Visual-Inertial Odometry Using Optical Flow from Deep Learning

Jeong Min Kang*, Zoran Sjanic*†, and Gustaf Hendeby*
* Dept. of Electrical Engineering, Linköping University, Linköping, Sweden
e-mail: {jeongmin.kang,zoran.sjanic,gustaf.hendeby}@liu.se
† Saab AB, Linköping, Sweden

***Abstract*—It is shown how dense optical flow obtained using deep learning can be used to provide high quality visual odometry. The obtained odometric information can be utilized as a component to reduce the inherent drift of inertial navigation systems (INS). This could be a key component to provide autonomous system with robust localization capability in GNSS denied environments. The method leverages the power of estimating optical flow from neural networks, which can provide reliable results even when feature based optical flow fails. Comparison of different methods to decide which points of the dense optical flow that should be used to provide as good visual odometry as possible has been performed. Furthermore, it is exemplified how the methodology can help limit the drift in an INS.**

I. Introduction

Accurate and reliable localization plays an essential role in autonomous navigation systems. It is an important enabler for wide variety of applications, for example autonomous driving or unmanned aerial vehicles navigation. Global navigation satellite systems (GNSS), which uses satellites orbiting the earth to calculate the navigation information is, by far, the most used system for providing an accurate location information in outdoor environments. However, GNSS are sensitive to signal disturbances, jamming and spoofing, suffers from degraded performance in urban canyons, and does not work at all in indoor environments where the GNSS signals are blocked. In contrast, inertial navigation system (INS), which uses inertial sensors, is a common way to obtain stand-alone localization information without relying on external signals. However, due to its nature, *i.e.*, dead-reckoning of measured accelerations and angular rates, it suffers from long-term drift which is a problem. Usually this is solved by an additional source of information to reduce the drift and provide a viable long-term navigation solution.

Vision-based localization systems, and especially monocular cameras, have attracted considerable attention over the past few decades due to low sensor costs and wide applicability. Vision-based systems can provide useful complementary sensor information to other sensors such as GNSS, INS, laser scanner, etc. There are many examples of localization systems that are vision-based, *e.g.*, visual odometry (VO) [1], where monocular camera is used for velocity estimation or visual simultaneous localization and mapping (V-SLAM), such as Mono-SLAM, [2], which is an example of a SLAM problem, [3]. Mono-SLAM is essentially an extended Kalman filter (EKF)-based visual navigation system which estimates the camera pose by utilizing information from translations between image frames. It also estimates the 3D-structure by using the 2D features from the images in an EKF. Parallel tracking and mapping (PTAM), [4], is a similar estimator that uses the FAST corner detector as a 2D feature extractor, [5], and also optimizes the map with bundle adjustment. It splits tracking and mapping sequences into two threads and adopts a key-frame idea in the mapping step. However, this system requires manual initialization. In recent years, state-of-the-art solutions, such as ORB-SLAM [6], has emerged as one of the widely used methods in different applications showing robust performance. The visual tracking thread of ORB-SLAM is based on the ORB feature, an oriented FAST and rotated BRIEF, to improve both feature detection and descriptor performance [7], and it overcomes the shortcoming of manual initialization of PTAM by utilizing key-frame-based homography and fundamental matrix estimation with automatic initialization. It proposed a pipeline with three parallel threads consisting of tracking, local mapping, and loop-closing. The direct sparse odometry [8] was proposed as a sliding window bundle adjustment method for the direct tracking method without using a loop closure. The direct methods show reliable tracking performance in large environments. However, it requires a high camera frame rate, and the monocular-based direct methods still have the scale issue. Moreover, due to the limitations of the field of view or information loss in 2D to 3D structure reconstruction from the visual measurements, the motion is only recovered up to an unknown scale factor depending on depth ambiguity, inherent in all monocular systems.

To overcome these disadvantages, visual-inertial navigation, which combines measurements from an inertial sensor and a camera, has been actively studied to solve the scale estimation using the IMU. In addition, it helps with the disadvantages of the inertial system's inherent drift by lowering the error accumulation with support from the visual measurements. The multi-state constraint Kalman filter (MSCKF) [9] is another successful example of robust visual-inertial estimation. It showed good performance in an environment which is sufficiently rich with landmarks that can be tracked by adjusting number of landmarks and, consequently, the size of the covariance (which normally increases quadratically in size with the number of landmarks) which can be an obstacle for real-time applications. MSCKF is utilized in Google's ARCore [10] and Apple's ARKit [11]. Other successful examples of inertial-visual estimation include [12–15]. The last mentioned, VINS-Mono, employs an IMU pre-integration method, [16], which integrates the IMU parameters between camera frames,

This work is sponsored by Swedish Innovation Agency VINNOVA within "Cooperation with Brazil for Innovation Project 2020" grant.

as well as modularized pipeline for the visual-inertial estimator. Expansions of this concept, such as tightly coupled pre-integration and loop closure used in global optimization framework, were done in [17]. It contains bootstrap initialization from arbitrary state and an expanded sensor configuration to *e.g.*, stereo camera and IMU fusion.

The rapid development of the deep learning in computer vision makes it possible to apply these techniques to the navigation problems where vision is utilized. DNN started with replacing obvious components such as feature detectors, descriptor, and depth estimation, [18, 19]. CNN-SLAM [20] was proposed by combining depth estimation from neural networks and the large scale direct monocular SLAM (LSD-SLAM) backend, [21]. It can estimate the camera pose from pure rotation via depth estimation. However, its robustness is limited in long-term navigation. A GoogleNet-based pose estimator, PoseNet, [22], is trained on structure from motion (SfM) data without map information for pose estimation. It requires sophisticated SfM with existing data in the training step, and therefore the performance is highly data dependent. DeepFactors [23], introduces the idea of combining the advantages from previous visual navigation paradigms. It utilizes geometric error from feature-based and deep learning depth estimation methods and photometric error from direct method. It shows clean 3D surface reconstruction, but it still has limitations in large-scale scenes. A study on how combined representations influence the performance [24] suggested that using an intermediate representations method including depth estimation or optical flow performed better than end-to-end learning. Hybrid methods utilizing optical flow networks for visual odometry were proposed [25, 26]. However, these methods still suffer from the inherent scale problem of the camera-only method. Depth networks have been applied to overcome this limitation [27], but it requires more learning parameters as well as more hardware resources.

