Efficient Visual Ego-Motion Estimation for Agile Flying Robots

Xu, Y.

DOI
10.4233/uuid:a5998475-a7be-46e5-9bd9-6fbd9b81c15c

Publication date
2023

Document Version
Final published version

Citation (APA)
Xu, Y. (2023). *Efficient Visual Ego-Motion Estimation for Agile Flying Robots*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:a5998475-a7be-46e5-9bd9-6fbd9b81c15c

Efficient Visual Ego-Motion Estimation for Agile Flying Robots

EFFICIENT VISUAL EGO-MOTION ESTIMATION FOR AGILE FLYING ROBOTS

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, Prof.dr.ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Thursday 7 September 2023 at 15:00 o'clock

by

Yingfu XU

Master of Science in Aeronautical and Astronautical Science and Technology,
Harbin Institute of Technology, China
born in Tieling, China

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof.dr. G.C.H.E. de Croon	Delft University of Technology, promotor
Dr.ir. C. De Wagter	Delft University of Technology, copromotor
Independent members:	
Dr. M. Sifalakis	Stichting IMEC Nederland
Dr. J. Martinez-Carranza	INAOE, Mexico
Prof.dr. D.M. Gavrila	Delft University of Technology
Prof.dr. E.O. Postma	Tilburg University
Prof.dr.ir. M. Mulder	Delft University of Technology

Keywords: Micro Air Vehicles, Ego-Motion Estimation, Deep Neural Networks, Self-Supervised Learning, Network Prediction Uncertainty, Monocular Visual-Inertial Odometry, Monocular Depth Prediction

Printing: Ridderprint | www.ridderprint.nl

Front & Back: Ir. Cai Huang

ISBN 978-94-6384-477-2

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

CONTENTS

SUMMARY

Micro air vehicles (MAVs) have shown significant potential in modern society. The development in robotics and automation is changing the roles of MAVs from remotely controlled machines requiring human pilots to autonomous and intelligent robots. There is an increasing number of autonomous MAVs involved in outdoor operations. In contrast, the deployment of MAVs in GPS-denied environments is relatively less practiced. The speed when flying indoors is often slow. One reason is that MAVs are surrounded by obstacles. But it should also be noticed that ego-motion estimation becomes more difficult to remain reliable during faster flight.

The reason for this is that fast motion brings challenges to the robustness and computational efficiency of ego-motion estimation solutions based on the limited onboard sensing and processing capacities. The challenge to robustness is that the motion blur induced by agile maneuvers reduces the amount of available visual information needed by the current mainstream ego-motion estimation solutions, given the fact that frame-based cameras are the primary sensor for most lightweight MAVs. The challenge of computational efficiency comes from the strong desire for smaller and smaller MAVs to better fit cluttered environments. Moreover, to compensate for the decrease in robustness, additional computational power is required to detect known landmarks or visual processing that better copes with motion blur. This dissertation responds to the challenges by investigating novel ego-motion estimation approaches that combine robustness and efficiency.

First, the goal of higher efficiency in the context of traditional visual feature points is pursued, albeit at the cost of reduced accuracy. The targeted scenarios are where known landmarks exist, such as gates in autonomous drone racing. The proposed velocity estimator's mission is to navigate the MAV until the next landmark appears in the field of view and corrects the accumulated drift in the position estimation. To prevent drift over time, a simple linear drag force model is used for estimating the pitch and roll angles of the MAV with respect to the gravity vector and its velocity within the horizontal plane of the propellers. The translational motion direction and the relative yaw angle are efficiently calculated from the correspondences of feature points using a RANSAC-based linear algorithm.

Secondly, the focus of this dissertation shifts to the robustness against motion blur. Specifically, artificial neural networks (ANNs) are chosen as the vision front-end. Unlike many prior works that train and test ANNs solely in a known environment, the present study demands the ANNs be able to generalize to unknown environments and perform self-supervised learning without relying on ground-truth labels. The goal is for ANNs to achieve a general deployment capability comparable or close to that of traditional vision front-ends. A monocular downward-facing camera is selected as the vision sensor, given that a planar homography transformation can be utilized when the ground is mainly planar to simplify the learning process and speed up onboard processing. Regarding motion blur, the downward-facing camera is more affected than the forward-facing one, espe-

cially in fast flight close to the ground, thus it is suitable for studying the blur robustness. MAV flight experiments show that the ANN-based method is more robust to blurry images than their classical counterparts using feature points. Furthermore, uncertainty estimation for network prediction is studied. The estimated uncertainty is utilized in a visual-inertial odometry (VIO) solution based on an extended Kalman filter (EKF). It contributes to high accuracy when tested on a high-speed dataset, comparable to state-of-the-art VIOs.

Ultimately, we remove the requirement for the camera to face a mainly planar surface and apply the gained experience to the forward-facing camera that films general 3-d structures. To conduct self-supervised learning, we adopt the mainstream approach of jointly learning the camera pose and dense depth map and enhance the accuracy using state-of-the-art techniques. The resulting computationally heavy networks achieve high prediction accuracy and act as the learning target in the training of lightweight student networks. The student pose network is capable of uncertainty estimation and serves as the vision front-end of an efficient EKF-based VIO system. The student depth network predicts downsampled depth maps that are field tested in obstacle avoidance of a nano quadrotor MAV.

Overall, this dissertation studies visual ego-motion estimation solutions for lightweight MAVs. Robustness towards motion blur and computational efficiency of the solutions have high priorities in the algorithmic designs. The main contribution is the development of generalization-capable learning-based approaches that cope with motion blur better than traditional methods and, at the same time, guarantee real-time performance when using an on-the-shelf mobile processor. There are additional contributions that lie in increasing efficiency when using traditional visual processing and an aerodynamic model, insight into network uncertainty estimation, and a training scheme of a lightweight monocular depth network for obstacle avoidance.

1

INTRODUCTION

Robots are gradually coming from science fiction novels and movies into people's daily lives. Over the past two decades, flying robots, also known as drones and micro air vehicles (MAVs), have made even greater progress than their robot peers that walk or swim. The most well-known application of MAVs is aerial photography. Carrying a camera, an MAV allows human beings as terrestrial lives to observe our world from the bird's view in the most affordable and accessible way. MAVs also succeed in other outdoor applications such as agricultural plant protection, geographic mapping, relay communications, delivery, etc. Satellite navigation technologies represented by the Global Positioning System (GPS) [1] provide reliable ego-motion information to navigate the MAV to accomplish its mission safely and effectively.

However, GPS signals are unavailable indoors and become less reliable near large buildings or in a forest. Similar to GPS, external infrastructures, such as motion capture systems (MCSs) [2] and wireless beacons [3], can measure the position of optical markers and radio markers, respectively. These infrastructures can be established indoors and work well after calibration. MCSs are known for their high measurement accuracy and are widely used by research laboratories. But it is difficult to promote them for real-world applications because of their high cost. More importantly, it restricts the flight range of MAVs within the effective range of the external sensors, rendering them useless for scenarios involving unknown environments.

The most common onboard sensors for robot navigation are frame-based cameras and 3-dimensional (3-d) light detection and ranging (LiDAR) sensors [4, 5]. A LiDAR sensor measures the bearing of the environmental point clouds and the metric-scale distance to them. LiDAR measurements are largely invariant to illumination change. However, the big size and weight of LiDAR sensors make them a heavy burden for MAVs. A stereo camera can provide the metric-scale depth of pixels via a stereo matching algorithm. However, given the shape of a small-size quadrotor MAV, mounting a monocular camera is much easier than mounting a stereo camera with a proper baseline. Monocular cameras have been shown to be suitable even for very lightweight (< 50 grams) MAVs [6, 7] thanks to their small size, weight, and energy consumption.

An MAV can estimate its translational velocity by combining the measurements of a monocular camera and a range finder sensor that are downward-facing mounted [8–10]. For higher accuracy in ego-motion estimation and acquiring surrounding information, visual simultaneous localization and mapping (V-SLAM) have been applied to MAVs. A V-SLAM system simultaneously estimates the camera's position and orientation and constructs a map of the surrounding. The map can be made of sparse visual feature points (feature-based method) [11] or be dense, containing a higher percentage of image information by processing raw pixel intensities (direct method) [12]. V-SLAM requires considerable computing power. For MAV platforms without onboard processors qualified for that, a ground station is required to process the sensor data transmitted from the MAV and send back the output of the V-SLAM system [13, 14]. Compared to onboard processing, using an external processor is susceptible to interference. Communication range and delay can be bottlenecks in the autonomous MAV system.

