applied elementwise to a vector, i.e. $f_i(\mathbf{x}_i; \boldsymbol{\theta}_i) = g(\mathbf{W}_i\mathbf{x}_i + \mathbf{b}_i)$, where $\mathbf{W}_i$ and $\mathbf{b}_i$ are the parameters also known as weights and biases respectively for that layer, and g is the nonlinear activation function. A commonly used activation function is ReLU (Rectified linear unit) defined by $g(z) = \max(0, z)$. There are many other types of layers and activation functions used in practice. The selection of output layers are specific to the task, they are further elaborated in [19].

The power of neural networks presides in the nested combination of functions which allows network approximate a wide range of functions given enough layers (depth of network) and appropriate parameters. The parameters are trained through backpropagation. For supervised learning where the labels $\mathbf{y}^*$ is known, given a loss function $L(\mathbf{y}, \mathbf{y}^*)$ that describes the error between the network output and expected ground truth values, backpropagation uses repeated application of chains rules to find the gradient with respect to the network parameters $\boldsymbol{\theta}$. With the gradients computed, gradient descent is used to update the weights and biases. We refer the reader to reference [19] for more details regarding the derivation of back propagation. In practice when implementing neural networks, deep learning frameworks such as PyTorch and Tensorflow are often used. These frameworks abstracts deep network as a computational graph, which keep tracks of operations and perform backpropagation for the user automatically with minimal lines of code. In addition, different flavors of stochastic gradient descent methods are generally used in place of vanilla gradient descent which allows an optimization step to be performed on a subset of training examples which greatly improves training speed.

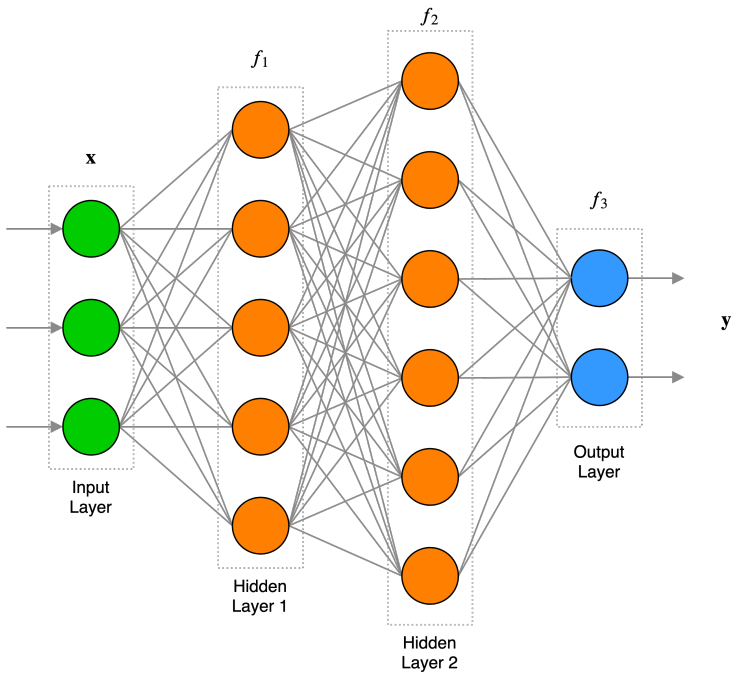


Figure 5: Feedforward neural network

3.3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) is a feedforward neural network that uses convolution in its operation. In a traditional fully connected neural networks (FCNs), every output unit interact with every input unit in the next layer, resulting in a dense matrix of parameters resulting in large memory usage. In contrast, CNNs enables parameter sharing. The CNN parameters are convolutional kernels, and the same kernel is applied cross the entire input. An illustration of this is shown in Figure 6, where it is clear parameter sharing reduces the number of parameters dramatically. Images are natural application for CNNs since 2D convolution is an ubiquitous process in classical image processing. However, rather than designing kernels by hand to extract desired features, CNNs allows many kernels to be learned through backpropagation. CNNs has shown enormous success in deep learning, they are part of almost every deep learning architecture that use image as inputs. An example of a typical CNN architecture for image classification is shown in Figure 7.

3.3.3 Recurrent Neural Networks

Unlike feedforward neural networks where information flows one way from input to the output, recurrent neural networks (RNNs) also incorporates information from an earlier time. This makes RNNs natural for modeling dynamical systems. At each time step, the RNN takes new data and previous state also known as the hidden state, and produces a transformed state and an output. This is described in equation 55. The process repeats itself in the next time step. Here $\mathbf{x}_k$ is the input at timestep k, $\mathbf{h}_{k-1}$ is the hidden state from the previous timestep, and $\mathbf{h}_k$ is the hidden state from the current timestep. $\mathcal{H}$ is a

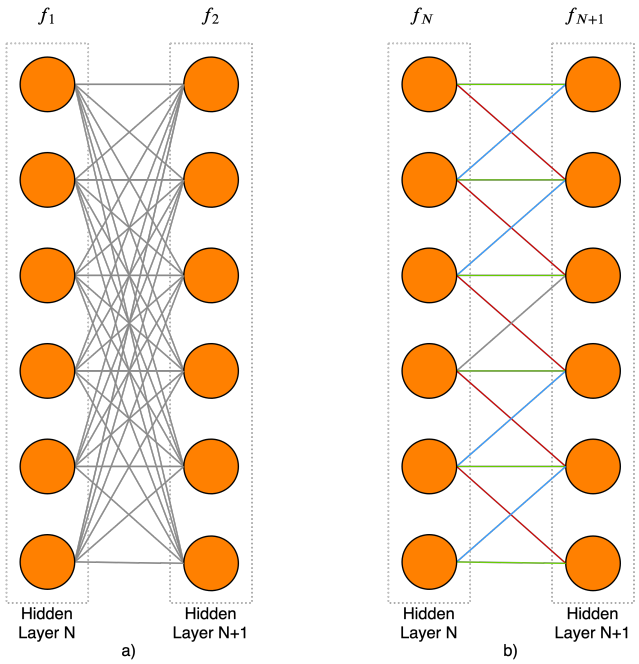


Figure 6: CNN parameter sharing, a) A fully connected feedforward network, b) network with parameter sharing, the same color indicates the same parameters, notice the number of parameters decreases dramatically

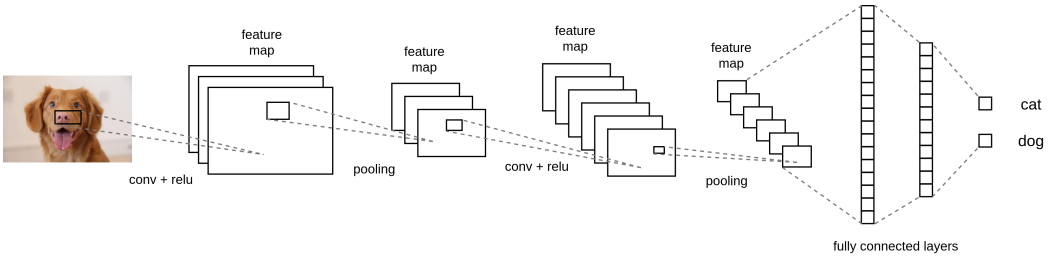


Figure 7: An example of CNN architecture for classification

nonlinear activation function such as tanh. The learned parameters are the weights $\mathbf{W}_{hx}$, $\mathbf{W}_{hh}$, and $\mathbf{W}_{hh}$, biases $\mathbf{b}_h$ and $\mathbf{b}_o$. We can draw similarities between RNN and recursive state estimation techniques such as Kalman Filters. In both cases the system takes some input and states from the previous timestep, update the states to be propagated to the next timestep, and produces some outputs. To train the RNN, the system is unrolled a fixed number of timesteps, and backpropagation is applied as if it is a feedforward network, this process is called Backpropagation Through Time (BPTT). A RNN and its unrolled form is shown in Figure 8.

$$\begin{aligned} \mathbf{h}_k &= \mathcal{H}(\mathbf{W}_{hx}\mathbf{x}_k + \mathbf{W}_{hh}\mathbf{h}_{k-1} + \mathbf{b}_h) \\ \mathbf{o}_k &= \mathbf{W}_{oh}\mathbf{h}_k + \mathbf{b}_o \end{aligned} \tag{55}$$

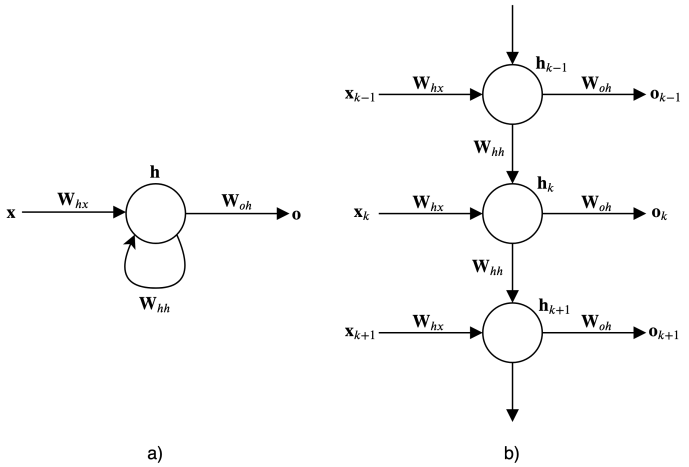


Figure 8: Recurrent neural network, a) shows a RNN unit, b) shows the same RNN unit unroll 3 timesteps between from time $k-1$ to $k+1$