In this paper, we propose a visual-inertial odometry method using deep learning-based dense optical flow estimation as the means to support the inertial system. We have chosen dense optical flow due to its versatility compared to sparse methods. It gives the possibility to use every pixel and is not limited to the information content in the scene. Deep learning has been chosen due to its capability to estimate dense optical flow in an efficient way, compared to calculating it in a classical way. Our previous study [28] shows that six different dense optical flow estimation methods obtained from deep networks, [29–34], can provide optical flow with good quality. The study compares deep network-based optical flows with classical feature-based flow estimation method. The proposed odometry pipeline combines the deep learning-based dense optical flow (used as visual measurements) with inertial data (accelerations and angular rates) in order to obtain as accurate as possible velocity estimate.

The main contributions of this paper are:

- a fusion method using dense optical flow, obtained from deep neural network, as visual measurements that can reduce drift in IMU based dead-reckoning;

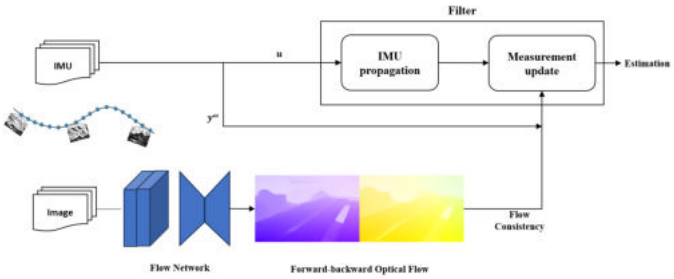


Fig. 1: System overview

- a comparison of different methods to decide the best optical flow from dense flow frames that can improve odometry performance; and
- a performance evaluation of the proposed odometry estimator on a publicly available UAV dataset.

The remainder of this paper is organized the following way. First, in Sec. II, the proposed visual-inertial fusion method is described. The experimental setup and the results from the experiments using the proposed method are presented and discussed in Sec. IV. Finally, the conclusions and discussions are presented in Sec. V.

II. Visual-Inertial Information Fusion

We consider the inertial-visual odometry formulation based on the optical flow from a monocular camera that is rigidly attached to the IMU. It is assumed that the dynamic model for the moving platform and the optical flow measurement model are on the following form:

$$\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) + \boldsymbol{n}_t, \tag{1a}$$

$$0 = h(\boldsymbol{x}_t, \boldsymbol{y}_t, \boldsymbol{e}_t), \tag{1b}$$

where, $\boldsymbol{x}_t$ is the platform's state and $\boldsymbol{y}_t$ is the measurements, both at time t. Here, $\boldsymbol{n}_t$ and $\boldsymbol{e}_t$ denote white Gaussian noise with $\boldsymbol{n}_t \sim \mathcal{N}(0, Q_t)$ and $\boldsymbol{e}_t \sim \mathcal{N}(0, R_t)$, respectively. The dynamic model takes the inertial data as control input $\boldsymbol{u}_t$. This model will be given in more detail below.

A. Notation

We first define the notation that we use throughout the paper. We denote the fixed world frame W, the IMU frame I, the camera frame C, and the platform body frame B. In the camera frames, we denote the pixel frame p, and the normalized image frame n. The points in the pixel frame p are denoted $\boldsymbol{m}_p$, and $\tilde{\boldsymbol{m}}_p$ is the homogeneous version. Therefore, $\tilde{\boldsymbol{m}}_n = K^{-1}\tilde{\boldsymbol{m}}_p$ is defined using the intrinsic camera matrix from a pinhole camera model [35]. Moreover, the skew-symmetric matrix operator is defined as

$$\Omega(\boldsymbol{u}) = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}, \tag{2}$$

acting on a 3D vector $\boldsymbol{u} = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^T$.

B. IMU dynamic model

The inertial measurements of the angular velocity $\boldsymbol{\omega}_t^I$ and acceleration $\boldsymbol{a}_t^I$ from the IMU are given by

$$\hat{\boldsymbol{\omega}}_t^I = \boldsymbol{\omega}_t^I + \boldsymbol{n}_t^\omega, \tag{3a}$$

$$\hat{\boldsymbol{a}}_t^I = \boldsymbol{a}_t^I + \boldsymbol{n}_t^a, \tag{3b}$$

where, $\hat{\boldsymbol{\omega}}_t^I$ and $\hat{\boldsymbol{a}}_t^I$ are the IMU measurements at time t. $\boldsymbol{n}_t^\omega$ and $\boldsymbol{n}_t^a$ denote angular velocity and acceleration noise, respectively, both assumed to be Gaussian white noise with covariance $\Sigma_\omega = \sigma_\omega^2 \boldsymbol{I}_3$ and $\Sigma_a = \sigma_a^2 \boldsymbol{I}_3$, respectively. Here, $\boldsymbol{I}_3$ denote an identity matrix of dimension 3×3.