The map of a V-SLAM system is built in an incremental way. Visual information in a newly captured image is first matched with the current map. Then, the positions of the filmed 3-d points are estimated together with the new camera pose. Lastly, the points and their descriptors are added to the map. As the camera travels more distance, the map becomes larger in size, and the error of pose estimation accumulates over time. When the camera revisits a scene, it can be relocalized in the map, the drift in its pose estimation can be eliminated, and (a part of) the map can be updated according to the constraint derived from the relocalization and become more accurate. This procedure is known as "*loop closure*". It contributes to the long-term accuracy of ego-motion estimation at the cost of a significant amount of computation.

In order to perform visual ego-motion estimation in real-time with the limited onboard computational resource, maintaining such a large-scale map is not an option. Instead, solely considering the visual information in the latest images within a sliding window can be enough for good short-term accuracy, while reducing the computational demand of long-term mapping and loop closure. Such an algorithm is commonly referred to as visual odometry (VO) [15, 16]. In the context of ORB-SLAM [11], a highly recognized V-SLAM system, VO corresponds to the tracking and local mapping components. Note that local mapping is only needed by VO algorithms that estimate the camera poses based on the estimated position of 3-d points in the scenes. This strategy is widely adopted by feature-based and direct VO solutions, but it is not indispensable. For example, as will be introduced later, an artificial neural network (ANN) can learn to directly regress to the relative pose using an image pair as input, without explicitly knowing the scene structure.

A monocular camera as the only sensor limits the robustness of ego-motion estimation. In the cases of visual tracking failure, *i.e.*, visual information in the newly captured image cannot match the old visual information well, or operating in a highly dynamic scene, a VO is forced to stop the current estimation and reinitialize or suffers from big drifts. An inertial measurement unit (IMU) made of a 3-axis accelerometer and a 3-axis gyroscope measures the 3-d angular rate and the 3-d translational acceleration caused by all forces applying to it except for gravity. As a complementary sensor to a camera, an IMU provides high-frequency measurements resistant to unfavorable conditions such as illumination change and fast motion. Algorithms that combine visual and inertial measurements and set ego-motion estimation as the main goal are called visual-inertial odometry

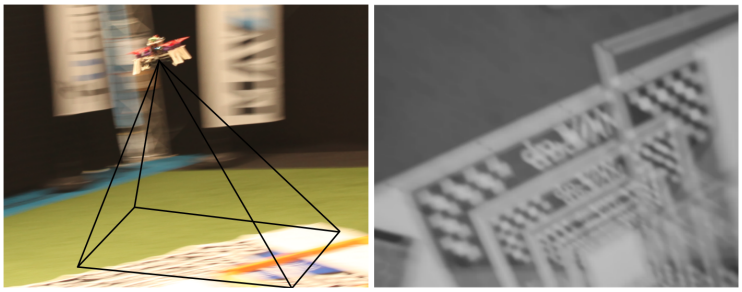


Figure 1.1: A quadrotor MAV in fast flight and a blurry image taken by its onboard camera.

(VIO) algorithms. In addition to higher accuracy, VIO has more robustness given that the ego-motion can be propagated by IMU measurements even when there is no reliable visual information. Besides, IMU measurements provide information on the metric scale and the direction of gravity [17], which are unobservable for a monocular VO.

To enable VIO to run in real-time onboard computation-constraint mobile devices, for instance lightweight MAVs, researchers have been working on improving the algorithmic efficiency of VIO. Compared to iterative optimization-based back-ends that involve many (previous) camera poses and world points, more computationally efficient VIO back-ends have been developed based on the extended Kalman filter (EKF). A smaller number of values essential to ego-motion estimation forms the state vector of the filter. For example, the computation in local mapping is reduced by one-shot calculation of the positions of points co-visible from multiple camera poses. The camera poses are then updated as filter states by the point position [18]. For [19], only the newest camera pose is kept in the state vector. Thanks to the robust visual tracking approach, the accuracy is decent even though the number of world points maintained in the state vector is relatively small. Besides algorithmic advances, thanks to more powerful mobile processors suitable for MAVs becoming available on the market, many works have emerged, which demonstrate onboard VIO guiding MAVs in autonomous flights [20–22]. The thorough research on this topic raises the question: is visual ego-motion estimation for MAVs a solved problem?

1.1. CHALLENGES BROUGHT BY AGILE MANUEVERS

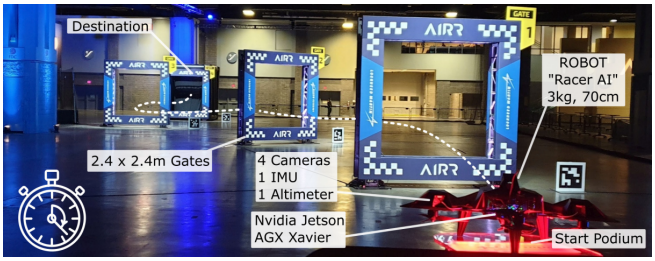
An advantage of MAVs over other types of robots is their flexibility. MAVs have no terrain requirements, and their small size allows operations in narrow spaces. But at the same time, the small size also prevents big onboard batteries. Given the limited battery life, flying fast is an important way to expand the range of flight. Before academia was attracted to high-speed and agile autonomous flight, there were already people who pursued pushing the speed of quadrotor MAVs to the limit. Such enthusiasm spawned a new e-sport called the drone race. Pilots fly along the predefined racing track indicated by the gates. The pilot who does not crash and uses the least time to finish the track wins. Human drone pilots wear goggles to see the video captured by a monocular camera mounted on the racing MAV and transmitted wirelessly to the goggles. The video is the only source of information from which the pilot estimates the position, attitude, and velocity of the MAV

and steers the MAV accordingly.

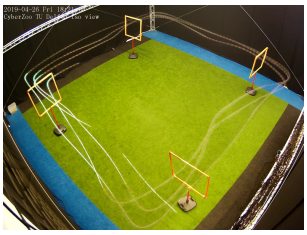
Autonomous drone racing (ADR) is a research topic that aims to replace the human pilot in drone racing with a machine. More specifically, the MAV uses its onboard sensing and processing resources to navigate itself on the racing track at high speed. Examples of ADR tracks are shown in Fig. 1.2. In pursuit of high-speed and agile autonomous flight, advances have been made in aerodynamic modeling [23], trajectory planning [24] and tracking [25]. When relying on a motion capture system for ego-motion information, autonomous MAVs achieved significantly fast speeds, even faster than some expert human pilots [26]. Progress is also made in visual ego-motion estimation and localization on the racing track. Since the appearance of the gates is pre-known, it is straightforward to localize the MAV in the track by detecting the gates in the image. Visual gate detection and a simple aerodynamic model of the quadrotor MAV are the key components of efficient ego-motion estimation solutions [27, 28]. Learning-based semantic segmentation for gate detection has been proven to be robust towards motion blur and illumination variance [28]. A problem of using vision only for gate detection is that the ego-motion estimation tends to drift when there is no gate in the field of view. When the onboard computational resource allows, VIO is a good choice to maintain the accuracy of ego-motion estimation. ADR solutions [21, 29] deploy a VIO [19] and use gate detection to compensate for VIO drift. In real-world applications where landmarks such as the gates are far away from each other, the accuracy of the VIO is a determining factor of whether the MAV succeeds in reaching the next landmark.