In theory, a RNN can learn from arbitrary length of data. However, this is limited in practice due to the known vanishing and exploding gradients [19] in networks that are relatively deep. This is due the repeated multiplication of matrices in applying the chain rule. A strong gradient is required for the network to effectively update its parameters. In the scenario of vanishing gradient, the gradient approaches zero, and the network becomes unable to learn using information from previous timesteps. In the scenario of exploding gradient, the gradient becomes large, and the training process becomes unstable and diverges. Long Short-term Memory (LSTM) alleviates this problem by introducing mechanism to forget or retain information from the previous state. In addition to hidden states $\mathbf{h}_k$, cell states $\mathbf{c}_k$ are also propagated from one timestep to the next. At each time step, $\mathbf{c}_k$ which is a function of $\mathbf{c}_{k-1}$, $\mathbf{h}_{k-1}$, and $\mathbf{x}_k$ are used to control how $\mathbf{h}_k$ is passed to the next timestep. Figure 9 shows a diagram of LSTM for a single timestep, and 56 presents the specific formula. $\mathbf{W}$ and $\mathbf{b}$ are the weights and biases which are learned parameters. $\mathbf{i}_k$, $\mathbf{f}_k$, $\mathbf{c}_k$, and $\mathbf{o}_k$, are the input gate, forget gate, cell state, and output gate at time k, respectively.

$$
\begin{aligned}
\mathbf{i}_k &= \sigma(\mathbf{W}_{ix}\mathbf{x}_k + \mathbf{W}_{ih}\mathbf{h}_{k-1} + \mathbf{b}_i) \\
\mathbf{f}_k &= \sigma(\mathbf{W}_{fx}\mathbf{x}_k + \mathbf{W}_{fh}\mathbf{h}_{k-1} + \mathbf{b}_f) \\
\mathbf{g}_k &= \tanh(\mathbf{W}_{gx}\mathbf{x}_k + \mathbf{W}_{gh}\mathbf{h}_{k-1} + \mathbf{b}_g) \\
\mathbf{c}_k &= \mathbf{f}_k \odot \mathbf{c}_{k-1} + \mathbf{i}_k \odot \mathbf{g}_k \\
\mathbf{o}_k &= \sigma(\mathbf{W}_{ox}\mathbf{x}_k + \mathbf{W}_{oh}\mathbf{h}_{k-1} + \mathbf{b}_o)
\end{aligned}
\tag{56}
$$

3.4 DeepVO

3.4.1 Network Architecture

DeepVO [41] and its extension ESP-VO [42] are deep architectures that learns monocular visual odometry (VO) in a supervised way. The work combined CNN with RNNs to learn VO from raw RGB images in an end-to-end manner. The CNN extracts features from input

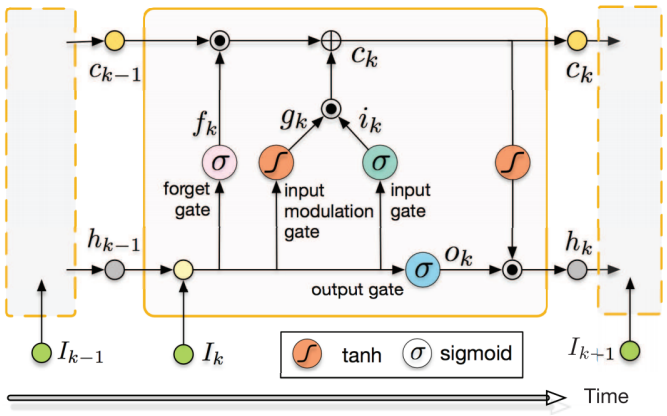


Figure 9: A LSTM cell at a single time step, image retrieved from [42]

Table 1: CNN configuration

Layer	Kernel Size	Padding	Stride	Channels
Conv1	7×7	3	2	64
Conv2	5×5	2	2	128
Conv3	5×5	2	2	256
Conv3_1	3×3	1	1	256
Conv4	3×3	1	2	512
Conv4_1	3×3	1	1	512
Conv5	3×3	1	2	512
Conv5_1	3×3	1	1	512
Conv6	3×3	1	2	1024

images and the RNN allow the system to learn complex motion dynamics over time. The authors demonstrates that the system yields comparable results to other monocular VO on the KITTI dataset [18]. The system architecture is shown in Figure 10.

The input to the network at each timestep is a pair of images at time t_k and t_{k+1} stack along the RGB channels. The images are then processed through a series of CNN layers. The architecture of the CNN layers are based on the encoder subsection of FlowNetS [11]. To learn the features at different scales, there are a total of 9 CNN layers with kernel sizes changing gradually from 7×7 to 5×5 and then to 3×3, the number of kernels increases by a factor of 2 from 64 channels to 1024 channels into the final layer, and strides are used to reduce the image size as the network depth increases. The CNN portion of the architecture is summarized in Table 1. After obtaining 1024 extracted features from the final layer of CNN, the features are flattened into a vector and fed into RNN. A LSTM is used alleviate the aforementioned vanishing and exploding gradient problem. The LSTM have 2 layers with each layer having a size of 1000 hidden states. Finally the output of RNN is passed to a FCN with a final output dimension of 6 representing 6 DoF relative motion from t_k to t_{k+1} in xyz translation and Euler angle rotations.

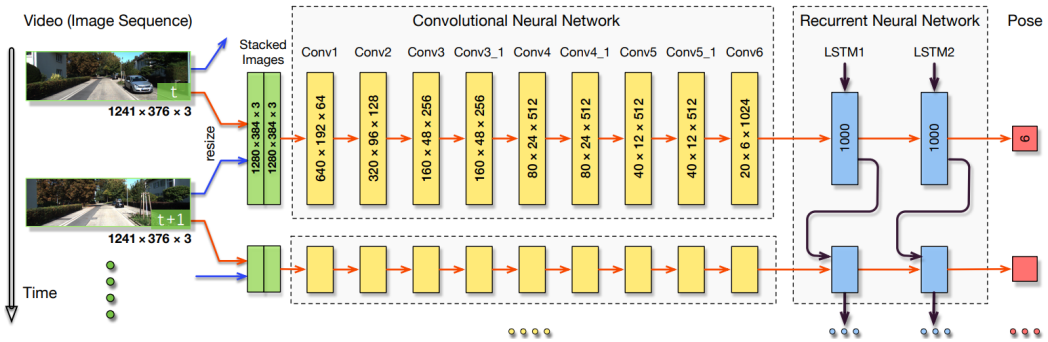


Figure 10: DeepVO architecture, image retrieved from [42]

3.4.2 Training

DeepVO is a supervised learning method, thus ground truth is required for training. Given ground truth absolute transformation $\mathbf{T}_i v_{k+1}$ and $\mathbf{T}_i v_k$ of a vehicle frame $\mathcal{F}_v$ at time t_{k+1} and t_k with respect to an inertial frame $\mathbf{F}_i$, the relative transformation of vehicle frame at time t_{k+1} with respect to t_k can be easily computed as $\mathbf{T}_{v_k v_{k+1}} = \mathbf{T}_{i v_k}^{-1} \mathbf{T}_i v_{k+1}$, and xyz translation and Euler angles can be easily extracted from the transformation matrix. The loss function used by DeepVO is a simple mean squared error of all xyz positions and Euler angles as shown in Equation 57. $\hat{\mathbf{r}}_{v_k}^{v_{k+1} v_k}$ and $\hat{\boldsymbol{\psi}}_{v_k v_{k+1}}$ are translation and rotation estimated by the network, and $\mathbf{r}_{v_k}^{v_{k+1} v_k}$ and $\boldsymbol{\psi}_{v_k v_{k+1}}$ are the corresponding ground truth values. Since rotations are generally smaller in magnitude than translations, a scaling factor of $\kappa = 100$ is applied to the rotation errors. Also note that it is assumed the rotation will be small from frame to frame, thus the directl subtraction Euler angles in Equation 57 is a good approximation of difference in rotation. N is the number of training examples in a single batch. The authors found that it is difficult for the network to converge directly from random initialization, thus weights from FlowNetS are used to initialize the CNN portion of the network. The network is trained with the Adam optimizer with a learning rate of 0.001.

$$\boldsymbol{\theta}^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{j=1}^{N} \sum_{k=1}^{t} \left\| \hat{\mathbf{r}}_{v_k}^{v_{k+1} v_k} - \mathbf{r}_{v_k}^{v_{k+1} v_k} \right\|_2^2 + \kappa \left\| \hat{\boldsymbol{\psi}}_{v_k v_{k+1}} - \boldsymbol{\psi}_{v_k v_{k+1}} \right\|_2^2 \tag{57}$$

3.4.3 ESP-VO

ESP-VO [42], with the architecture shown in Figure 11, is an extension of DeepVO with the addition of SE(3) layer and uncertainty estimation. ESP-VO replaces the final 6 dimensional FCN layer with a 128 dimensional FCN layer followed by a 12 dimensional FCN layer. The 12 dimensions represents the 6DoF relative motion plus uncertainties associated with the estimation. Then, 6 outputs that corresponds to the motion estimation as passed to the SE(3) layer. The SE(3) layer has no parameters, it simply concatenates the relative pose $(\hat{\boldsymbol{\psi}}_{v_k v_{k+1}}, \hat{\mathbf{r}}_{v_k}^{v_{k+1} v_k})$ to absolute pose $(\hat{\mathbf{q}}_{i v_k}, \hat{\mathbf{r}}_i^{v_k i})$ producing the latest absolute pose estimate $(\hat{\mathbf{q}}_{i v_{k+1}}, \hat{\mathbf{r}}_i^{v_{k+1} i})$. Unit quaternions are used to represent absolute rotations instead of Euler angles to avoid gimbal locks and angle wrappings. The equations for the SE(3) layers

are described in 58. $\mathbf{q}\{\cdot\}$ operator converts the Euler angle to equivalent rotation in unit quaternion, $\mathbf{q}^*$ is the quaternion conjugate, and $\otimes$ indicates quaternion multiplication.