The state $\boldsymbol{x}$ of the dynamic model comprises the position $\boldsymbol{p}^W = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$, the velocity $\boldsymbol{v}^W = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T$, and a unit quaternion $\boldsymbol{q}^{WB} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T$. The quaternion $\boldsymbol{q}^{WB}$ defines the orientation of the body frame B expressed in the world frame W [36]. We can now use the measurements from IMU to infer the system dynamics model in time interval T as

$$\boldsymbol{p}_t^W = \boldsymbol{p}_{t-1}^W + T\boldsymbol{v}_{t-1}^W + \frac{T^2}{2}\left(R_{t-1}^{WB}(\boldsymbol{a}_t^B + \boldsymbol{n}_t^a) + \boldsymbol{g}^W\right) \tag{4a}$$

$$\boldsymbol{v}_t^W = \boldsymbol{v}_{t-1}^W + T\left(R_{t-1}^{WB}(\boldsymbol{a}_t^B + \boldsymbol{n}_t^a) + \boldsymbol{g}^W\right) \tag{4b}$$

$$\boldsymbol{q}_t^{WB} = \mathbf{exp}\big(\tfrac{T}{2}\mathcal{S}(\boldsymbol{\omega}_t^B + \boldsymbol{n}_t^\omega)\big)\boldsymbol{q}_{t-1}^{WB}, \tag{4c}$$

where the skew-symmetric 4×4 matrix

$$\mathcal{S}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}, \tag{5}$$

and R_{t-1}^{WB} is the shorthand notation for the rotation matrix parametrized using the unit quaternion as in $R(\boldsymbol{q}_{t-1}^{WB})$ and $\boldsymbol{g}^W = \begin{bmatrix} 0 & 0 & -9.81 \end{bmatrix}^T$ denotes the gravity vector. Note also that acceleration and angular velocity are expressed in the body frame (B). This is done by applying the rotation matrix between inertial and body frame, R^{BI} (assumed to be known and constant), to accelerations and angular velocities, $\{\boldsymbol{a}, \boldsymbol{\omega}\}^B = R^{BI}\{\boldsymbol{a}, \boldsymbol{\omega}\}^I$. Another frame that is important is the camera frame (C), and in this work we will assume that the camera and body frame are the same.

C. Optical Flow

Optical flow is defined as the movement of a pixel between consecutive image frames. The basic assumption is that the pixel should represent the same point in 3D scenes, then it is possible to describe properties of the motion of the camera without knowledge of the underlying 3D world. Assume a set of correspondences between images I^k and I^{k+1}, $\boldsymbol{m}_p^k$ and $\boldsymbol{m}_p^{k+1}$, the optical flow for the point $\boldsymbol{m}_p^k$ is then defined as

$$\dot{\boldsymbol{m}}_p^k = \boldsymbol{m}_p^{k+1} - \boldsymbol{m}_p^k. \tag{6}$$

Assuming the underlying point in 3D giving rise to the point in the image is the same, the optical flow contain information about the movement of the camera.

Traditionally optical flow has been computed using feature points which could be tracked in consecutive images, *i.e.*, I^k, from which the optical flow is easily computable.

D. Visual Measurement Update Model

We use the derivations in [37, 38] which considers the image point denoted $\boldsymbol{m}_n$ and the optical flow in that point denoted $\dot{\boldsymbol{m}}_n$. It is assumed that the measurement of the flow is calculated from the neural networks. Following the derivations in the references above, optical flow measurements can be related to the kinematics of the camera motion and combined with the IMU dynamics model for estimation purposes in the following way: from the 3D point observation model (which we assume is time dependent), $\boldsymbol{m}_C = R^{CW}\boldsymbol{m}_W + \boldsymbol{t}^C$, and taking its time derivative, we can derive the following equation:

$$\begin{aligned} \dot{\boldsymbol{m}}_C &= \dot{R}^{CW}\boldsymbol{m}_W + R^{CW}\dot{\boldsymbol{m}}_W + \dot{\boldsymbol{t}}^C \\ &= \Omega(\boldsymbol{\omega}^C)(\boldsymbol{m}_C - \boldsymbol{t}^C) + \dot{\boldsymbol{t}}^C, \end{aligned} \tag{7}$$

where $\dot{\boldsymbol{m}}_C$ is (3D) optical flow in camera frame, R^{CW}, $\boldsymbol{t}^C$ are the extrinsic parameters of the camera observation model, and $\Omega(\boldsymbol{\omega}^C)$ is the skew-symmetric matrix obtained from the angular velocity $\boldsymbol{\omega}^C$.

Since the depth $\lambda = \boldsymbol{m}_{C,z}$ is unknown, it is convenient to eliminate it from (7) by using following relationship; $\boldsymbol{m}_C = \lambda\tilde{\boldsymbol{m}}_n$, where $\tilde{\boldsymbol{m}}_n$ is the homogeneous form of the image point. Equation (7) can now be rewritten as follows:

$$\dot{\tilde{\boldsymbol{m}}}_n = \boldsymbol{\omega}^C \times \tilde{\boldsymbol{m}}_n + \frac{1}{\lambda}\boldsymbol{v}^C - \frac{\dot{\lambda}}{\lambda}\tilde{\boldsymbol{m}}_n, \tag{8}$$

where $\dot{\tilde{\boldsymbol{m}}}_n$ is the homogeneous form of optical flow in which the z-axis component is zero per definition, and $\boldsymbol{v}^C$ is related to the states as

$$\boldsymbol{v}^C = \dot{\boldsymbol{t}}^C - \Omega(\boldsymbol{\omega}^C)\boldsymbol{t}^C \tag{9a}$$

$$\boldsymbol{t}^C = -R^{CW}\boldsymbol{p}^W \tag{9b}$$

$$\dot{\boldsymbol{t}}^C = -\Omega(\boldsymbol{\omega}^C)R^{CW}\boldsymbol{p}^W - R^{CW}\boldsymbol{v}^W \tag{9c}$$