However, VIO can become less accurate in fast flight. Visual feature points based on image gradient [30–32] are widely used in the visual processing front-end of VIO. In high-speed translational motion and especially fast rotation that occurs when the MAV makes sharp turns, the optical flow in the camera's field of view massively increases, causing significant motion blur that lowers the image gradient. It becomes harder to detect or track feature points. The motion blur changes the observed appearance of the environment, which makes it hard to use direct photometric feedback [19, 33] as well. An example of motion blur is shown in Fig. 1.1. Another negative effect is that agile maneuvers result in feature points leaving the field of view quickly. Fewer observations of feature points are not only unfavorable to accuracy but also require more frequent new point detection and thus lead to more computational effort in image processing [20]. Some of these issues can be tackled by adopting novel hardware, *i.e.*, event cameras. The event camera is a new-generation vision sensor that is known for low latency, high temporal resolution, and high dynamic range [34]. It does not suffer from motion blur and thus can be a promising sensor for fast MAV flight. But an event camera provides little information in slow-speed flight and hover. It requires the MAV to keep uninterrupted motion to generate enough events. Also due to the much higher prices, event cameras are not likely to replace the dominating position of frame-based cameras on MAVs in the near future.

Given that deep learning outperforms conventional approaches in many computer vision tasks, researchers started to explore learning-based ego-motion estimation. Observed in [35, 36], convolutional neural networks (CNNs) predicting relative pose between consecutive images show robustness to motion blur. Although appealing, many learning-based works train and test on the same dataset. Thus the generalizability is not verified. In addition, techniques detecting outliers in network predictions are currently immature.



(a) A racing track of AIRR [28]. The gates were printed with patterns.



(b) A racing track consisting of four thin orange gates [27].

Figure 1.2: Two examples of autonomous drone racing tracks.

Thus, network performance is questioned in generic usage. Besides, pursuing higher accuracy leads to more and more layers and parameters, which makes the computation unaffordable for onboard processors.

Based on the above discussion and looking back to the question asked before, although VIO solutions satisfy the requirement of autonomous MAVs in many usage scenarios, more research is required for visual ego-motion estimation to gain more robustness towards the negative effects caused by agile maneuvers of MAVs. This dissertation mainly studies learning-based approaches, emphasizing generalization, algorithmic efficiency, and robustness to fast motion. Before diving into the solutions, an overview of previous works comes first in the remainder of this chapter, followed by the research questions and the outline of this dissertation.

1.2. PREVIOUS RESEARCH

This section is an overall description of the background and research progress related to this dissertation. The related works are divided into two types according to whether the visual information is used in an explicit way. The first type is mainstream and traditional. The visual information is explicitly used as pixel-level correspondences to establish constraints on camera motion based on multi-view geometry. The other way is based on deep learning. Artificial neural networks (ANNs) are trained to infer required information from raw images.

1.2.1. TRADITIONAL METHODS

After fast development over the past two decades, monocular VIO has grown into a big family. Survey papers [37–39] provide comprehensive and detailed introductions and discussions. The following is a brief overview of main-stream VIO solutions within which selected works are discussed in more detail because of their algorithmic efficiency and performance in high-speed flight.

The vision processing front-end of VIO solutions can be categorized into feature-based methods and direct methods. Feature-based pipelines extract image features, mostly point features [30, 40], according to the requirements of neighboring pixel intensities. Feature correspondences between images are established by feature tracking based on optical flow [41] or matching using descriptors [32, 42]. The constraints on camera poses and feature

point positions is derived from the 2-d reprojection error or epipolar geometry [43].

In contrast, direct methods take advantage of more image information. For example, LSD-SLAM [12] tracks all pixels with enough gradient and builds a semi-dense map. Using pixel intensity gradient, the optimization process adjusts camera pose and pixel depth to minimize the photometric error of the aligned pixels. Although directly working on raw pixels avoids extra computational resources on extracting and matching features, it is sensitive to the initialization of the camera pose and pixel depth, and how well the photometric consistency holds.

There are also hybrid methods that not only extract feature points but also directly establish correspondences using pixel intensities. Semi-direct visual odometry (SVO) [44] extracts FAST [40] point features only in keyframes and uses the direct method to align the image patches around the points. Processing sparse patches results in low processing time and thus brings SVO advanced time efficiency [45]. Robust visual-inertial odometry (ROVIO) [19] adopts the combination of FAST points and image patches as well. For MAVs, feature-based methods and hybrid methods are more popular thanks to their robustness and maturity. But feature points are sensitive to image gradient so image blur can lead to less robust detection and tracking. A mitigation measure adopted by OpenVINS [46] is to conduct image preprocessing by histogram equalization [47] to increase the contrast.

In terms of the state estimation back-end, the extended Kalman filter (EKF) is an option for fusing inertial and visual measurements. VIO solutions with such a back-end are often referred to as filter-based methods. For example, ROVIO [19] aligns sparse image patches iteratively using the photometric error as the innovation term in the measurement update of an IEKF (iterative EKF). The bearings and distances of 3-d world points captured by the image patch centers are included in the filter's state vector. When the number of tracked patches drops below a threshold, FAST feature points are extracted and the corresponding image patches surrounding them are added to the tracking process. The design of ROVIO enables it to operate on a single thread. Given its robustness, multi-camera capability and computational efficiency, ROVIO was adopted in autonomous drone racing [21]. But its real-time processing capability is constrained by the size of the state vector, making it difficult to track a large number of image patches at the same time.

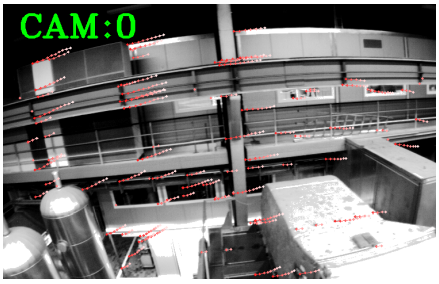
Multi-state constraint Kalman filter (MSCKF) [18] is one of the earliest successful VIO methods and has many variants [20, 46, 48, 49]. Its vision front-end is feature point detection and tracking. The 3-d positions of the points are not in the state vector of the EKF. Instead, the state vector is augmented by the stochastically cloned camera poses (position and orientation) of multiple image frames. The observations of a 3-d world point in multiple image frames encode the constraint of the camera motion. When the tracking of a feature point has been lost or the number of camera poses reaches the maximum, the visual measurements (bearing vectors) of this point in multiple frames are used to triangulate its 3-d position in a least-square manner. The residual in the EKF update is the reprojection error, *i.e.*, the difference between the 2-d observation of a world point and the 2-d projection of this point in the image plane according to its triangulated 3-d position. Thus, different from ROVIO which updates the camera pose once a new image is available, the visual update of MSCKF is triggered in a "delayed" manner that utilizes previous observations of a feature point. Because the position of a point is not maintained in the state vector, MSCKF can make use of more feature points without significantly increasing

the computational cost.

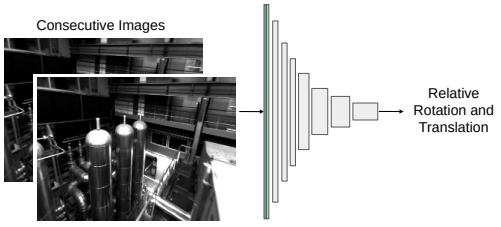
Another type of back-end is based on batch optimization, iteratively solving a nonlinear least-squares problem over a set of measurements. It is also known as bundle adjustment (BA). To prevent the computation to increase over time, recent states in a bounded-size sliding window are actively optimized while past states and measurements are marginalized. There are open-source libraries for solving the optimization problem. For example, the Ceres Solver [50] adopted by [51] and the general graph optimization (g2o) adopted by [11]. Compared to filter-based methods, BA-based methods weigh accuracy more than efficiency. It is often the choice when mapping the environment is required. Highly recognized BA-based works include OKVIS (open keyframe-based visual-inertial SLAM) [52], VINS-Mono (monocular visual-inertial system) [51], and ORB-SLAM3 [53].

VIO methods are compared with each other on benchmark datasets. The EuRoC MAV dataset [54] is the most used visual-inertial dataset collected by a multirotor MAV. There is motion blur caused by fast rotation in some sequences of this dataset. But in general, the flight speed is slow given the mobility of an MAV. In contrast, the UZH-FPV dataset [55] focuses more on fast flight. It was collected by a racing quadrotor MAV flown by an expert pilot. The speed of the MAV (up to 12.8 m/s indoor and 23.4 m/s outdoor) is much higher than EuRoC (2.3m/s indoor). As a result, the optical flow is much bigger and induces more motion blur. According to [56], MSCKF variants using a monocular camera [46, 49], a variant of VINS-Mono using a stereo camera [57], and variants of OKVIS using a stereo camera [58, 59] perform well on this dataset.