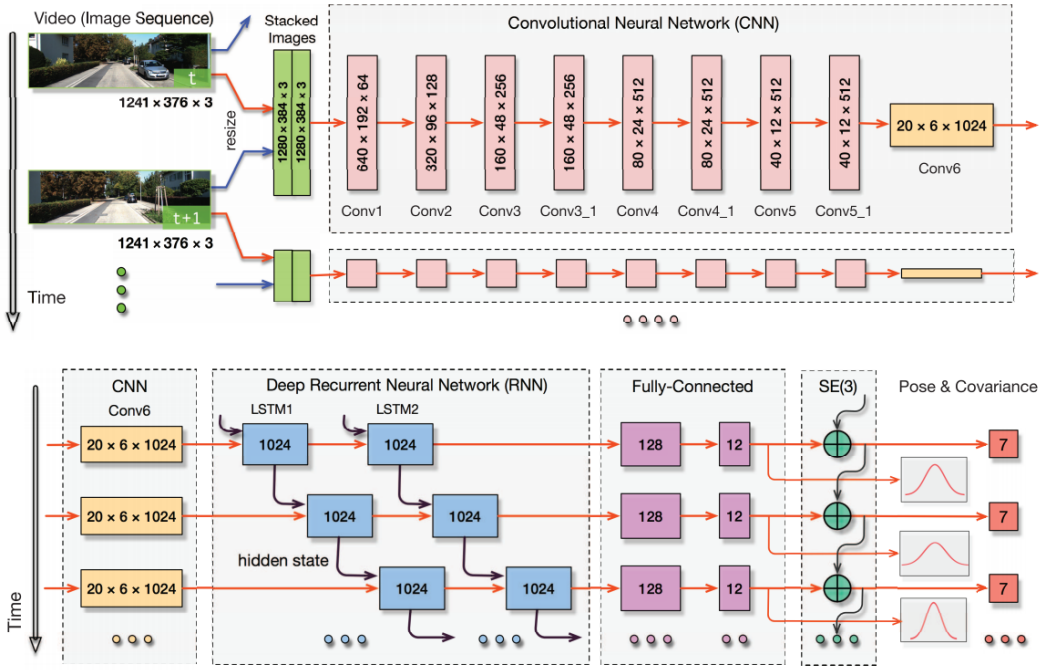


Figure 11: ESP-VO architecture, image retrieved from [42]

$$\begin{aligned}\hat{\mathbf{q}}_{iv_{k+1}} &= \hat{\mathbf{q}}_{iv_k} \otimes \mathbf{q}\{\hat{\boldsymbol{\psi}}_{v_k v_{k+1}}\} \\ \hat{\mathbf{r}}_i^{v_{k+1}i} &= \hat{\mathbf{r}}_i^{v_k i} + \hat{\mathbf{q}}_{iv_k} \otimes \hat{\mathbf{r}}_{v_k}^{v_{k+1}v_k} \otimes \hat{\mathbf{q}}_{iv_k}^*\end{aligned} \tag{58}$$

It is difficult to obtain ground truth values to train uncertainties. Instead, the relative pose uncertainties are trained directly through a modified form the MSE loss function. Consider a multivariate Gaussian distribution shown in Equation 59, where $f(\mathbf{x};\boldsymbol{\theta})$ is a network with parameters $\boldsymbol{\theta}$, $\mathbf{x}$ is the input $\mathbf{y}$ is the ground truth, and $\mathbf{R}$ is the associated error covariance.

$$p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) = \frac{\exp(-\frac{1}{2}(f(\mathbf{x};\boldsymbol{\theta})-\mathbf{y})^T\mathbf{R}^{-1}(f(\mathbf{x};\boldsymbol{\theta})-\mathbf{y}))}{\sqrt{(2\pi)^k|\mathbf{R}|}} \tag{59}$$

The training process optimizes on $\boldsymbol{\theta}$ which maximizes the probability in 59 by minimizing its negative log likelihood as described in 60. In MSE, $\mathbf{R}$ is a constant value that does not affect the output of the optimization. However, if $\mathbf{R}$ is predicted by the network, the loss function can be used for the network to predict the uncertainty associated with its estimate. The positive semi-definiteness of $\mathbf{R}$ is ensured by using Cholesky decomposition, where $\mathbf{R} = \mathbf{L}\mathbf{L}^T$ where $\mathbf{L}$ is a lower triangular matrix. This is further simplified by assuming the output dimensions are independent of each other, making $\mathbf{R}$ a diagonal matrix. Thus, $|\mathbf{R}|$ is simply the product of its diagonals.

$$
\begin{aligned}
\boldsymbol{\theta}^* &= \underset{\theta}{\operatorname{argmax}}\ p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) \\
&= \underset{\theta}{\operatorname{argmin}}\ -\log p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) \\
&= \underset{\theta}{\operatorname{argmin}}\ \log|\mathbf{R}| + (f(\mathbf{x};\boldsymbol{\theta}) - \mathbf{y})^T \mathbf{R}^{-1} (f(\mathbf{x};\boldsymbol{\theta}) - \mathbf{y})
\end{aligned} \tag{60}
$$

To train ESP-VO, two loss functions are used. One loss function 61 is applied to optimize the relative pose outputs and their uncertainties. Here, $\mathbf{z}_k = [\boldsymbol{\psi}^T_{v_k v_k+1}\ \mathbf{r}^{v_{k+1} v_k T}_{v_k}]^T$ and its covariance $\mathbf{R}_k$ are obtained from outputs of the last FCN layer. Another loss function 62 is used to optimize absolute poses of the entire training sample trajectory, which is applied on the outputs of the SE(3) layer. The authors shown the combination of two losses improves convergence to low error estimates.

$$
\boldsymbol{\theta}^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{j=1}^{N} \sum_{k=1}^{t} \log|\mathbf{R}_k| + (\hat{\mathbf{z}}_k - \mathbf{z}_k)^T \mathbf{R}_k^{-1} (\hat{\mathbf{z}}_k - \mathbf{z}_k) \tag{61}
$$

$$
\boldsymbol{\theta}^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{j=1}^{N} \sum_{k=1}^{t} \left\| \hat{\mathbf{r}}_i^{v_k i} - \mathbf{r}_i^{v_k i} \right\|_2^2 + \kappa \left\| \hat{\mathbf{q}}_{i v_k} - \mathbf{q}_{i v_k} \right\|_2^2 \tag{62}
$$

4 Method

The proposed architecture E2E-VIO combines aspects of ESP-VO with a robocentric error state Kalman filter (RC-KF). A RC-KF layer is used in place of the SE(3) layer to fuse estimates from the neural network with predictions made from IMU. The physical IMU models are used for the prediction step of the RC-KF, and the relative pose from the neural network is used for the RC-KF update step which corrects the errors accumulated in the prediction step. We experimented with two variations of the architecture, one with a simple fusion of vanilla DeepVO with IMU through an RC-KF shown in Figure 12 with the black connections only, and the other with a hybrid recurrent unit where predicted states from the RC-KF is concatenated with the CNN features and fed into a RNN to learn additional dynamics from the IMU, shown with the red connections in Figure 12.

4.1 Neural Network Frontend

The Some modifications were made to the ESP-VO architecture for integration with RC-KF. The network outputs relative poses $\mathbf{z}_k = [\tilde{\boldsymbol{\phi}}^T_{r_k v_{k+1}}\ \tilde{\mathbf{r}}^{v_{k+1} r_k T}_{r_k}]^T$ and the associated uncertainty quantities $\mathbf{w}_k = [\mathbf{w}^T_{\mathbf{r}_k}\ \mathbf{w}^T_{\boldsymbol{\phi}_k}]^T$. the rotation output at the at the final FCN layer is parameterized as rotation vector $\boldsymbol{\phi} \in \mathbb{R}^3$. Rotation vector can be easily converted to rotation matrix through the exponential map as in Equation 6. Second, to ensure the positive semi-definiteness of the covariance output, we applied equation 63 elementwise to $\mathbf{w}_k$ as first proposed in [4]. w_i is each element of the vector $\mathbf{w}_k$, σ_0 is the initial guess of the noise standard deviation, and $\beta \in \mathbb{R}_{>0}$ is a tuning parameter. The value of tanh (Figure 13) is bounded between -1 and 1 and by changing β, we can tune how much orders of magnitude σ^2 can deviate from σ_0^2, this allows us to set a lower and upper bounds to reasonable values of σ^2. This ability to set the bounds greatly improved stability during training which will be discussed in Section 5. Another advantage is that this formulation keeps $\mathbf{w}_k$ small

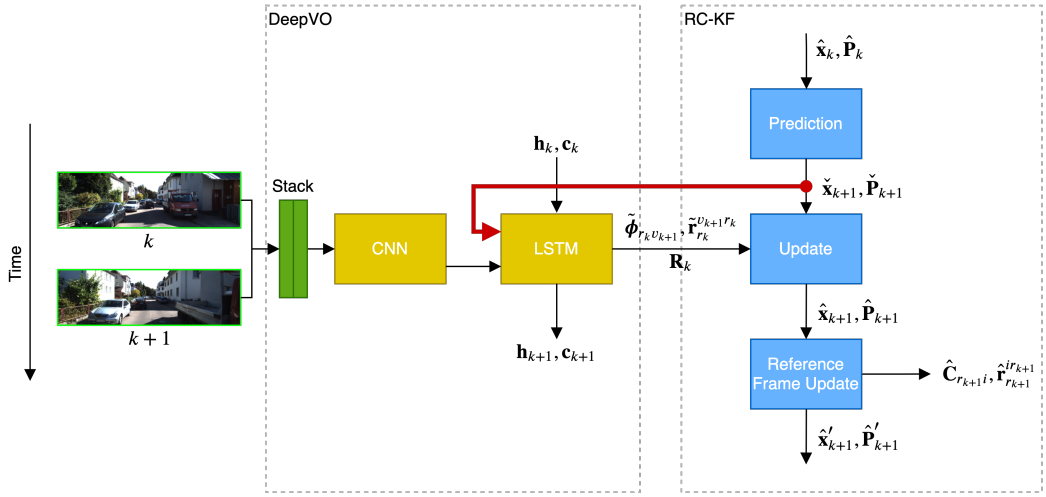


Figure 12: E2E-VIO architecture, with the hybrid connection shown in red

since the magnitude of the gradient of tanh approaches zero as w_i approaches $+\infty$ and $-\infty$. When using Equation 61 for training, the error is large at the start of the training process, thus the covariance $\mathbf{R}$ in Equation 69 will be driven to large values which means that network contains large weights and biases which leads to poor generalization. We keep the assumption and the translation and rotation on each axis are independent of each other, thus $\mathbf{\Sigma}_k = \text{diag}(\boldsymbol{\sigma}_0^2 10^{\beta \tanh(\mathbf{w}_k)})$.