Scalar-multiplying both sides of (8) by $(\boldsymbol{v}^C \times \tilde{\boldsymbol{m}}_n)^T$ from the right, the depth parameter λ can be eliminated resulting in

$$0 = \dot{\tilde{\boldsymbol{m}}}_n^T(\boldsymbol{v}^C \times \tilde{\boldsymbol{m}}_n) + \tilde{\boldsymbol{m}}_n^T(\boldsymbol{\omega}^C \times (\boldsymbol{v}^C \times \tilde{\boldsymbol{m}}_n)). \tag{10}$$

We link the parameters with the system model to obtain the final measurement equation. We assume that the camera and IMU are mounted on a rigid body, and that the initial alignments are given. The angular velocity $\boldsymbol{\omega}^C$ is from the world frame to camera frame, which can be obtained from the IMU measurement with rotation matrix R^{CI}, the IMU-camera rotation information. Using the angular velocity $\boldsymbol{\omega}^C$, image coordinates $\tilde{\boldsymbol{m}}_n$ and their optical flow $\dot{\tilde{\boldsymbol{m}}}_n$, as well as mutually independent measurement noise at time t, the final measurement equation is obtained by

$$\begin{aligned} 0 &= h(\boldsymbol{x}_t, \boldsymbol{y}_t, \boldsymbol{e}_t) \\ &= (\dot{\tilde{\boldsymbol{m}}}_n + \boldsymbol{e}^{of})^T(\boldsymbol{v}^C \times \tilde{\boldsymbol{m}}_n) \\ &\quad + \tilde{\boldsymbol{m}}_n^T((\boldsymbol{\omega}^C + \boldsymbol{e}^\omega) \times (\boldsymbol{v}^C \times \tilde{\boldsymbol{m}}_n)) \\ &= \dot{\tilde{\boldsymbol{m}}}_n^T(\boldsymbol{v}^C \times \tilde{\boldsymbol{m}}_n) + \tilde{\boldsymbol{m}}_n^T(\boldsymbol{\omega}^C \times (\boldsymbol{v}^C \times \tilde{\boldsymbol{m}}_n)) \\ &\quad + \tilde{\boldsymbol{m}}_n^T(\boldsymbol{e}^\omega \times (\boldsymbol{v}^C \times \tilde{\boldsymbol{m}}_n)) + (\boldsymbol{e}^{of})^T(\boldsymbol{v}^C \times \tilde{\boldsymbol{m}}_n), \end{aligned} \tag{11}$$

where $\boldsymbol{e}^{of}$ and $\boldsymbol{e}^{\omega}$ are the mutually independent measurement noise of optical flow and angular velocity and $\boldsymbol{y}_t = \begin{bmatrix}\boldsymbol{\omega}^C & \dot{\boldsymbol{m}}_n\end{bmatrix}^T$.

The above discussion shows how the optical flow in a given image point can be formulated as a measurement. The information in a single point of optical flow is limited, using more points may provide more information. This is achieved by simply stacking several measurements equations (11) for different points, as described above. This composite measurement can then be used in an EKF to infer the state.

Since optical flow and the angular velocity are used as the measurements, the proposed method can update the velocity and orientation of the state more directly. This means that the position estimate will still inherit drift from the velocity estimation error. However, for completeness sake, we evaluate performance comprehensively for all the states in the next section.

III. Obtaining Optical Flow Measurements

Recent advances in deep learning methods have resulted in new ways to compute optical flow, and DNN can be used to compute dense optical flow from image pairs (I^k, I^{k+1}). The selection of in which points of the optical flow to used has a significant impact on how much information can be extracted. We will first outline how DNN can be used to obtain dense optical flow, followed by a discussion about different ways to select which of estimated points to use.

A. Optical Flow from Deep Learning

Compared to feature based, sparse, optical flow, where the optical flow is recovered in only a few points, DNN can provide dense optical flow, *i.e.*, optical flow for each pixel in an image. As such, it can provide optical flow in cases when the feature detector used in a sparse method fails to find matching feature points.

Most DNN for optical flow estimation are trained using synthetic data. The reason is that it is very difficult to accurately label image pairs with respect to optical flow, whereas it is easy to generate high quality optical flow between two simulated images. The paper [28] presents an evaluation of six different methods to compute dense optical flow using DNN, trained using synthetic data, and benchmarked on real images. Based on these results, which show no significant difference between the evaluated methods, the *global motion aggregation* (GMA) [34] method trained on synthetic data will be used to compute dense optical flow in this paper.

It should be noted that, even though DNN methods provide dense optical flow, the quality of the computed optical flow typically depends on the available texture in the images from which to infer the optical flow. This leads up to the question of how to select which points of the optical flow should be used in the visual-inertial information fusion.

B. Feature-Based Optical Flow Selection

When using sparse, feature-based, optical flow, optical flow points to use are naturally determined by the detected feature points. These in turn depend on the feature detector used, *e.g.*, ORB as in this paper. The number of points can be controlled by setting the threshold for feature matching. A benefit of using features is that the quality of the optical flow is high (in feature points), as they with good accuracy connects detected features in consecutive images. The downside is that, depending on the structure in the image, the number of detected features might be low, resulting in a low diversity amongst the points in which the optical flow is available.

C. Grid-Based Optical Flow Selection

When using dense optical flow, potentially all image pixels could be used for estimation. However, this strategy is not advisable due to shear amount of data to process in the filter leading to quick increase in computational complexity, as well as flow quality. This quality varies in the image and including poor optical flow in the estimation should be avoided. The measurements of optical flow are most informative if the optical flow points in different directions. One method to select suitable optical flow measurements is to divide the image into a grid, and then pick one or more measurements from each cell, *e.g.*, the center point or the point with the strongest optical flow. The idea being that, while being an inexpensive selection criterion, by choosing points from different cells the optical flow diversity should be maximized. Picking the strongest optical flow should maximize the signal to noise ratio which, in turn, increases the quality of the measurement.