There are also works that utilize the aerodynamic model of the quadrotor MAV in ego-motion estimation. A velocity estimator was proposed by [60]. It adopts an aerodynamic model that calculates the thrust force and the drag force of the propellers from rotor speeds. The camera is only used for estimating the relative yaw angle of an image pair using the correspondences of feature points. VIMO (visual inertial model-based odometry) [61] extends [51] by establishing a residual term derived from the dynamic model of the quadrotor and adding it to BA.



(a) The trajectories of FAST feature points [40] plotted by OpenVINS [46]. The image is from the EuRoC dataset [54].



(b) A basic ANN for relative pose prediction. Multiple convolutional layers extract information from the channel-dimension concatenated images and downscale the feature map.

Figure 1.3: Two categories of vision front-end introduced in this section. The mainstream is detecting and tracking visual feature points. A newer approach is to train an ANN that predicts the relative pose of the input images.

1.2.2. LEARNING-BASED METHODS

In the presence of significant motion blur, visual features are harder to detect and track because of less image gradient. It is also problematic to assume that the pixels filming the same 3-d world point in different images maintain constant intensities. Human beings can recognize most objects even in a very blurry image and approximately predict the camera motion. This "intelligence" can be explained by the experiences that humans have learned throughout their lifetime. Researchers have been working on training artificial neural networks (ANNs) to predict the relative pose of a moving camera to empower ANNs with similar intelligence in the presence of image blur and other possible unfavorable conditions [35, 36, 62, 63], *e.g.*, varying illumination, motion blur, and dynamic objects. In the rest of this dissertation, an ANN predicting the camera's relative rotation and translation between the time points when the input images are captured is referred to as a *pose network*.

Training a pose network is straightforward when ground-truth pose labels are available. The input of a pose network can be an image pair with overlapping contents [35, 64–66] or a dense optical flow map of the image pair [36, 62]. The flow map explicitly expresses the pixel-wise matching information that encodes the motion of the camera, assuming a mostly static scene. There are works that use both images and IMU measurements as input to the network, *i.e.*, learning-based VIO solutions. VINet [64] is an end-to-end trainable VIO solution supervised by ground truth. Two separate networks are in charge of visual processing and inertial processing, respectively, at different sensor rates. They output intermediate tensors without physical meaning. Another network takes the concatenated output tensors as input to perform sensor fusion and pose prediction. Long Short-Term Memory (LSTM) is utilized to process IMU measurements in [64, 65] to retain the effects of past input on the current prediction. Instead of training an IMU network, the learning-based VIO proposed in [66] integrates IMU measurements to propagate motion states, since IMU has well-understood models grounded in physics. The pose network predictions are used in the EKF updating. The ground truth poses supervise not only the pose network predictions but also the *a posteriori* poses after sensor fusion. The training is end-to-end thanks to the differentiable extended Kalman filter (EKF).

As ground-truth pose is expensive to obtain in the real world, datasets with labels are often limited in size. This is the reason that *self-supervised learning* draws great research interest. For temporally consecutive images, the captured scenes usually have an overlap. The intensities of the pixels filming the overlapping scene can be used as a constraint of the camera motion. SfMLearner [67] first proposed to simultaneously train a pose network and a monocular depth network. The pose network predicts the relative pose $T_{t\to s}$ between the source image I_s and the target image I_t. The depth network predicts D_t, the pixel-wise depth map of I_t. An image $\widetilde{I}_s$ can be synthesized by warping I_s according to the 2-d projections of the 3-d point cloud established from D_t in the image plane of I_s located at $T_{t\to s}$. Minimizing the photometric difference between $\widetilde{I}_s$ and I_t leads to accurate $T_{t\to s}$ and D_t, when the following assumptions hold: the scene is static, the camera has enough translational motion, the pixel is visible in both images, and the scene appearance keeps constant in different images. In summary, this self-supervised loss function is derived from the temporal consistency of the structure and appearance. It is referred to as the reprojection-based loss in some works and in this dissertation because $\widetilde{I}_s$ is synthesized

through the reprojection of 3-d points defined by D_t.

This scheme has been further improved by many following works. Monodepth2 [68] proposed a pixel-wise auto-masking strategy that uses more than one source images to construct multiple photometric error maps with the target image. For each pixel, only the smallest photometric error in the error maps is minimized by network training. This strategy can avoid minimizing the incorrect loss derived from a world point occluded in a source image. Another strategy is to exclude pixels at whose locations the intensities have little change in consecutive images. Because its constant intensity can be caused by objects moving at the same velocity as the camera or a stationary camera. Both situations violate the assumptions of the reprojection-based loss. Works [69, 70] introduce the 3-d geometric consistency constraint into the loss function. Both the depth maps D_t and D_s of the images I_t and I_s are predicted by the depth network. D_t and D_s and $T_{t\to s}$ are adjusted together in training to align the two point clouds corresponding to D_t and D_s.

For the above-mentioned works, the scales of the translational motion and depth are trained to be consistent through the gradient flow. At inference time, the two networks run independently, so their scales are decoupled. In [71] and [72], the pose network inference is conducted for multiple times by iterative view synthesis. In the first iteration, the inputs of the pose network are the original I_s and I_t. With the first pose prediction $T_{t\to s}$ and D_t, the warped source image $\widetilde{I}_{s,1}$ can be synthesized. I_t should have smaller visual disparities with $\widetilde{I}_{s,1}$ than with I_s. In the second iteration, the pose network infers the relative pose $T_{t\to s,1}$ from the inputs $\widetilde{I}_{s,1}$ and I_t. $T_{t\to s,1}$ is then composed to $T_{t\to s}$. The composed transformation is used to synthesize the $\widetilde{I}_{s,2}$ that is an input of the third iteration. In this way, the pose network incrementally refines the transformation by inferring from more and more similar image pairs. The input image $\widetilde{I}_{s,i}$ is a function of D_t. So their scale is coupled.

Learning-based VIO can also perform self-supervised learning using the reprojection-based loss. Same as [64, 65], SelfVIO [73] has three networks for vision, IMU, and fusion, respectively. The relative pose $T_{t\to s}$ in the reprojection-based loss is the output of the fusion network. Although accelerometer measurements have metric scale, as pointed out in [63], a network has no knowledge of the IMU kinematic so the metric scale in sensor measurements is not preserved. The scale of translational motion and depth is still unknown. Wagstaff *et al.* [63] extended [66] to a self-supervised monocular VIO with metric scale. The scale of IMU data is explicitly kept in the state propagation based on IMU integration. A depth network predicts a depth map that constructs the reprojection-based loss together with the *a posteriori* pose of the differentiable EKF.

Besides EKF, pose and depth predictions have also been involved in optimization-based VIO back-ends. In D3VO [74], network predictions are integrated into a traditional direct visual odometry framework [33]. Depth predictions are leveraged to initialize the sparse depths that are later optimized by the photometric BA. The predicted relative pose not only initializes the pose in the optimization but also builds a factor graph of poses as a regularizer. The depth network additionally predicts a pixel-wise uncertainty map reflecting how well the appearance consistency and the static scene assumption hold. The uncertainty serves as the weight of the photometric energy in BA.

The works discussed above have a pose network to directly regress to the relative pose between an image pair. Besides, deep learning can be used in other ways to contribute

to ego-motion estimation. Feature points as the vision front-end of VIO can be detected [75, 76] and matched [77, 78] using learning-based approaches. The authors of [79] use learning-based dense optical flow [80] to track feature points. DF-VO [81] calculates the relative pose by solving the essential matrix of epipolar geometry according to the pixel correspondences predicted by an optical flow network. DROID-SLAM [82] also adopts an optical flow network. Its predictions are used as constraints for a BA back-end that optimizes camera poses. A learnable update operator iteratively refines the dense optical flow map. The BA maps an optical flow revision and its confidence map to the pose and depth update, ensuring that the reprojected points match the revised optical flow. DROID-SLAM is trained end-to-end thanks to the differentiable BA layer. The camera pose and the optical flow induced by the estimated depth and pose are supervised by the ground truth.