$$\sigma^2 = \sigma_0^2 10^{\beta \tanh(w_i)} \tag{63}$$

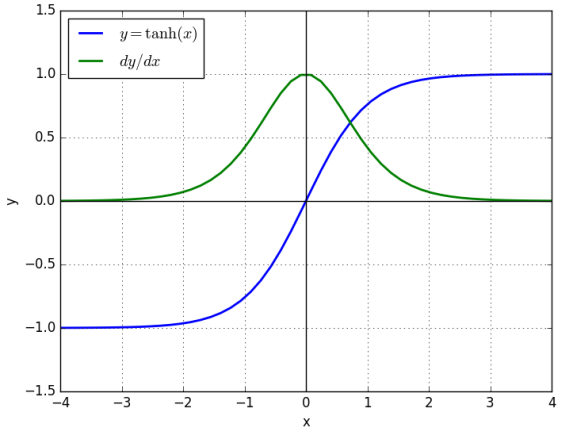


Figure 13: tanh function and its derivative

4.2 RC-KF Update

RC-KF uses relative pose for frame from t_k to t_{k+1} as part of its states, this provides a natural way to fuse relative pose measurements from the neural network. If absolute poses where used to update the RC-KF, the measurement uncertainty would have grow unbounded, which could cause numerical problems. In comparison, relative pose measurements would have bounded uncertainty. The measurements from the network is $\mathbf{z}_k = [\tilde{\boldsymbol{\phi}}^T_{r_k v_{k+1}} \ \tilde{\mathbf{r}}^{v_{k+1} r_k T}_{r_k}]^T$ and the measurement residual is shown in Equation 64. However, we cannot use this residual equation as it is as part of the network. Backpropagation requires network to be differential at all times, the derivative of log mapping $\ln(\cdot)$ for $\mathbb{SO}(3)$ is not defined at π. This is problematic since the network outputs large errors at the start of the training and the term $\exp(\tilde{\boldsymbol{\phi}}^{\wedge}_{r_k v_{k+1}})\mathbf{C}^T_{r_k v_{k+1}}$ may be near π. The circumvent this problem, we used first order Baker-Campbell-Hausdorff (BCH) formula shown in Equation 65 which approximates the log mapping as subtraction of two rotational vectors. This approximation is valid since the rotation between frame t_k and t_{k+1} is small.

$$\boldsymbol{\epsilon}_{k+1} = \begin{bmatrix} \boldsymbol{\epsilon}_\theta \\ \boldsymbol{\epsilon}_r \end{bmatrix} = \begin{bmatrix} \ln(\exp(\tilde{\boldsymbol{\phi}}^{\wedge}_{r_k v_{k+1}})\mathbf{C}^T_{r_k v_{k+1}})^{\vee} \\ \tilde{\mathbf{r}}^{v_{k+1} r_k}_{r_k} - \mathbf{r}^{v_{k+1} r_k}_{r_k} \end{bmatrix} = \begin{bmatrix} \ln(\exp(\tilde{\boldsymbol{\phi}}^{\wedge}_{r_k v_{k+1}})\exp(\boldsymbol{\phi}^{\wedge}_{r_k v_{k+1}})^T)^{\vee} \\ \tilde{\mathbf{r}}^{v_{k+1} r_k}_{r_k} - \mathbf{r}^{v_{k+1} r_k}_{r_k} \end{bmatrix} \tag{64}$$

$$\boldsymbol{\epsilon}_\theta \approx \tilde{\boldsymbol{\phi}}^{\wedge}_{r_k v_{k+1}} - \boldsymbol{\phi}^{\wedge}_{r_k v_{k+1}} \tag{65}$$

We then differentiate Equation 64 with respect to the error states to find the measurement Jacobian $\mathbf{H}_{k+1}$. The derivations are shown in Equation 66 and Equation 67, and the final $\mathbf{H}_{k+1}$ is shown in Equation 68. The best estimate of the error state computed according to Equation 45, and the subsequent error injection reference frame update must be performed as well.

$$\begin{aligned} \boldsymbol{\epsilon}_\theta &\approx \tilde{\boldsymbol{\phi}}^{\wedge}_{r_k v_{k+1}} - \ln(\mathbf{C}_{r_k v_{k+1}}) \\ &\approx \tilde{\boldsymbol{\phi}}^{\wedge}_{r_k v_{k+1}} - \ln(\hat{\mathbf{C}}_{r_k v_{k+1}} \exp(\delta\boldsymbol{\phi}^{\wedge}_{r_k v_{k+1}})) \\ &\approx \underbrace{\tilde{\boldsymbol{\phi}}^{\wedge}_{r_k v_{k+1}} - \hat{\boldsymbol{\phi}}^{\wedge}_{r_k v_{k+1}}}_{\bar{\epsilon}_\theta} - \mathbf{J}_r(\boldsymbol{\phi}^{\wedge}_{r_k v_{k+1}})^{-1}\delta\boldsymbol{\phi}_{r_k v_{k+1}} \end{aligned} \tag{66}$$

$$\begin{aligned} \boldsymbol{\epsilon}_r &= \tilde{\mathbf{r}}^{v_{k+1} r_k}_{r_k} - \mathbf{r}^{v_{k+1} r_k}_{r_k} \\ &= \underbrace{\tilde{\mathbf{r}}^{v_{k+1} r_k}_{r_k} - \hat{\mathbf{r}}^{v_{k+1} r_k}_{r_k}}_{\bar{\epsilon}_r} - \delta\mathbf{r}^{v_{k+1} r_k}_{r_k} \end{aligned} \tag{67}$$

$$\begin{aligned} \mathbf{H}_{k+1} &= \frac{\partial \boldsymbol{\epsilon}_{k+1}}{\partial \delta\mathbf{x}_{k+1}} \\ \mathbf{H}_{k+1} &= \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{J}(-\check{\boldsymbol{\phi}}_{r_k v_{k+1}})^{-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \end{aligned} \tag{68}$$

5 Training

We Adopted the loss function from ESP-VO with some minor changes. As in ESP-VO, we applied a loss on the relative poses and uncertainties produced from the neural network. We found that the training becomes more stable and converges more rapidly when the supervisory signal is applied to the intermediate result. This loss function, C_1, is shown in

Equation 69. We also experimented with a simple MSE as supervisory signal, but we found that the final result is worse and the network was not able to provide a good uncertainty for the measurement its provides to RC-KF. In addition, we found that with $\mathbf{R}_k$ being a diagonal matrix, as diagonal elements of $\mathbf{R}_k$ approaches zero as $\log|\mathbf{R}_k|$ approaches $-\infty$, and diagonal elements of $\mathbf{R}_k^{-1}$ approaches $+\infty$, which makes it problematic in terms of numerical stability as training progress and the network becomes increasingly certain of its estimates. We found using 63 in forming $\mathbf{R}$ improves training stability, and it works better in that regard than simply adding a small epsilon to the diagonal.

$$C_1 = \frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{t}\log|\mathbf{R}_k| + (\hat{\mathbf{z}}_k - \mathbf{z}_k)^T\mathbf{R}_k^{-1}(\hat{\mathbf{z}}_k - \mathbf{z}_k) \tag{69}$$

The cost function C_2 is applied to minimize the best estimate states $\hat{\mathbf{C}}_{r_k i}^T$ and $\hat{\mathbf{r}}_{r_k}^{ir_k T}$ which represents the transformation of the inertial frame $\mathcal{F}_i$ with respect to the reference frame $\mathcal{F}_{r_k}$. In contrast to Equation 62 used in ESP-VO, we used Frobenius norm on the difference between identity and $\hat{\mathbf{C}}_{v_k i}^T\mathbf{C}_{v_k i}$ for computing difference in orientation between the estimate and ground truth. First, compared to ESP-VO, using rotation matrices is more ideal since it does not suffer from quaternion flips when error is large at the start of the training process. Second, the Frobenius norm allows us to avoid using log mapping which is not differentiable at π, while still maintaining a valid distance measurement that we would use to minimize on. Note that when $\hat{\mathbf{C}}_{v_k i}$ is close to $\mathbf{C}_{v_k i}$, $\mathbf{I} - \hat{\mathbf{C}}_{v_k i}^T\mathbf{C}_{v_k i}$ approximates $\ln(\hat{\mathbf{C}}_{v_k i}^T\mathbf{C}_{v_k i})$. In addition to training the CNN and LSTM weights and biases, we also train IMU noise covariances from Equation 41 using Equation 63. The weights for the IMU noise covariances is a 12-dimensional vector. The supervisory signal for training the IMU noise covariances come from Equation 70. We use $\kappa_2 = 500$ to scale the rotational portion of the loss. The two loss functions are summed for training.

$$C_2 = \frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{t}\left\|\hat{\mathbf{r}}_{v_k}^{iv_k} - \mathbf{r}_{v_k}^{iv_k}\right\|_2^2 + \kappa_2\left\|\mathbf{I} - \hat{\mathbf{C}}_{v_k i}^T\mathbf{C}_{v_k i}\right\|_F^2 \tag{70}$$

During training, we break down a long sequence into short sub-sequences of 32 timesteps with steps of 10 as shown in Figure 14. We randomly sample sub-sequences and combine them into a single batch of size 16. An epoch of training iterates through all sub-sequences. Instead of initializing all LSTM states at zero, the final LSTM states at the end of each sub-sequence is kept after a forward pass of the network, and used in the connecting sub-sequence. To combat overfitting, We applied data augmentation techniques such as random noise in brightness, contrast, saturation and hue as well as left-right image flipping. In addition, dropouts and batch normalization were used as part of the network. For training vision-only comparisons, we also included traveling backwards in time as a form of augmentation to prevent the network overfitting to a constant forward motion in a driving dataset. During training, we monitor the dataset specific evaluation criteria at the completion of each epoch on the test set and saved the weights for the best results. The networked is trained for a maximum of 400 epochs and a batch size of 16 on a single NVIDIA Titan Xp GPU.