D. Consistency-Based Optical Flow Selection

Another method to select optical flow points is to try to identify high quality optical flow points across the whole image and use them as measurements. The idea pursued in [39] is that points for which the optical flow between the images I^k and I^{k+1} and back, from I^{k+1} to I^k, is consistent should have high quality flow. This is because the GMA finds the same optical flow in both directions. These points can then be used to construct the measurement.

To define a consistency measure, consider the point $\boldsymbol{m}_p^k$ in image I^k with the associated optical flow $\dot{\boldsymbol{m}}_p^k$, for which the matching pixel in image I^{k+1} is

$$\hat{\boldsymbol{m}}_p^{k+1} = \boldsymbol{m}_p^k + \dot{\boldsymbol{m}}_p^k.$$

Now, using the reverse optical flow from image I^{k+1} to I^k,

$$\hat{\boldsymbol{m}}_p^k = \hat{\boldsymbol{m}}_p^{k+1} + \dot{\boldsymbol{m}}_p^{k+1,b},$$

where superscript b is used to indicate the backward direction of the optical flow. The optical flow in $\boldsymbol{m}_p^k$ is fully consistent if $\hat{\boldsymbol{m}}_p^k = \boldsymbol{m}_p^k$. That is, filter out pixels where

$$\boldsymbol{\mathcal{D}}^k(\boldsymbol{m}_p^k) = \left\|\hat{\boldsymbol{m}}_p^k - \boldsymbol{m}_p^k\right\| = \left\|\dot{\boldsymbol{m}}_p^k + \dot{\boldsymbol{m}}_p^{k+1,b}\right\| \tag{12}$$

is small. This concept is illustrated in Fig. 2.

We select optical flows with $\boldsymbol{\mathcal{D}}^k(\boldsymbol{m}_p^k)$ lower than a certain threshold, i.e., high flow consistency, and use them in the visual measurement model. That is, pixels with high confidence based on the flow consistency are used as measurements. This procedure can be combined with other methods if the number of measurements needs to be limited further.

TABLE I: Summary of estimation results. Boldface figures denote the lowest value for each parameter and data set.

		Velocity Error [m/s]				Orientation Error [°]				Position Error [m]			
		x	y	z	**Total**	ψ	θ	ϕ	**Total**	x	y	z	**Total**
	DR	3.02	0.73	0.48	3.14	**0.05**	0.41	**0.05**	0.49	83.25	36.40	16.80	92.40
mh_01_easy	ORB Selection	0.34	0.38	**0.10**	0.52	**0.05**	0.35	0.07	0.38	4.58	7.84	1.80	9.26
(3 682 frames)	Grid Selection	0.25	0.32	0.17	0.44	**0.05**	**0.17**	**0.05**	0.24	2.25	3.62	**1.48**	4.51
	Consistency Selection	**0.18**	**0.23**	0.13	**0.32**	**0.05**	0.19	**0.05**	**0.22**	**1.57**	**1.92**	2.21	**3.32**
	DR	1.38	1.35	0.18	1.94	0.04	**0.12**	**0.06**	**0.15**	58.55	64.93	6.54	87.68
mh_03_medium	ORB Selection	0.81	0.60	0.08	1.01	0.66	0.44	1.12	1.24	7.83	**4.06**	**2.55**	**9.18**
(2 700 frames)	Grid Selection	0.54	**0.33**	0.08	0.64	0.11	0.23	0.17	0.28	6.11	8.51	4.98	11.61
	Consistency Selection	**0.30**	**0.33**	**0.06**	**0.45**	**0.03**	0.18	**0.06**	0.16	**3.43**	13.45	4.21	14.50
	DR	1.36	1.22	0.38	1.87	0.17	0.59	**0.22**	0.85	21.24	24.96	19.38	38.08
v2_01_easy	ORB Selection	0.33	0.41	0.18	0.56	**0.16**	0.48	**0.22**	0.75	**2.53**	3.87	4.13	6.20
(2 280 frames)	Grid Selection	0.21	0.34	0.16	0.43	**0.16**	**0.46**	**0.22**	0.71	2.98	**2.76**	3.17	5.16
	Consistency Selection	**0.20**	**0.32**	**0.15**	**0.40**	**0.16**	**0.46**	**0.22**	**0.70**	2.60	3.24	**2.54**	**4.87**
	DR	7.87	1.58	0.15	8.03	0.26	0.66	0.37	0.91	277.77	22.63	4.11	278.72
v2_03_difficult	ORB Selection	0.37	0.39	**0.09**	0.54	0.25	**0.57**	0.35	**0.73**	8.49	2.85	1.82	9.14
(1 922 frames)	Grid Selection	0.38	**0.36**	**0.09**	0.53	**0.24**	0.60	**0.33**	0.75	7.61	**1.27**	**0.87**	7.76
	Consistency Selection	**0.30**	0.41	**0.09**	**0.52**	0.33	0.60	0.42	0.81	**4.06**	2.22	1.83	**4.98**

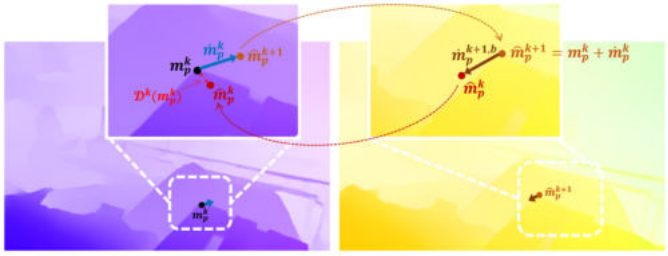


Fig. 2: An example of the forward-backward optical flow relationship between two frames. $\dot{\boldsymbol{m}}_p^k$ is the forward optical flow of frame I^k to frame I^{k+1}, and $\dot{\boldsymbol{m}}_p^{k+1,b}$ is the matching backward optical flow.