Compared with traditional solutions, learning-based ego-motion estimation has not been widely accepted for navigating MAVs. The reasons can be summarized as three-fold. The first one is the high computational demand. In the pursuit of accuracy, the size of networks grows. Iterative inferencing [71, 72, 82] can not be paralleled and thus further increases the overall time consumption. The optimization-based back-end is another computational burden. For instance, DROID-SLAM [82] is a computationally demanding system that requires a powerful NVIDIA GeForce RTX-3090 GPU for tracking and local BA to process the monocular video of EuRoC dataset in real-time, with images downsampled and frame rate reduced to 10 fps.

The second reason is the lower accuracy than traditional solutions when tested on an MAV dataset. Many approaches achieve higher accuracy than traditional approaches on KITTI [83], a car dataset with camera motion in three degrees of freedom (DoF). A few works [35, 36, 63–66, 74, 82] expanded their evaluation to the EuRoC [54] MAV dataset. Compared to KITTI, EuRoC has a smaller number of training samples, more difficult motion patterns (6 DoFs), and more complex environmental factors. Only TartanVO [36], D3VO [74], and DROID-SLAM [82] outperform traditional approaches on EuRoC. TartanVO uses an optical flow map as the input to the pose network, and the translational motion prediction has no scale. D3VO and DROID-SLAM have BA-based back-ends. All of them require too high computational power for an MAV.

The third reason is about failure detection. For feature point matching, random sample consensus (RANSAC) [84] is well-recognized to be robust and effective. Although ANNs can show high average accuracy and low outlier rate on the testing dataset, the absence of RANSAC-like failure detection techniques gives the impression that a very wrong network prediction is possible and unpredictable. It is a big obstacle for ANNs to be deployed for ego-motion estimation which is critical for flight safety.

1.3. RESEARCH OBJECTIVES AND QUESTIONS

This dissertation focuses on developing visual ego-motion estimation solutions that are robust toward agile MAV maneuvers and efficient enough for onboard processors. The main research goal is formulated as follows.

Research Goal

To develop real-time onboard-processing visual ego-motion estimation solutions for autonomous MAVs deployable in unknown environments. The visual processing must maintain robustness during agile maneuvers.

The key requirements can be summarized as the following aspects:

- Efficiency: Trade-off accuracy and efficiency. Perform real-time processing with a limited sacrifice of accuracy.
- Robustness: Maintain the prediction accuracy in general in the appearance of significant motion blur and reveal inaccurate predictions by estimating the prediction uncertainty.
- Generalization: Be deployable in unknown environments and have no strict requirement on the filmed environmental structures.

The research objective is further split into four research questions, formulated as follows. In some applications, computational power for ego-motion estimation is very limited and long-term accurate ego-motion estimation is not necessary. It suffices to ensure that the error remains within acceptable bounds until it reaches the next waypoint where the accumulated error can be corrected. It is better that the estimator remains functional when visual information becomes unavailable suddenly, *e.g.*, motion blur is too much for detecting and tracking feature points. Hence, the first research question is:

Research Question 1

How to design an ego-motion estimator that uses as little computing power as possible while maintaining acceptable accuracy?

Although wrong matches of visual feature points can be detected and rejected, the numbers of detected and tracked points drop facing motion blur. In contrast, ANNs have been found to possess a degree of resistance to image blur [35, 62, 63]. However, the generalization and robustness of ANNs have not been verified in real-world experiments with high-speed MAVs. A relatively simple application scenario can help explore the generalization, robustness, and efficiency of ANN in MAV ego-motion estimation. Thus the specific research question is:

Research Question 2

Can an ANN maintain the accuracy of inferring translation velocity from blurry images captured by a downward-facing camera while achieving real-time performance on an onboard processor and generalization to unknown environments?

The previously discussed ANN only predicts translational motion and lacks the ability to evaluate the accuracy of its predictions. This poses a hidden danger as an erroneous

prediction could cause disastrous errors in ego-motion estimation. To achieve higher accuracy in ego-motion estimation, the ANN prediction should also reflect camera rotation. Moreover, it is crucial to have network uncertainty estimation that weighs the prediction confidence. Self-supervised learning is preferred because the requirement for ground-truth labels can limit the amount of training data, thereby compromising generalization. Therefore, the third research question is:

Research Question 3

How to train a planar homography network without using ground-truth labels, estimate the prediction uncertainty of the network, and build an efficient and accurate VIO upon it?

If a visual processing algorithm can only handle mainly planar surfaces, its application scope becomes limited. Compared to the homograph network mentioned above, pose networks that regress camera rotation and translation from images of general 3-d structures are more suitable for small MAVs equipped with a single forward-facing camera. Training a lightweight and uncertainty-aware pose network without using ground-truth labels requires highly accurate target poses obtained from self-supervised learning. Since the self-supervised learning of a pose network for 3-d structures requires depth information, a depth network can be trained simultaneously. For MAVs, the depth network should provide sufficient information for obstacle avoidance while remaining lightweight. Considering both pose and depth, the fourth research question is:

Research Question 4

How to obtain accurate network predictions of pose and depth without ground truth and use them to train a lightweight uncertainty-aware pose network and a lightweight depth network for a forward-facing camera?

1.4. DISSERTATION OUTLINE

This dissertation has six chapters. The current Introduction chapter introduces the background, current state-of-the-art research outcomes, and the remaining challenges. Research questions are then raised accordingly. In the following four chapters, the above research questions are investigated in detail one by one.

In Chapter 2, an EKF-based ego-motion estimator is proposed. It detects and tracks traditional visual feature points and aims at minimizing computational demand. The frequency of feature point detection operation is reduced for less time consumption in visual processing. A linear aerodynamic model of the drag force is utilized. It leads to bounded estimation errors in the velocity components orthogonal to the shafts of propellers and the attitude relative to the gravity direction, even when vision information is unavailable. The gravity-related attitude is then taken as known information to simplify the RANSAC-based calculation of the relative heading angle and the translational direction, using the epipolar constraints.

After digging into the potential of higher efficiency given the traditional vision front-

end, Chapter 3 switches the front-end to ANNs and explores their ability to handle motion blur better. Using a de-rotated image pair as input, the networks predict 3-d translational motion that is scaled by the distance from the camera to the observed planar surface. To preserve accuracy while squeezing network inference time cost, the network architecture of cascaded network blocks connected by image warping is adopted. The network training is self-supervised on a big-scale synthetic dataset. Experiments verify the network's generalization, inference efficiency, and robustness towards motion blur.

Building upon the translation network that needs known rotation developed in Chapter 3, Chapter 4 expands it to predict the full 8-d planar homography transformation. This homography network is able to estimate its prediction uncertainty, utilize the prior knowledge of camera motion, and maintain high and stable inference efficiency. Additionally, accounting for 3-d objects on the ground improves the network's adaptability to environments. The filter-based VIO designed based on the uncertainty-aware homography network balances accuracy and efficiency well, rivaling mainstream VIO solutions.

Chapter 5 inherits the methodology developed in Chapter 4, which is for training a network that can estimate its prediction uncertainty without requiring ground-truth labels. It is applied to a pose network that regresses rotation and translation from image pairs capturing general 3-d structures. The joint training scheme that involves training a pose network and a monocular depth network simultaneously is enhanced by iterative pose network inference based on depth-dependent view synthesis and produces higher prediction accuracy. The more accurate pose prediction serves as the learning target for the lightweight uncertainty-aware pose network. Additionally, this chapter explores how to increase the prediction accuracy of a lightweight depth network that is trained based on the joint training scheme.

Chapter 6, the final chapter, summarizes the answers to the research questions and concludes that the proposed ego-motion estimation solutions are efficient enough for MAVs and show robustness towards motion blur. In addition, more profound thoughts derived from the development process of the solutions are discussed and are expected to point out issues to be addressed for future research.

REFERENCES

[1] Wikipedia, *Global positioning system,* `https://en.wikipedia.org/wiki/Global_Positioning_System` (2023), accessed on 3rd April 2023.

[2] Bitcraze, *Motion capture positioning,* `https://www.bitcraze.io/documentation/system/positioning/mocap-positioning/` (2023), accessed on 3rd April 2023.

[3] M. W. Mueller, M. Hamer, and R. D'Andrea, *Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation,* in *2015 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2015) pp. 1730–1736.