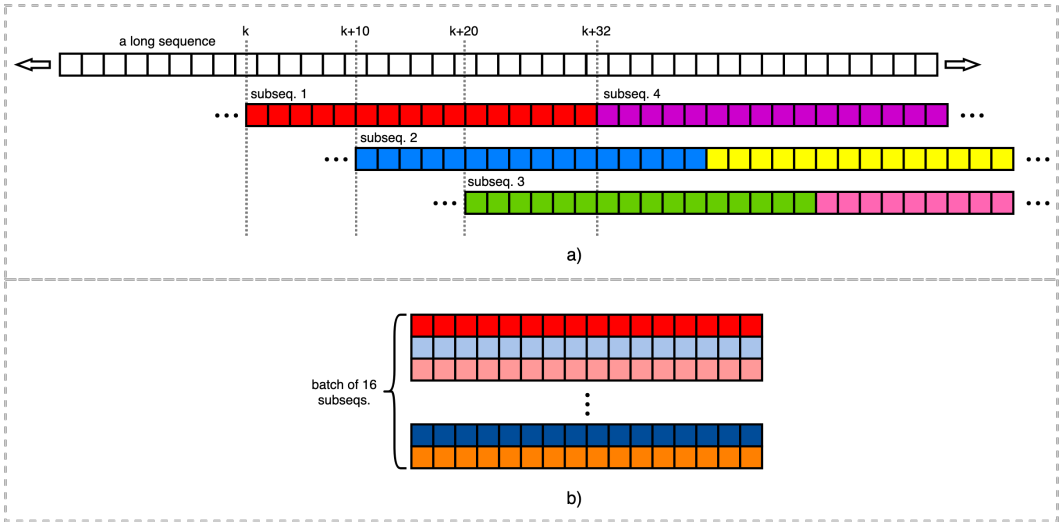


Figure 14: Forming sub-sequences from long data data sequences, a) how the sub-sequences are selected, b) forming sub-sequences into a batch of data for training

6 Experimental Results

6.1 KITTI

We tested the proposed method on the KITTI VO dataset [18]. The dataset features a outdoor driving environment on urban road with speeds varying from completely stopped to 90km/h shown in Figure 15. It provides camera and LiDAR data at 10Hz and IMU data at 100Hz. The Odometry data has 22 sequences, however, only 10 sequences have the corresponding raw datasets that contain IMU data. We used sequences 00, 01, 02, 05, 07, and 09 for training and sequences 04, 06, 07, and 10 for testing. The evaluation metric used is the standard KITTI metric computes the translation and rotation error per unit of distance traveled, evaluated at 100, 200, 300, 400, 500, 600, 700, and 800 meters. The units for translation errors t_{err} are in percentage(%), and the units for rotational errors r_{err} are in degree per 100m ($\circ/100m$). The mean errors for these sequences are computed by taking the mean of all 100 to 800 segments in all the testing sequences, thus longer sequences such as 07 and 10 have more influence on the average error.

Figure 15: An example from the KITTI dataset, taken from sequence 00

In KITTI, the camera and IMU data are not tightly synchronized, and the IMU data is interpolated at camera timesteps. Sequences 00, 02, and 05 have IMU outages for over 2 seconds. To cope with this, we broke these sequences at these outages into smaller sequences for training. In many sequences, there are also duplicate data and data that are not temporally monotonic, these pieces of invalid data are removed in preprocessing. Ground truth values are used to initialize the RC-KF during training and testing. KITTI provides earth referenced orientation in the East-North-Up (ENU) frame, this can be used to rotate the gravity vector of $[0\ 0\ g]^T$ to the start of each sequence for use as initial gravity g_{r_0}. The KITTI dataset also provides ground truth velocities in the body frame which are used to initialize the RC-KF velocities directly. The initial accelerometer and gyroscope biases are assumed to be zero since they are small, and it is expected the biases estimates with measurement updates from the network.

Table 3 shows the experiments performed on the KITTI dataset under the scenario of IMU only, vision only, vanilla E2E-VIO, and hybrid E2E-VIO, the plots are shown in Figures 16, 17, 18, and 19. These experiments are conducted with fixed IMU parameters as outlined in Table 2. The IMU only estimate is obtained by integrate the accelerometer and gyroscope measurements from the IMU. Vision only implementation is obtained by concatenating relative pose transformations obtained by DeepVO. Vanilla implementation of E2E-VIO uses the architecture described in Figure 12 with the black block connections. Hybrid implementation of E2E-VIO uses architecture described in Figure 12 with the additional red connection.

Table 2: RC-KF parameters used for KITTI

Parameter	Description	Value
σ_w	gyroscope noise s.d.	3.16×10^{-4}
$\sigma_{\mathbf{b}_\omega}$	gyroscope bias noise s.d.	3.16×10^{-4}
σ_a	accelerometer noise s.d.	1×10^{-1}
$\sigma_{\mathbf{b}_a}$	accelerometer bias noise s.d.	3.16×10^{-2}
$\sigma_{\mathbf{g}_{r_0}}$	initial gravity uncertainty s.d.	1×10^{-4}
$\sigma_{\mathbf{v}_0}$	initial velocity uncertainty s.d.	1×10^{-2}
$\sigma_{\mathbf{b}_{\omega_0}}$	initial gyroscope bias uncertainty s.d.	1×10^{-8}
$\sigma_{\mathbf{b}_{a_0}}$	initial accelerometer bias uncertainty s.d.	1×10^{-1}

Table 3: KITTI Results with IMU only, vision only, vanilla, and hybrid implementation

Seq.	IMU Only		Vision Only		Vanilla		Hybrid	
	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}
04	1.5055	**0.0036**	7.0586	1.8942	**0.6960**	0.0076	0.7305	0.0066
06	6.3787	**0.1468**	7.4296	2.0433	2.0744	0.1557	**2.0254**	0.1556
07	14.1284	**0.2634**	4.7598	2.9869	**1.2549**	0.3072	1.3150	0.3059
10	14.0479	**0.2033**	9.0501	3.8747	1.1960	0.2522	**1.1704**	0.2521
mean	10.5499	**0.1875**	7.3503	2.8648	1.5467	0.2177	**1.5332**	0.2177

Inspecting results, it is clear that both vanilla and hybrid implementations of E2E-VIO significantly outperformed the IMU only and vision only implementations. The IMU only implementation suffers from significant drift, this is especially evident from Figure 18 where