IV. Experimental Setup and Evaluations

In this section, we describe the details of the experimental evaluation of our method using the publicly available EuRoc datasets [40]. The estimation performance of the sparse feature-based optical flow and optical flow without flow consistency, described in Section III-A, is compared to the proposed method.

A. Implementation Details

The optical flow network of the proposed framework is implemented using PyTorch [41] with the mixed precision strategy. The network is initialized from scratch with random weights using Tesla T4 GPU. We follow the standard optical flow training conventions [30, 33, 42]. We first pre-train GMA network on FlyingChairs [29], and fine-tune it on a combination of FlyingThings3D [43] and Sintel [42]. We train on FlyingThings3D for 120 000 iterations with a batch size of 8, followed by another 120 000 iterations on FlyingThings with batch size of 6. Finally, the model is further fine-tuned on Sintel for 120 000 iterations with batch size 6. We implement the same hyperparameters as GMA for the training. The runtime of the measurement step is significantly affected by the inference time of this trained GMA networks.

B. Experimental setup

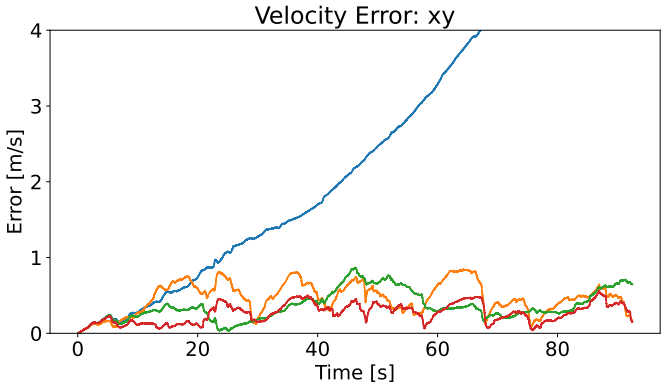
We compare our approach with pure inertial dead-reckoning (DR), i.e., no support from the optical measurements, classical sparse feature-based optical flow instead of deep networks, and optical flow without considering flow consistency. For the sparse optical flow, ORB feature detector [7] is used. To compare the influence of flow consistency on point selection in dense optical flow, we assign 16 equally distributed small patches in the dense flow frame and extract optical flow from each patch under the norm constraint that satisfies some threshold to remove outliers. We then compose the optical flow measurements as the maximum flow value in the patch. The threshold for consistency-based optical flow is selected empirically, and we set it to 8×10^{-4}.

All the methods are evaluated on image sequences from the EuRoC dataset [40], in particular the "Machine Hall 01" (here denoted "mh_01_easy"), "Machine Hall 03" ("mh_03_medium"), "Vicon Room 02" ("v2_01_easy"), and "Vicon Room 03" ("v2_03_difficult"), which are the same sequences used in our previous work [28]. All comparisons are performed against the ground truth available in the dataset.

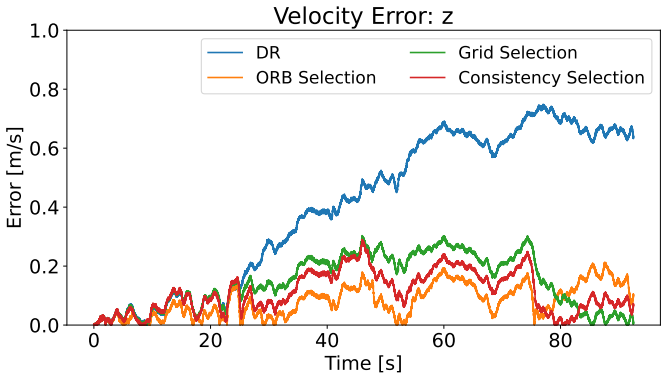
C. Results

The quantitative results of the proposed method on all sequences are shown in Table I. Velocity error, angle error, and position error measure the root-mean-square error from their respective ground truth. Below, the velocity and orientation estimates will be evaluated first, then the resulting position estimates.

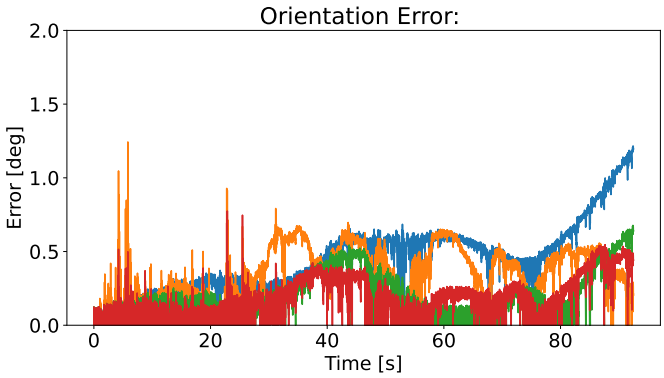
1) Velocity and Orientation: The velocity and position errors are expressed as each axis and total errors. The orientation is considered as Euler angles; roll, pitch, and yaw. The qualitative results are shown in Fig. 3–4, containing velocity error in xy-plane, z-axis velocity error, orientation error expressed as angle differences and the position error for each axis for mh_01_easy and v2_03_difficult sequences. The table and figures indicate that the proposed method using flow with the consistency check shows better average