[4] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, *Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping,* in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (IEEE, 2020) pp. 5135–5142.

[5] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, *Fast-lio2: Fast direct lidar-inertial odometry,* IEEE Transactions on Robotics **38**, 2053 (2022).

[6] S. Li, C. De Wagter, and G. C. De Croon, *Self-supervised monocular multi-robot relative localization with efficient deep neural networks,* in *2022 International Conference on Robotics and Automation (ICRA)* (IEEE, 2022) pp. 9689–9695.

[7] G. C. De Croon, J. J. Dupeyroux, C. De Wagter, A. Chatterjee, D. A. Olejnik, and F. Ruffier, *Accommodating unobservability to control flight attitude with optic flow,* Nature **610**, 485 (2022).

[8] PX4Autopilot, *Px4flow smart camera,* `https://docs.px4.io/main/en/sensor/px4flow.html` (2023), accessed on 3rd April 2023.

[9] Bitcraze, *Flow deck v2,* `https://www.bitcraze.io/products/flow-deck-v2/` (2023), accessed on 3rd April 2023.

[10] P.-J. Bristeau, F. Callou, D. Vissière, and N. Petit, *The navigation and control technology inside the ar. drone micro uav,* IFAC Proceedings Volumes **44**, 1477 (2011).

[11] R. Mur-Artal and J. D. Tardós, *Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,* IEEE transactions on robotics **33**, 1255 (2017).

[12] 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.

[13] D. Gehrig, M. Göttgens, B. Paden, and E. Frazzoli, *Scale-corrected monocular-slam for the ar. drone 2.0,* (2017).

[14] S. H. Lee and G. de Croon, *Stability-based scale estimation for monocular slam,* IEEE Robotics and Automation Letters **3**, 780 (2018).

[15] D. Scaramuzza and F. Fraundorfer, *Visual odometry [tutorial],* IEEE robotics & automation magazine **18**, 80 (2011).

[16] M. O. Aqel, M. H. Marhaban, M. I. Saripan, and N. B. Ismail, *Review of visual odometry: types, approaches, challenges, and applications,* SpringerPlus **5**, 1 (2016).

[17] A. Martinelli, *Vision and imu data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination,* IEEE Transactions on Robotics **28**, 44 (2011).

[18] A. I. Mourikis and S. I. Roumeliotis, *A multi-state constraint kalman filter for vision-aided inertial navigation,* in *Proceedings 2007 IEEE international conference on robotics and automation* (IEEE, 2007) pp. 3565–3572.

[19] 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 **36**, 1053 (2017).

[20] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, *Robust stereo visual inertial odometry for fast autonomous flight,* IEEE Robotics and Automation Letters **3**, 965 (2018).

[21] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, *Alphapilot: Autonomous drone racing,* Autonomous Robots **46**, 307 (2022).

[22] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, *et al.*, *Swarm of micro flying robots in the wild,* Science Robotics **7**, eabm5954 (2022).

[23] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza, *Neurobem: Hybrid aerodynamic quadrotor model,* arXiv preprint arXiv:2106.08015 (2021).

[24] P. Foehn, A. Romero, and D. Scaramuzza, *Time-optimal planning for quadrotor waypoint flight,* Science Robotics **6**, eabh1221 (2021).

[25] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, *A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight,* IEEE Transactions on Robotics **38**, 3357 (2022).

[26] D. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Penicka, Y. Song, G. Cioffi, E. Kaufmann, and D. Scaramuzza, *Past, present, and future of autonomous drone racing: A survey,* arXiv preprint arXiv:2301.01755 (2023).

[27] S. Li, E. van der Horst, P. Duernay, C. De Wagter, and G. C. de Croon, *Visual model-predictive localization for computationally efficient autonomous racing of a 72-g drone,* Journal of Field Robotics **37**, 667 (2020).

[28] C. De Wagter, F. Paredes-Vallés, N. Sheth, and G. de Croon, *The sensing state-estimation and control behind the winning entry to the 2019 artificial intelligence robotic racing competition,* Field Robot. **2**, 1263 (2022).

[29] E. Kaufmann, M. Gehrig, P. Foehn, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, *Beauty and the beast: Optimal methods meet learning for drone racing,* in *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019) pp. 690–696.

[30] J. Shi *et al.*, *Good features to track,* in *1994 Proceedings of IEEE conference on computer vision and pattern recognition* (IEEE, 1994) pp. 593–600.

[31] M. Trajković and M. Hedley, *Fast corner detection,* Image and vision computing **16**, 75 (1998).

[32] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, *Orb: An efficient alternative to sift or surf,* in *2011 International conference on computer vision* (Ieee, 2011) pp. 2564–2571.

[33] J. Engel, V. Koltun, and D. Cremers, *Direct sparse odometry,* IEEE transactions on pattern analysis and machine intelligence **40**, 611 (2017).

[34] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, *et al.*, *Event-based vision: A survey,* IEEE transactions on pattern analysis and machine intelligence **44**, 154 (2020).

[35] S. Wang, R. Clark, H. Wen, and N. Trigoni, *End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks,* The International Journal of Robotics Research **37**, 513 (2018).

[36] W. Wang, Y. Hu, and S. Scherer, *Tartanvo: A generalizable learning-based vo,* in *Conference on Robot Learning* (PMLR, 2021) pp. 1761–1772.

[37] C. Chen, H. Zhu, M. Li, and S. You, *A review of visual-inertial simultaneous localization and mapping from filtering-based and optimization-based perspectives,* Robotics **7**, 45 (2018).

[38] G. Huang, *Visual-inertial navigation: A concise review,* in *2019 international conference on robotics and automation (ICRA)* (IEEE, 2019) pp. 9572–9582.

[39] M. Servières, V. Renaudin, A. Dupuis, and N. Antigny, *Visual and visual-inertial slam: State of the art, classification, and experimental benchmarking,* Journal of Sensors **2021**, 1 (2021).

[40] E. Rosten and T. Drummond, *Machine learning for high-speed corner detection,* in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9* (Springer, 2006) pp. 430–443.

[41] B. D. Lucas and T. Kanade, *An iterative image registration technique with an application to stereo vision,* in *IJCAI'81: 7th international joint conference on Artificial intelligence*, Vol. 2 (1981) pp. 674–679.

[42] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, *Brief: Binary robust independent elementary features,* in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11* (Springer, 2010) pp. 778–792.

[43] Wikipedia, *Epipolar geometry,* `https://en.wikipedia.org/wiki/Epipolar_geometry` (2022), accessed on 3rd April 2023.

[44] 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**, 249 (2016).

[45] 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)* (IEEE, 2018) pp. 2502–2509.

[46] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, *Openvins: A research platform for visual-inertial estimation,* in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2020) pp. 4666–4672.

[47] OpenCV, *Histogram equalization,* https://docs.opencv.org/3.4/d4/d1b/tutorial_histogram_equalization.html (2023), accessed on 3rd April 2023.

[48] M. Li and A. I. Mourikis, *High-precision, consistent ekf-based visual-inertial odometry,* The International Journal of Robotics Research **32**, 690 (2013).

[49] Q. Xiaochen, H. Zhang, and F. Wenxing, *Lightweight hybrid visual-inertial odometry with closed-form zero velocity update,* Chinese Journal of Aeronautics **33**, 3344 (2020).

[50] S. Agarwal, K. Mierle, and T. C. S. Team, *Ceres Solver,* (2022).

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

[52] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, *Keyframe-based visual–inertial odometry using nonlinear optimization,* The International Journal of Robotics Research **34**, 314 (2015).

[53] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, *Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam,* IEEE Transactions on Robotics **37**, 1874 (2021).

[54] 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 **35**, 1157 (2016).

[55] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, *Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset,* in *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019) pp. 6713–6719.

[56] Robotics and U. o. Z. Perception Group, *Uzh fpv leader board,* https://fpv.ifi.uzh.ch/uzh/uzh-fpv-leader-board/ (2023), accessed on 3rd April 2023.

[57] H. Zhang and C. Ye, *Vins-stereo for the fpv drone racing vio competition 2020,* .

[58] S. Leutenegger, *Okvis 2.0 for the fpv drone racing vio competition 2020,* (2020).