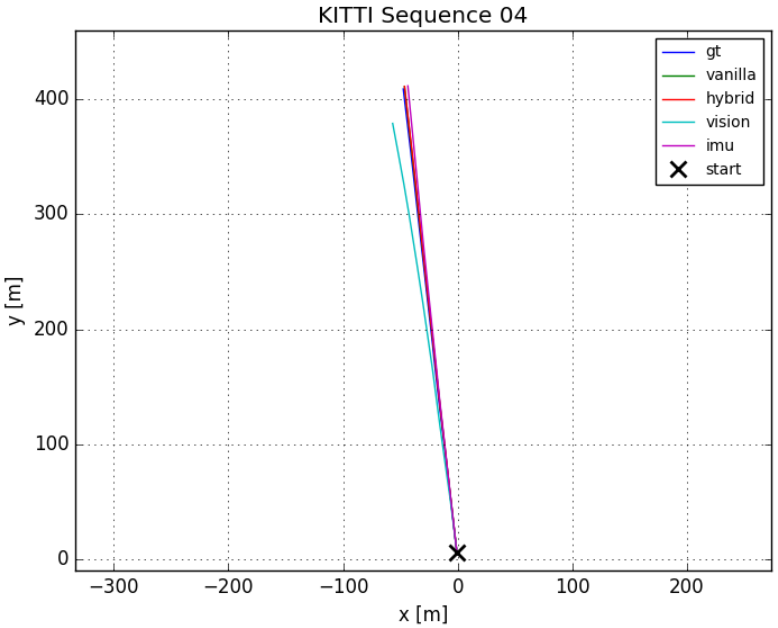


Figure 16: KITTI sequence 04 results

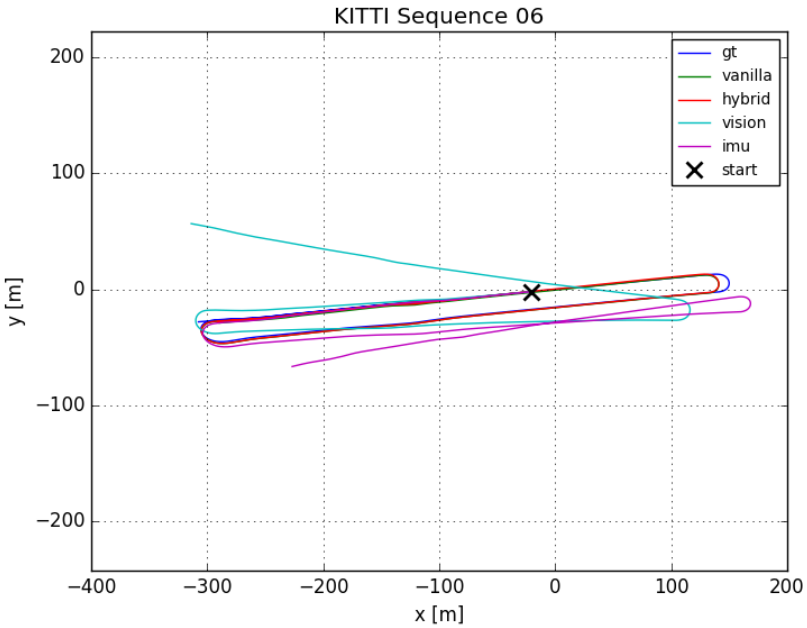


Figure 17: KITTI sequence 06 results

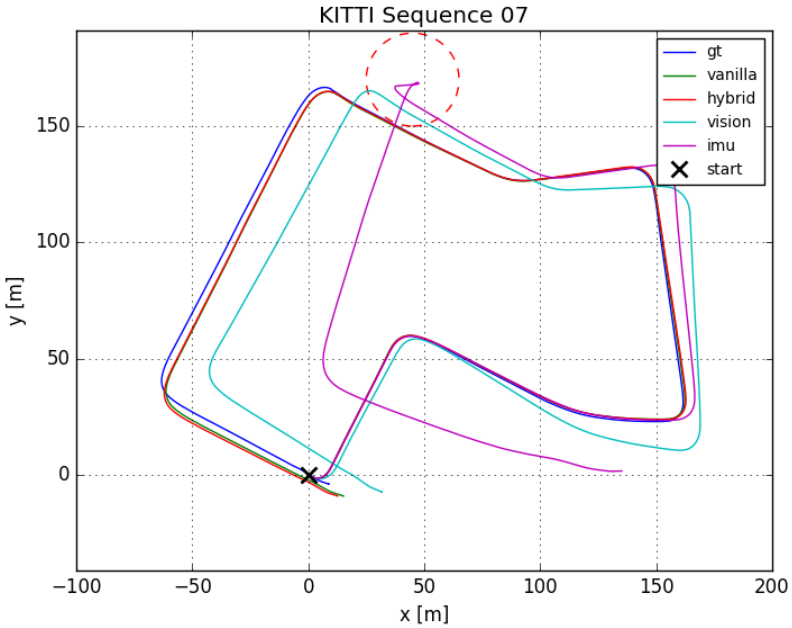


Figure 18: KITTI sequence 07 results, large drift in IMU only circled in red

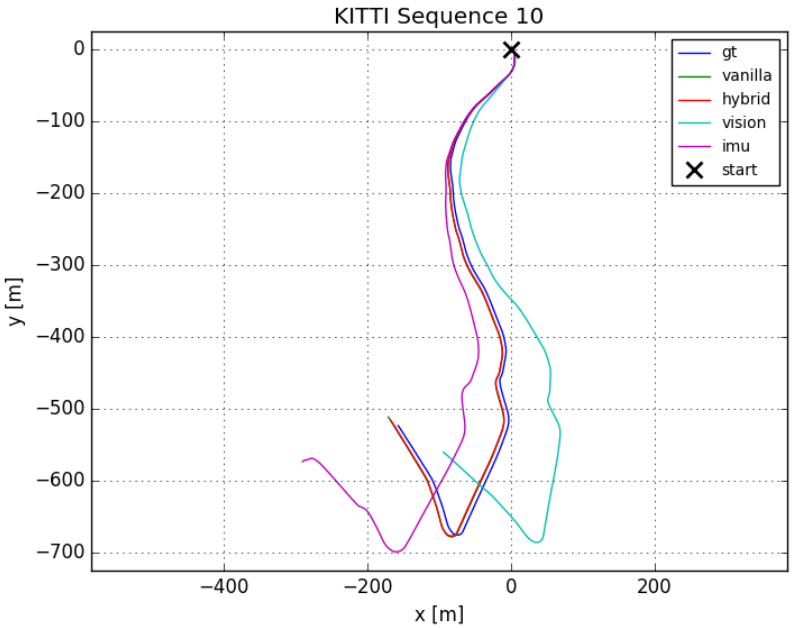


Figure 19: KITTI sequence 10 results

Table 4: KITTI Results with different covariance training

Seq.	Vanilla		MSE C_1		Train IMU Covar.	
	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}
04	**0.6960**	**0.0076**	8.7005	0.0948	1.2252	0.0080
06	**2.0744**	**0.1557**	11.9905	0.7657	5.5659	0.1563
07	**1.2549**	0.3072	4.3924	1.0836	2.6167	**0.3043**
10	**1.1960**	0.2522	7.7494	0.801	2.8858	**0.2507**
mean	**1.5467**	**0.2177**	8.7253	0.8251	3.8487	0.2177

IMU only estimate experienced significant drift, this is circled in the the figure. The drift occurred because the vehicle stopped for a few seconds and the IMU continues integrating with its noise and biases uncorrected for. This becomes very problematic since the position integration is the result of three nested numerical integrations shown in Equation 36, this caused the position estimate to drift much more rapidly in comparison to orientation estimate as shown in Figure 20. In contrast to IMU only, the vision only implementation was able to correctly estimate motion of the stopped vehicle. However, we can also see its errors slowly accumulating as the vehicle travels. This is especially evident during sharp turns in sequence 06. In Figure 17, we can observe the rotational error increases dramatically with each of the two hairpin turns. The qualitative observations are supported in Table 3, where the vision only implementation has smaller and more consistent translation errors among the sequences and significantly worse orientation errors. Fusing vision estimate from the network with the IMU predictions corrects for the accumulated error from the noise and biases. It also makes IMU biases observable which improves subsequent IMU predictions. We can see from vanilla and hybrid results, the translation error is greatly reduced in comparison to the non-fused methods. In comparison to IMU only, the fused methods have slightly worse orientation error. This is because the IMU used in the KITTI dataset is very good with little noise and biases, and by providing orientation information from vision which has over one order of magnitude more error, the estimation accuracy may decrease if the corresponding covariances are not set to optimal values. In such case, IMU only orientation estimate does not benefit from additional vision information. The hybrid implementation performs marginally better than the vanilla implementation. The small improvement does not conclusively determine the helpfulness of learning with additional information from RC-KF prediction. However, [5] demonstrated the ability of the network to gain information from IMU data, and we believe additional work can be performed with the hybrid implementation to improve results.

Table 4 showed the results on experimenting with different methods of learning measurement and IMU covariances. In the MSE C_1 experiment, the C_1 loss shown in 69 is replaced with a simple MSE. The covariance is still obtained in the same manner from the network as the vanilla implementation, but the supervisory signal comes from loss function C_2 shown in Equation 70 instead of from C_1. From the results, we can see the using MSE C_1 performs significantly worse, this shows that using Equation 69 is able to guide the network learn better covariances for the RC-KF update. We also trained IMU covariances instead of keeping them to a reasonable fixed value. We set the initial covariance σ_0 in Equation 63 to 0.01 and anticipated the network will learn a optimal IMU noise covariance, which should produce a better result in comparison to the vanilla. In actuality, as shown in Table 4, training IMU covariance has comparable orientation error in comparison to vanilla, but has much

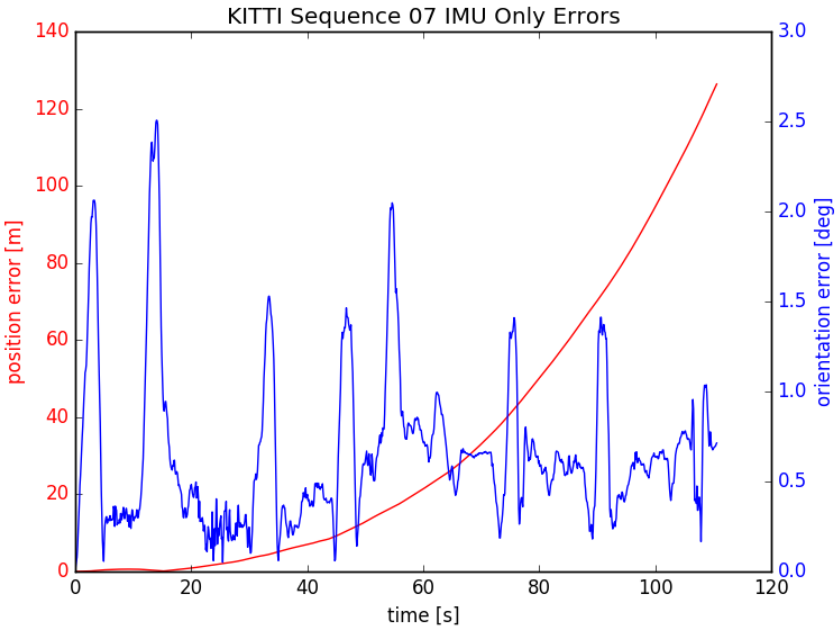


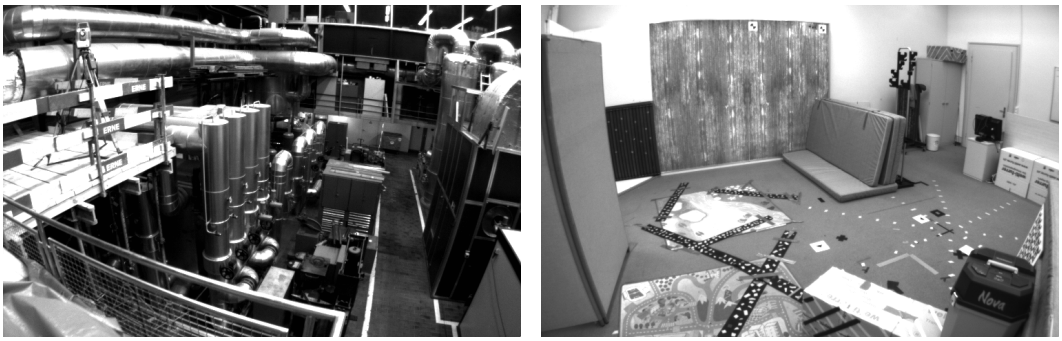
Figure 20: KITTI sequence 07 IMU only errors

higher translation error. This occurred because the network would overfit to the training data producing low error measurement updates to the RC-KF, which allows the optimal IMU covariance to increase. However, when evaluating on the test data, the measurement error from the network becomes larger, which would need different IMU noise covariances to obtain the best results. This problem can be mitigated by reducing overfitting through improving network architecture and hyperparameter selection.