(a) xy velocity error

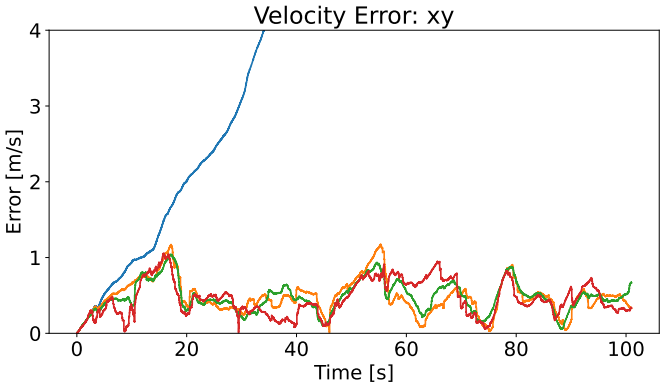


(b) z velocity error

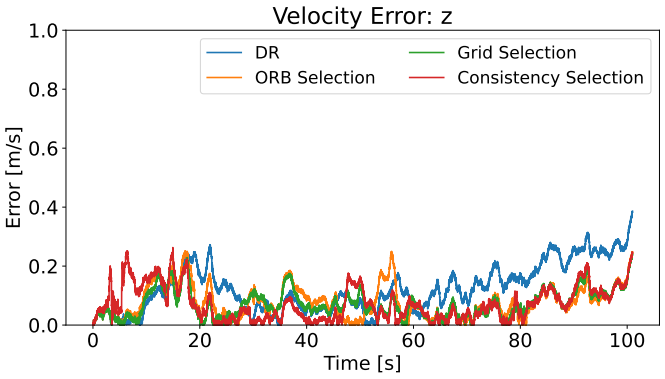


(c) Orientation error

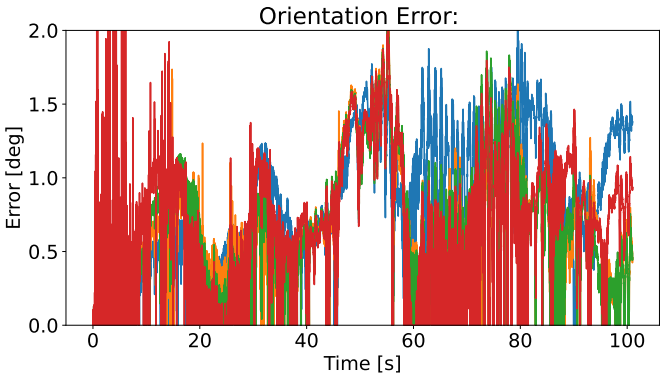
Fig. 3: Estimation errors using the different methods on the mh_01_easy sequence.



(a) xy velocity error



(b) z velocity error



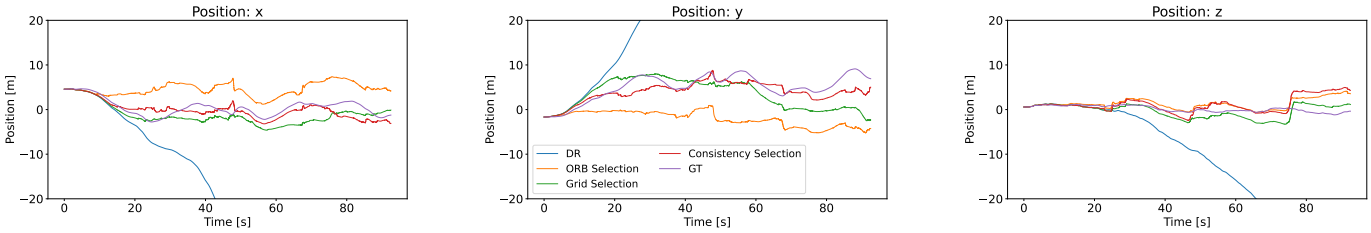
(c) Orientation error

Fig. 4: Estimation errors using the different methods on the v2_03_difficult sequence.

estimation performance than pure dead-reckoning, the feature-based method, and the grid selection method.

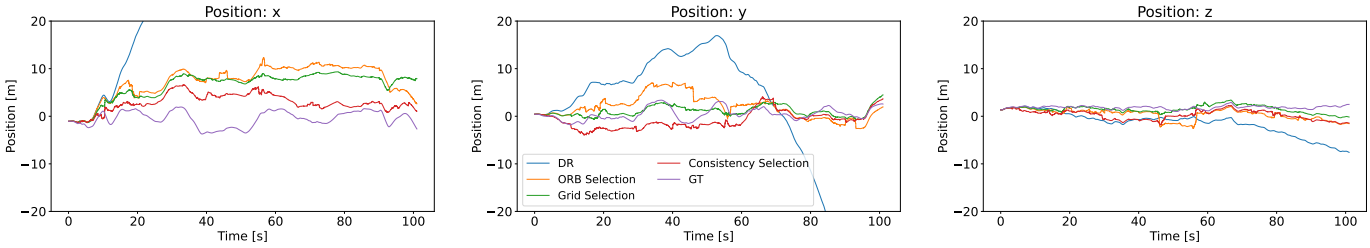
The velocity estimation from DR data shows large errors compared to other methods and, in some sequences, it diverges. This is more or less the expected behavior of the dead-reckoning with uncorrected inertial measurement. However, estimators with added visual measurement do not diverge and show less velocity errors than DR. As can be seen in the quantitative analysis, although the velocity error between methods varies depending on axis, the proposed method with flow consistency shows improved total velocity estimation error in all cases compared to other methods. On the other hand, all compared methods including DR show small orientation errors in all sequences. It means that the proposed dynamic model provides more accurate orientation estimation although drift occurs in the velocity estimate.

One interesting result is that the z-axis velocity estimation shows relatively small errors compared to other axes (this includes DR as well). The probable reason for this is the camera configuration on the UAV. The camera is mounted in



(a) x position error (b) y position error (c) z position error

Fig. 5: Position estimates for mh_01_easy sequence.



(a) x position error (b) y position error (c) z position error

Fig. 6: Position estimates for v2_03_difficult sequence.

such a way that it faces in the direction of flight most of the time. This means that z-axis is basically parallel with the image plane and most of the xy-plane is perpendicular to it. In turn, the flow induced by the movement in the xy-plane will in general be small in most of the pictures, while the flow in the z-direction is easier to detect. These two things together will give an estimation result similar to what was obtained here.