[59] J. Huai and Y. Lin, *A keyframe-based sliding window filter,* .

[60] J. Svacha, G. Loianno, and V. Kumar, *Inertial yaw-independent velocity and attitude estimation for high-speed quadrotor flight,* IEEE Robotics and Automation Letters **4**, 1109 (2019).

[61] B. Nisar, P. Foehn, D. Falanga, and D. Scaramuzza, *Vimo: Simultaneous visual inertial model-based odometry and force estimation,* IEEE Robotics and Automation Letters **4**, 2785 (2019).

[62] 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**, 18 (2015).

[63] B. Wagstaff, E. Wise, and J. Kelly, *A self-supervised, differentiable kalman filter for uncertainty-aware visual-inertial odometry,* arXiv preprint arXiv:2203.07207 (2022).

[64] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, *Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem,* in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31 (2017).

[65] L. Han, Y. Lin, G. Du, and S. Lian, *Deepvio: Self-supervised deep learning of monocular visual inertial odometry using 3d geometric constraints,* in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2019) pp. 6906–6913.

[66] C. Li and S. L. Waslander, *Towards end-to-end learning of visual inertial odometry with an ekf,* in *2020 17th Conference on Computer and Robot Vision (CRV)* (IEEE, 2020) pp. 190–197.

[67] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, *Unsupervised learning of depth and ego-motion from video,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017) pp. 1851–1858.

[68] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, *Digging into self-supervised monocular depth estimation,* in *Proceedings of the IEEE/CVF international conference on computer vision* (2019) pp. 3828–3838.

[69] R. Mahjourian, M. Wicke, and A. Angelova, *Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018) pp. 5667–5675.

[70] J. Bian, Z. Li, N. Wang, H. Zhan, C. Shen, M.-M. Cheng, and I. Reid, *Unsupervised scale-consistent depth and ego-motion learning from monocular video,* Advances in neural information processing systems **32** (2019).

[71] M. Hosseinzadeh, R. Fahimi, Y. Wang, *et al.*, *Unsupervised learning of camera pose with compositional re-estimation,* in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2020) pp. 11–20.

[72] B. Wagstaff, V. Peretroukhin, and J. Kelly, *On the coupling of depth and egomotion networks for self-supervised structure from motion,* IEEE Robotics and Automation Letters **7**, 6766 (2022).

[73] Y. Almalioglu, M. Turan, M. R. U. Saputra, P. P. de Gusmão, A. Markham, and N. Trigoni, *Selfvio: Self-supervised deep monocular visual–inertial odometry and depth estimation,* Neural Networks **150**, 119 (2022).

[74] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers, *D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020) pp. 1281–1292.

[75] D. DeTone, T. Malisiewicz, and A. Rabinovich, *Superpoint: Self-supervised interest point detection and description,* in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (2018) pp. 224–236.

[76] A. B. Laguna and K. Mikolajczyk, *Key. net: Keypoint detection by handcrafted and learned cnn filters revisited,* IEEE Transactions on Pattern Analysis and Machine Intelligence (2022).

[77] K. M. Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua, *Learning to find good correspondences,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018) pp. 2666–2674.

[78] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, *Superglue: Learning feature matching with graph neural networks,* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020) pp. 4938–4947.

[79] C. Huang, R. Yan, and X. Liu, *A Filter-Based Visual-Inertial Odometry with RAFT*, Tech. Rep. (Megvii, Tech. Rep, 2020).

[80] Z. Teed and J. Deng, *Raft: Recurrent all-pairs field transforms for optical flow,* in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16* (Springer, 2020) pp. 402–419.

[81] 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.

[82] Z. Teed and J. Deng, *Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras,* Advances in neural information processing systems **34**, 16558 (2021).

[83] A. Geiger, P. Lenz, and R. Urtasun, *Are we ready for autonomous driving? the kitti vision benchmark suite,* in *2012 IEEE conference on computer vision and pattern recognition* (IEEE, 2012) pp. 3354–3361.

[84] Wikipedia, *Random sample consensus,* `https://en.wikipedia.org/wiki/Random_sample_consensus` (2023), accessed on 3rd April 2023.

2

EFFICIENT MODEL-AIDED VISUAL-INERTIAL EGO-MOTION ESTIMATION FOR MULTIROTOR MICRO AIR VEHICLES

A promising approach to ego-motion estimation of multirotor micro air vehicles (MAVs) is to fuse information from a monocular camera and an inertial measurement unit (IMU). Visual-inertial odometry (VIO) solutions face an efficiency-accuracy trade-off when they are deployed onboard small-sized MAVs with limited processing power. This chapter proposes an aerodynamic-model-aided approach that emphasizes time efficiency over estimation accuracy. A linear drag force model of propellers guarantees bounded estimation errors in the velocity components orthogonal to the shafts of propellers and the attitude relative to the gravity direction. Feature point correspondences are extracted from the monocular image stream to compute the relative heading angle and translational direction, which is fused with inertial measurements by an extended Kalman filter (EKF) in a loosely coupled manner. We evaluate our approach on a publicly-available dataset and compare its accuracy and time efficiency against a state-of-the-art approach. It shows balanced performance in accuracy and efficiency. The robustness to the situations where vision information becomes unavailable is also observed.

Parts of this chapter have been accepted by the International Micro Air Vehicle Conference and Competition (2023).

2.1. INTRODUCTION

The autonomous flight of micro air vehicles (MAVs) in GPS-denied environments is a very challenging yet very popular problem in the field of robotics research. For large flying robots (*i.e.* heavier than ~300 grams, and larger than ~50 centimeters in diameter), this task is substantially simpler since they are not constraint by computational or payload capacity, and can process information from numerous sensors. However, they require more flying space and stringent safety checks, and are generally more expensive. On the other hand, smaller palm-sized drones [1, 2] are a promising alternative as they are safer and cheaper, but come with the disadvantage that they are generally limited in terms of resources. This limitation makes the combination of a monocular camera and an inertial measurement unit (IMU) the most promising option for ego-motion estimation. The camera provides abundant up-to-scale information about the surroundings, and the metric scale can be recovered using accelerometers or by means of control stability [3, 4].

Visual-inertial odometry (VIO) [5–9] has shown good performance in MAV navigation in recent years. It is intuitive to perform ego-motion estimation and environment mapping simultaneously by taking the reprojection error from observations of the same landmark in different frames and IMU measurements as constraints between the poses of frames and landmarks' locations. Environment mapping not only contributes to the accuracy of the ego-motion estimation, but also provides an obstacle map for motion planning. However, this comes at the cost of a complex iterative batch graph optimization and the need for an initialization procedure, which may be required multiple times in case of tracking failure. Additionally, to perform the mapping, the poses of landmarks need to be treated as states to be estimated. Even for indirect approaches [6, 7], which only map sparse-point features and use a sliding window to bound the optimization, the computational demand of this task is still relatively high [10].

Focusing on ego-motion, the Multi-State Constraint Kalman Filter (MSCKF) [5] also maintains a window of poses but does not optimize the location of points. It is based on an extended Kalman filter (EKF) whose state vector is augmented with poses of the previous frames that observed the same features. The least-squares location of one point is only calculated once after this point is no longer tracked. Its reprojection errors, which express geometric constraints between the frames' poses, are then used to perform visual measurement updates. Instead of reprojecting three-dimensional points into frames, [9] enforces multiple constraints on a pair of frames directly through the pixel locations of their point correspondences. The drift in the relative pose is reflected by the residual of the epipolar geometry constraint. The poses of the frame pair are updated by this residual in an EKF. The absence of 3D point locations further reduces computational demand, and only one frame's pose is needed to augment the state vector. However, since the pixel location of the tracked point goes directly into the measurement equation and point correspondences perform updating one by one, pose accuracy suffers from frame-to-frame feature tracking noise. This one-point-one-time updating also requires more processing time.

The approaches mentioned above use the raw visual features processed together with IMU measurements in a tightly-coupled manner. Visual odometry systems like SVO [11], which can produce up-to-scale environment map and ego-motion estimation, can be loosely coupled with IMU measurements by an EKF [12, 13]. The EKF-based, loosely-

coupled approach SVO+MSF [14] and the EKF-based, tightly-coupled, map-less MSCKF outperform others in efficiency among several openly-available visual-inertial ego-motion estimation solutions [10].