6.2 EUROC

In additional to the KITTI dataset, we also tested the proposed method on the EUROC drone dataset. The EUROC dataset features a drone flying in indoor environments shown in Figure 21. It provides hardware synchronized camera and IMU data at 20Hz and 200 Hz respectively. The dataset is divided into three scenes: Machine Hall, Vicon Room 1, and Vicon Room 2. The Machine Hall has 5 sequences: MH_01, MH_02, MH_03, MH_04, and MH_05. The Vicon Rooms each have 3 sequences, and they are named V1_01 , V1_02, V1_03, V2_01, V2_02, V2_03. We used MH_01 to MH_04, V1_01, V1_02, and V2_01 for training, and MH_05, V1_03, and V2_02 for testing. Sequence V2_03 is omitted due issues with some image data not synchronized with the IMU at 20Hz. To speed up the training and testing process, the image data is downsampled to 10Hz and the IMU data is downsampled to 100Hz. The most common evaluation metrics used for the EUROC dataset is RMSE of the translation portion of the absolute trajectory error. This is obtained by calculating the distance at each timestep between the estimated trajectory and ground truth trajectory, then computing the RMSE. The estimated trajectory is also aligned to the ground truth

before error metric is calculated. EUROC images are single channel grayscale images. Due to the hierarchical structure of CNN, weight and architectural modifications in the shallow layers invalidates the pre-trained weights in the deeper layers. Thus, in order to use the same weights from FlowNet for initialization, the same channel is stacked 3 times channel wise before using it as an into to the network, effectively recreating the same grayscale image in RGB.



(a) (b)

Figure 21: Examples of scenes from the EUROC dataset, a) Machine Hall, b) Vicon Room 2

EUROC provides post-processed ground truth for position, velocity, orientation, and gyroscope and accelerometer biases. Ground truth velocities can be used for initialization, but it is also possible to initialize with zero velocity before the drone takes flight. EUROC IMU has large biases in its measurements. The initial gyroscope bias can be easily obtained while the drone is still as it is simply the averaged gyroscope readings. The accelerometer bias is difficult to isolate due to the presence of the gravity vector, thus we assume it to be zero at the start, give it a high initial covariance, and allow the RC-KF estimate it with vision measurements. Unlike the KITTI dataset which provides ground truth orientation in the ENU frame, the EUROC dataset provides ground truth relative to the Leica laser tracker or the Vicon motion capturing system. As a result, the initial gravity vector $\mathbf{g}_{r_0}$ cannot be obtained directly from ground truth orientation. Instead, $\mathbf{g}_{r_0}$ is obtained through a simple linear least squared optimization given the IMU data and the ground truth poses. Consider Equation 71 which is a modified version of Equation 35 with all quantities defined in the very first reference frame r_0. Here, between two times from t_k to t_{k+1} indicates integration of an arbitrary amount of IMU data which is chosen to be 10, Δt is the time elapsed from t_k to t_{k+1}. Equation 71 is rearranged to Equation 72, and we can set up the least squared problems as shown in Equation 73 with $\check{\mathbf{z}}_k = [\check{\boldsymbol{\alpha}}_{r_0}^{v_k r_0 T} \ \check{\boldsymbol{\beta}}_{r_0}^{v_k r_0 T}]^T$ and $\mathbf{H}_k = \frac{\partial \mathbf{z}_k}{\partial \mathbf{g}_{r_0}}$.

$$\begin{aligned}
\mathbf{C}_{v_k r_0} \mathbf{r}_{r_0}^{v_{k+1} r_0} &= \mathbf{C}_{v_k r_0} (\mathbf{r}_{r_0}^{v_k r_0} + \mathbf{v}_{r_0}^{v_k r_0} \Delta t_k - \frac{1}{2} \mathbf{g}_{r_0} \Delta t_k^2) + \boldsymbol{\alpha}_{r_0}^{v_k r_0} \\
\mathbf{C}_{v_k r_0} \mathbf{v}_{r_0}^{v_{k+1} r_0} &= \mathbf{C}_{v_k r_0} (\mathbf{v}_{r_0}^{v_k r_0} - \mathbf{g}_{r_0} \Delta t_k) + \boldsymbol{\beta}_{r_0}^{v_k r_0}
\end{aligned} \tag{71}$$

$$\begin{aligned}
\boldsymbol{\alpha}_{r_0}^{v_k r_0} &= \mathbf{C}_{v_k r_0} \mathbf{r}_{r_0}^{v_{k+1} r_0} - \mathbf{C}_{v_k r_0} (\mathbf{r}_{r_0}^{v_k r_0} + \mathbf{v}_{r_0}^{v_k r_0} \Delta t_k - \frac{1}{2} \mathbf{g}_{r_0} \Delta t_k^2) \\
\boldsymbol{\beta}_{r_0}^{v_k r_0} &= \mathbf{C}_{v_k r_0} \mathbf{v}_{r_0}^{v_{k+1} r_0} - \mathbf{C}_{v_k r_0} (\mathbf{v}_{r_0}^{v_k r_0} - \mathbf{g}_{r_0} \Delta t_k)
\end{aligned} \tag{72}$$

$$\min_{\mathbf{g}_{r_0}} \|\check{\mathbf{z}}_k - \mathbf{H}_k \mathbf{g}_{r_0}\|_2^2 \tag{73}$$

Table 6 shows the EUROC results tested on the sequences MH_05, V1_03, and V2_02. The plots for overlaid vision only, vanilla and hybrid results are shown in Figure 22, 23, and 24 for each of the test sequences. We also plotted the IMU only results for MH_05 in Figure 25. The selected parameters for vanilla and hybrid results are shown in Table 5. We can observe from the IMU only results that direct integration of IMU have enormous errors due to the significant biases and noises in the IMU which is approximately 0.08 $\mathrm{m/s^2}$ for the accelerometer and 0.15 rad/s for the gyroscope. The accelerometer bias caused the velocity to accumulate, and in turn the caused the position estimation to drift rapidly in one direction as shown in Figure 25. Both vanilla and hybrid methods have performances of similar magnitude. In contrast to results from KITTI, fusing network vision measurements with IMU does not produce conclusively better results. In addition, the results are much worse in comparison to the model-based VIO methods: MSCKF [31] and OKVIS [28].

The poor performance is mainly due two factors. First, unlike the IMU from KITTI which has relatively low noise ans biases, the EUROC IMU has large biases and is relatively noisy. Second, the network setup was not able to learn good measurements to correct for the IMU noise and biases as shown in the Table 6 where the scaled monocular ORB-SLAM outperforms the vision only implementation by nearly two orders of magnitude. With both sensors in the architecture fusing in noisy information, the resulting estimate is unsurprisingly inaccurate. This is in contrast to the result from the KITTI dataset which fuses noisy vision data with low noise IMU data. In a model-based monocular method, after applying the same scaling factor to the trajectory to match the ground truth, the majority of the error is the result of the very slow scale drift. In fact, given scale information in the case of stereo implementation or using learning based methods [29], vision based SLAM methods are have shown to be much more accurate. From Table **??**, we can see that OKVIS has a order of magnitude worse performance to Monocular ORB-SLAM when ORB-SLAM is scaled to match the ground truth. In the case of the EUROC dataset, fusing IMU with vision does not actually improve performance in terms of error metric. Fusing IMU with monocular vision addresses the most urgent short coming of scale ambiguity and scale drift, and improves reliability under fast motion, thus allowing vision and IMU to complement each other. Learning based monocular visual SLAM methods such as [41] and [42] have shown to produce better performance than monocular model-based methods such as ORB-SLAM. However, the majority of error from these methods are not due to scale drift, but rather from the learned model, which reduces the effectiveness of the IMU fusion algorithm.

Table 5: RC-KF parameters used for EUROC

Parameter	Description	Value
σ_w	gyroscope noise s.d.	3.16×10^{-2}
$\sigma_{\mathbf{b}_\omega}$	gyroscope bias noise s.d.	3.16×10^{-3}
σ_a	accelerometer noise s.d.	3.16×10^{-1}
$\sigma_{\mathbf{b}_a}$	accelerometer bias noise s.d.	1×10^{-1}
$\sigma_{\mathbf{g}_{r_0}}$	initial gravity uncertainty s.d.	1×10^{-1}
$\sigma_{\mathbf{v}_{r_0}}$	initial velocity uncertainty s.d.	1×10^{-2}
$\sigma_{\mathbf{b}_{\omega_0}}$	initial gyroscope bias uncertainty s.d.	1×10^{-1}
$\sigma_{\mathbf{b}_{a_0}}$	initial accelerometer bias uncertainty s.d.	1×10^{1}

Table 6: EUROC Results

Seq.	IMU Only	Vision Only	Vanilla	Hybrid	MSCKF[a]	OKVIS[a]	ORB-SLAM Mono[b]
MH_05	73.90	3.03	2.00	2.30	0.48	0.47	0.049
V1_03	66.68	1.75	2.31	4.88	0.67	0.24	Not Completed
V2_02	47.83	2.89	1.98	2.02	1.13	0.29	0.074

[a] Results obtained from [10]
[b] monocular vision only, results scaled to fit ground truth

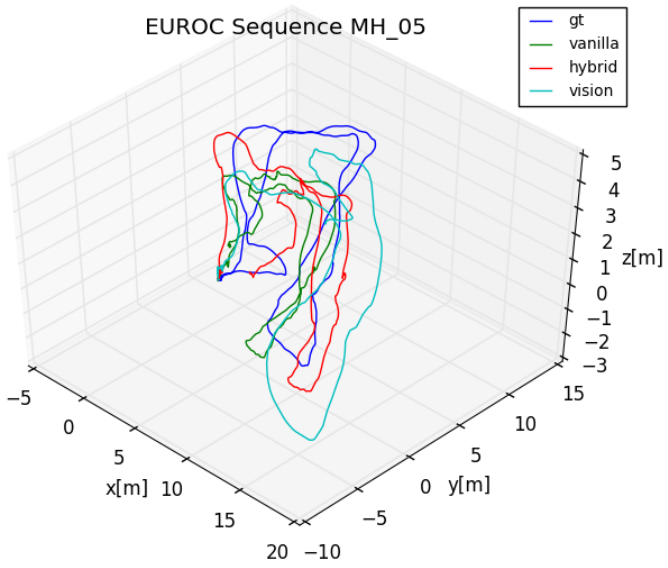


Figure 22: EUROC sequence MH_05 results

7 Summary

In this chapter, we described the details of the proposed end-to-end learning of VIO incorporating the RC-KF algorithmic prior as part of the network architecture. We used RC-KF to propagate nominal and error states from the IMU from one timestep to the next. The measurement update for RC-KF are formulated as relative pose measurements produced from the DeepVO network which uses images as inputs. The entire system is trained end-to-end with two loss functions, one supervises the relative pose measurements and the corresponding covariances, and the other applies to the final absolute poses. We tested the system on the KITTI and EUROC dataset. On the KITTI dataset, we showed that the our fused results are significantly better than results obtained from using only IMU or only vision, the vision measurements from the network is able to correct errors accumulated in the IMU. For the EUROC dataset, we found that vision measurements are not accurate enough to