2) Position: The position estimates from DR diverges with large drifts in all cases, as illustrated in Fig. 5–6. However, the results show the effect of using measurement update from the optical flow extracted with both feature-based and deep learning methods. Additionally, in mh_03_medium, the feature based optical flow method shows small average position error, but the results of other sequences show that the proposed method (with flow consistency) is efficient in reducing position estimation drift over long sequences. Overall, using optical flow as a measurement is efficient in reducing velocity and, mostly, position drift and deep learning-based optical flow is useful in providing dense flow in a relatively straightforward way. The dense flow is more preferable to sparse one, since the possibility to choose the best parts of the image is much easier in that case. In addition, the results also show that using the flow consistency prediction the estimation is further improved.

V. Conclusions and Future Work

In our previous study [28], deep learning-based dense optical flow was evaluated and compared to the sparse feature-based flow, and the performance was quite satisfactory. We proposed that deep learning-based dense flow can be used for support of the navigation system by fusion with inertial information. To extract optical flow from dense frames, a grid-based method and a flow consistency method was proposed and evaluated. The results show that deep learning-based optical flow provides efficient measurement information with relatively superior visual inertial navigation compared to traditional feature-based methods, and in particular, can update stable measurements by removing outliers through flow consistency.

The proposed method is shown to be effective in reducing the velocity drift. However, the position error is still relatively large, and it can be considered to further reducing it by adding measurements that can update position information in addition to optical flow. Future work includes expanding experiments on other environments and compare with reference to other approaches.

References

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

[2] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, 2007.

[3] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, 2006.

[4] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 225–234.

[5] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, 2005, pp. 1508–1515.

[6] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, 2015.

[7] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571.

[8] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, 2017.

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

[10] G.-D. Voinea, F. Girbacia, C. C. Postelnicu, and A. Marto, "Exploring cultural heritage using augmented reality through Google's project Tango and ARCore," in *VR Technologies in Cultural Heritage: First International Conference, VRTCH 2018, Brasov, Romania, May 29–30, 2018, Revised Selected Papers 1*. Springer, 2019, pp. 93–106.

[11] U. Dilek and M. Erol, "Detecting position using ARKit II: generating position-time graphs in real-time and further information on limitations of arkit," *Physics Education*, vol. 53, no. 3, p. 035020, 2018.

[12] 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)*, 2015, pp. 298–304.

[13] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[14] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart, "maplab: An open framework for research in visual-inertial mapping and localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1418–1425, 2018.

[15] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, 2018.

[16] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, 2016.

[17] T. Qin, S. Cao, J. Pan, and S. Shen, "A general optimization-based framework for global pose estimation with multiple sensors," *arXiv preprint arXiv:1901.03642*, 2019.

[18] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: Learned invariant feature transform," in *Proceeding of the 14th European Conference on Computer Vision–ECCV*, Amsterdam, The Netherlands, Oct. 2016, pp. 467–483.

[19] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[20] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6243–6252.

[21] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13*. Springer, 2014, pp. 834–849.

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

[23] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "Deepfactors: Real-time probabilistic dense monocular slam," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 721–728, 2020.

[24] B. Zhou, P. Krähenbühl, and V. Koltun, "Does computer vision matter for action?" *Science Robotics*, vol. 4, no. 30, p. eaaw6661, 2019.

[25] P. Muller and A. Savakis, "Flowdometry: An optical flow and deep learning based approach to visual odometry," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017, pp. 624–631.

[26] T. Pandey, D. Pena, J. Byrne, and D. Moloney, "Leveraging deep learning for visual odometry using optical flow," *Sensors*, vol. 21, no. 4, p. 1313, 2021.

[27] X. Ban, H. Wang, T. Chen, Y. Wang, and Y. Xiao, "Monocular visual odometry based on depth and optical flow using deep learning," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–19, 2020.

[28] J. M. Kang, Z. Sjanic, and G. Hendeby, "Optical flow revisited: how good is dense deep learning based optical flow?" in *2023 IEEE Symposium Sensor Data Fusion and International Conference on Multisensor Fusion and Integration (SDF-MFI)*. IEEE, 2023, pp. 1–6.

[29] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2758–2766.

[30] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2462–2470.

[31] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-net: CNNs for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8934–8943.

[32] T.-W. Hui, X. Tang, and C. C. Loy, "LiteFlowNet: A lightweight convolutional neural network for optical flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8981–8989.

[33] Z. Teed and J. Deng, "RAFT: Recurrent all-pairs field transforms for optical flow," in *Proceedings of the 16th European Conference on Computer Vision–ECCV*, Glasgow, UK, Aug. 2020, pp. 402–419.

[34] S. Jiang, D. Campbell, Y. Lu, H. Li, and R. Hartley, "Learning to estimate hidden motions with global motion aggregation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9772–9781.

[35] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge university press, 2003.

[36] Z. Sjanic, M. A. Skoglund, T. B. Schön, and F. Gustafsson, "A nonlinear least-squares approach to the SLAM problem," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 4759–4764, 2011.

[37] G. Bleser and G. Hendeby, "Using optical flow as lightweight SLAM alternative," in *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, 2009, pp. 175–176.

[38] G. Bleser and G. Hendeby, "Using optical flow for filling the gaps in visual-inertial tracking," in *2010 18th European Signal Processing Conference*, 2010, pp. 1836–1840.

[39] H. Zhan, C. S. Weerasekera, J.-W. Bian, and I. Reid, "Visual odometry revisited: What should be learnt?" in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 4203–4210.

[40] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[41] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS 2017 Workshop on Autodiff*, 2017.

[42] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI 12*, 2012, pp. 611–625.

[43] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4040–4048.