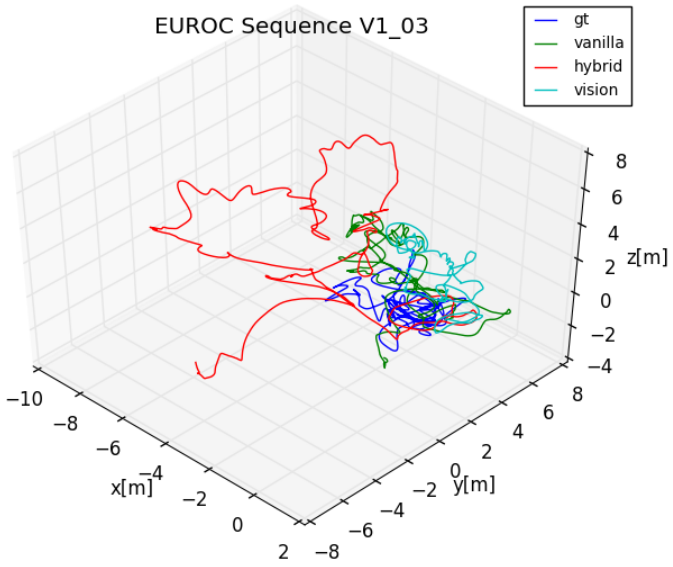


Figure 23: EUROC sequence V1_03 results

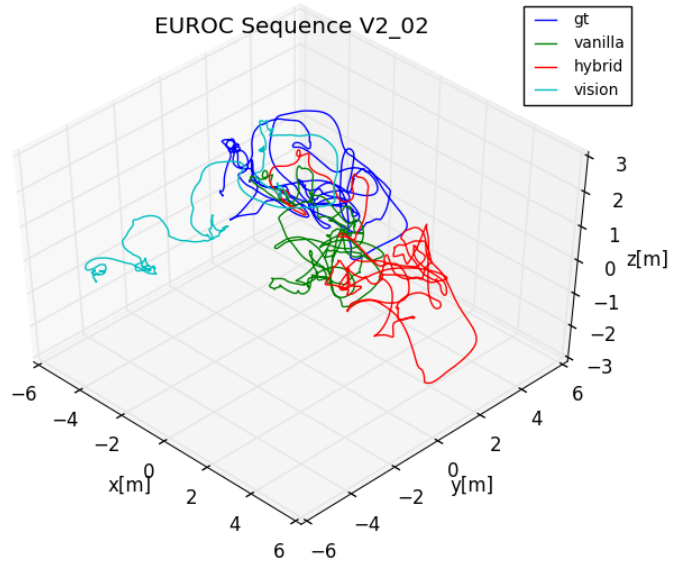


Figure 24: EUROC sequence V2_02 results

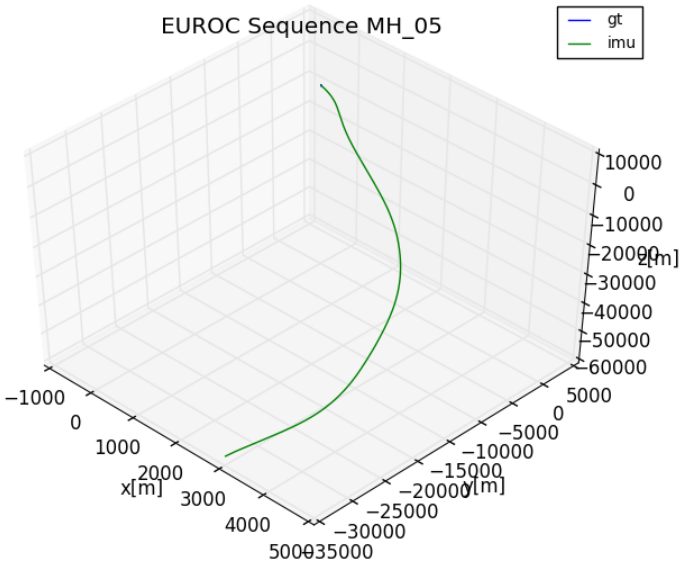


Figure 25: EUROC sequence MH_05 IMU only results

provide sufficient correction for the errors accumulated by a more noisy IMU compared to KITTI, producing unsatisfactory fusion results. Further work are required to improve the vision measurement in terms of accuracy as well as providing a better measurement update model than one presented.

References

[1] Timothy D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2017.

[2] 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 (IROS)*, pages 298–304, Sep. 2015.

[3] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J. Davison. Codeslam - learning a compact, optimisable representation for dense visual slam. In *CVPR*, 2018.

[4] Martin Brossard, Axel Barrau, and Silvère Bonnabel. AI-IMU dead-reckoning. *CoRR*, abs/1904.06064, 2019.

[5] Martin Brossard, Axel Barrau, and Silvère Bonnabel. AI-IMU dead-reckoning. *CoRR*, abs/1904.06064, 2019.

[6] Ronald Clark, Michael Bloesch, Jan Czarnowski, Stefan Leutenegger, and Andrew J. Davison. Learning to solve nonlinear least squares for monocular stereo. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[7] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki 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.

[8] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia. Exploring representation learning with cnns for frame-to-frame ego-motion estimation. *IEEE Robotics and Automation Letters*, 1(1):18–25, Jan 2016.

[9] Andrew J. Davison, Ian D. Reid, Nicholas Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1052–1067, 2007.

[10] 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 (ICRA)*, pages 2502–2509, May 2018.

[11] A. Dosovitskiy, P. Fischer, E. Ilg, P. Husser, 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 (ICCV)*, pages 2758–2766, Dec 2015.

[12] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, March 2018.

[13] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, September 2014.

[14] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *2013 IEEE International Conference on Computer Vision*, pages 1449–1456, Dec 2013.

[15] D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico, and D. Scaramuzza. Vision-based autonomous quadrotor landing on a moving platform. In *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pages 200–207, Oct 2017.

[16] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, Feb 2017.

[17] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, April 2017.

[18] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[20] Tuomas Haarnoja, Anurag Ajay, Sergey Levine, and Pieter Abbeel. Backprop kf: Learning discriminative deterministic state estimators. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4376–4384. Curran Associates, Inc., 2016.

[21] Joel A Hesch, Dimitrios G Kottas, Sean L Bowman, and Stergios I Roumeliotis. Camera-imu-based localization: Observability analysis and consistency improvement. *The International Journal of Robotics Research*, 33(1):182–201, 2014.

[22] Z. Huai and G. Huang. Robocentric visual-inertial odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6319–6326, Oct 2018.

[23] Rico Jonschkowski and Oliver Brock. End-to-end learnable histogram filters. In *Workshop on Deep Learning for Action and Interaction at NIPS*, December 2016.

[24] Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. Differentiable Particle Filters: End-to-End Learning with Algorithmic Priors. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[25] Peter Karkus, David Hsu, and Wee Sun Lee. Particle filter networks with application to visual localization. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 169–178. PMLR, 29–31 Oct 2018.

[26] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[27] Kishore Reddy Konda and Roland Memisevic. Learning visual odometry with a convolutional network. In *VISAPP*, 2015.

[28] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visualinertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[29] Shing Yan Loo, Ali Jahani Amiri, Syamsiah Mashohor, Sai Hong Tang, and Hong Zhang. CNN-SVO: improving the mapping in semi-direct visual odometry using single-image depth prediction. *CoRR*, abs/1810.01011, 2018.

[30] 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*, pages 3923–3929, Nov 2013.

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

[32] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015.

[33] R. Mur-Artal and J. D. Tards. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017.

[34] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327, Nov 2011.

[35] V. Peretroukhin and J. Kelly. Dpc-net: Deep pose correction for visual localization. *IEEE Robotics and Automation Letters*, 3(3):2424–2431, July 2018.

[36] T. Qin, P. Li, and S. Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, Aug 2018.

[37] D. Scaramuzza and F. Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics Automation Magazine*, 18(4):80–92, Dec 2011.

[38] K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6565–6574, July 2017.

[39] Mehmet Turan, Yasin Almalioglu, Helder Araujo, Ender Konukoglu, and Metin Sitti. Deep endovo: A recurrent convolutional neural network (rcnn) based visual odometry approach for endoscopic capsule robots. *Neurocomputing*, 275:1861 – 1870, 2018.

[40] Lukas von Stumberg, Vladyslav C. Usenko, and Daniel Cremers. Direct sparse visual-inertial odometry using dynamic marginalization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2510–2517, 2018.

[41] 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 (ICRA)*, pages 2043–2050, May 2017.

[42] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *The International Journal of Robotics Research*, 37(4-5):513–542, 2018.

[43] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In *2012 IEEE International Conference on Robotics and Automation*, pages 957–964, May 2012.

[44] S. Weiss and R. Siegwart. Real-time metric state estimation for modular vision-inertial systems. In *2011 IEEE International Conference on Robotics and Automation*, pages 4531–4537, May 2011.

[45] Stephan Weiss, Markus W. Achtelik, Simon Lynen, Michael C. Achtelik, Laurent Kneip, Margarita Chli, and Roland Siegwart. Monocular vision for long-term micro aerial vehicle state estimation: A compendium. *Journal of Field Robotics*, 30(5):803–831, 2013.

[46] Cheng Zhao, Li Sun, Pulak Purkait, Tom Duckett, and Rustam Stolkin. Learning monocular visual odometry with dense 3d mapping from dense 3d flow. *CoRR*, abs/1803.02286, 2018.