In windless environments, since all the aerodynamic force acting on an MAV is caused by propeller rotation and ego-motion, the aerodynamic model becomes an additional source of information for ego-motion estimation. As shown in [15], a simplified linear drag model in the propeller plane can be combined with IMU measurements to estimate horizontal components of attitude and velocity in the multirotor's body frame. Unlike estimation from purely integrated IMU measurements, the error of this model-aided estimation does not increase over time [15]. It is related to IMU bias and model fidelity. This approach has been used for velocity prediction in visual-inertial ego-motion estimation [1, 9] and trajectory tracking [16]. In [17], high estimation accuracy was reached with a more precise dynamic model that takes into account thrust forces and the effect of rotor speed on the drag, which makes the vertical speed in the body frame to be observable. In this case, the camera provides point correspondences only to estimate the relative yaw angle between pair of frames. Worth noticing is that the drift-free roll and pitch angles were used as known values to simplify calculating the essential matrix.

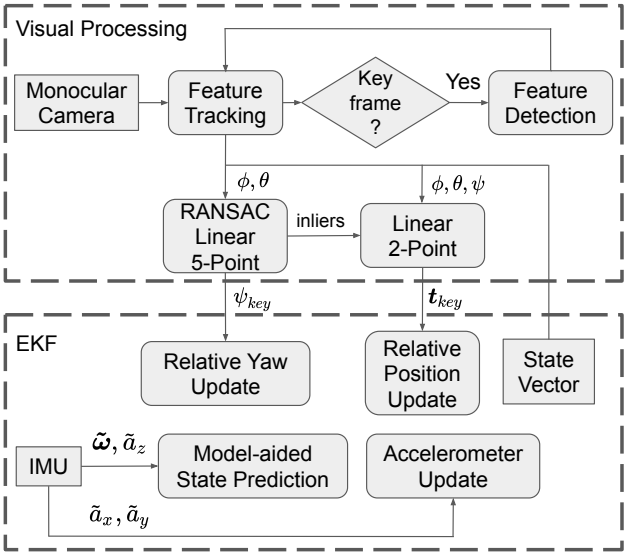


Figure 2.1: Pipeline of the proposed approach.

To gain robustness and accuracy with as little computation as possible, we propose an approach that combines all the previously mentioned strategies that benefit time efficiency. It is a map-less, model-aided, EKF-based, loosely-coupled ego-motion estimator for multirotor MAVs. The linear drag force model prevents attitude and velocity estimation in the body's horizontal plane from drifting over time when there is no visual information. The relative heading angle and the direction of translational motion between two frames are calculated from visual feature point correspondences using the epipolar constraint.

The attitude estimation is taken as known information to simplify the visual pose calculation. The visual update executes in a one-frame-one-time manner.

The proposed ego-motion estimator has a relatively low-complexity modular pipeline that is easy to implement and debug. We choose the EuRoC [18] dataset as the validation tool and compare our approach with MSCKF [5], a relatively efficient (yet accurate) VIO solution. The accuracy of the proposed approach is compromised due to the prioritized efficiency. But it is sufficient for short-time navigation. And it is observed that the estimator can maintain its accuracy when visual information is no longer available.

2.2. ESTIMATOR FRAMEWORK

The proposed ego-motion estimator pipeline is shown in Fig. 2.1. The *Visual Processing* module receives estimated roll and pitch angles from the *EKF* module. The relative pose is calculated from the feature point correspondences in the current frame and the keyframe based on the estimated roll and pitch angles. The relative yaw angle and the direction of translation with respect to the keyframe are taken as measurements for the visual update step of the EKF.

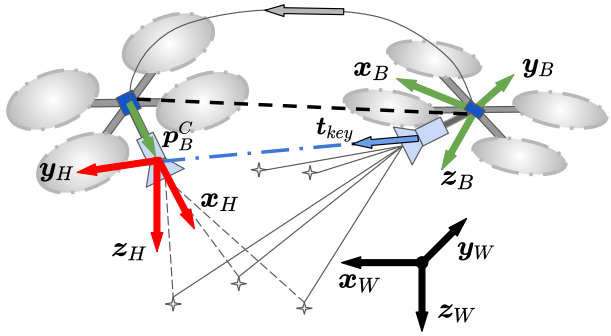


Figure 2.2: Schematic that illustrates the definition of the different coordinate frames and the motion of the MAV from the key frame pose to the current pose.

2.2.1. DEFINITIONS

In this chapter, we denote all scalars by lowercase letters x, vectors by lowercase bold letters $\boldsymbol{x}$, and matrices by bold uppercase letters $\boldsymbol{X}$. Coordinate frames are denoted by non-bold uppercase letters X. Estimated values are written as $\hat{x}$, and raw measurements as $\tilde{x}$.

As shown in Fig. 2.2, the body frame B (green) is defined according to the pose of the propellers. The plane defined by the x-axis and y-axis of B is orthogonal to the propellers' rotation axis. The IMU is located at the origin of B. IMU measurements are required to be expressed in B before being used by the estimator. The world frame W (black) is stationary. Its z-axis z_W has the same direction as gravity. The heading frame H (red) is defined by rotating the world frame around the z-axis by the yaw angle. Thus z_H always points to the direction of gravity. The heading frame's origin coincides with the origin of the camera frame C. The Euler angles roll ϕ, pitch θ, and yaw ψ reflect the rotation

between W and B. The rotation sequence is first z-axis (ψ), then y-axis (θ), and last x-axis (ϕ). ϕ and θ reflect the attitude of B respect to the gravity diretion.

If the number of tracked feature points is below the preset threshold, or there are not enough inlier points in the RANSAC-based calculation of the essential matrix, the current frame is defined as the new keyframe. A certain number of feature points are detected in a new keyframe. Unit vector $\boldsymbol{t}_{key}$ points from the camera position of the keyframe (right) to the current camera position (left), as shown in Fig. 2.2. The solid lines connect the camera with the feature points detected in the keyframe. The dashed lines connect the camera with points tracked in the current frame.

For cameras mounted far from the body center, we consider the translational velocity of the camera caused by its rotation relative to IMU, which is neglected in [9, 17]. The vector that points from the IMU to the camera expressed in the body frame is denoted by $\boldsymbol{p}_B^C$.

2.2.2. LINEAR DRAG MODEL

There are various aerodynamic forces acting on an multirotor MAV in flight. In [17], the thrust produced is modelled to be proportional to the sum of the squares of propeller speed. The drag acting on propellers is proportional to the product of the MAV's velocity and the sum of the rotor speeds. The velocity-square-proportional parasitic drag on the airframe was considered in [19] to improve the accuracy of the model in high-speed flight.

With the sensor setup that only consists of an IMU and a camera, we choose the simplified velocity-proportional drag force model [15]. We define the propeller plane as the plane orthogonal to the shafts of propellers. The drag force vector inside the propeller plane is approximately proportional to the projection of the MAV's velocity vector to the propeller plane. The drag parameter k_d is estimated as an EKF state to compensate for its variation in different aerodynamic regimes. Its derivative is modeled as a small Gaussian white noise $\boldsymbol{w}_{k_d}$.

The biased and noisy IMU measurements are modeled as

$$\tilde{\boldsymbol{a}} = \hat{\boldsymbol{a}} + \boldsymbol{b}_a + \boldsymbol{w}_a, \quad \tilde{\boldsymbol{\omega}} = \hat{\boldsymbol{\omega}} + \boldsymbol{b}_g + \boldsymbol{w}_g, \tag{2.1}$$

where $\boldsymbol{w}_a$ and $\boldsymbol{w}_g$ are white Gaussian noise. We model the derivatives of the additive accelerometer bias $\boldsymbol{b}_a$ and gyroscope bias $\boldsymbol{b}_g$ as small white Gaussian noise $\boldsymbol{w}_{\boldsymbol{b}_a}$ and $\boldsymbol{w}_{\boldsymbol{b}_g}$, respectively. $\hat{\boldsymbol{a}}$ denotes the translational acceleration caused by the aerodynamic force acting on a flying MAV, mainly consisting of the drag force and the thrust acting on the propellers. Hence

$$\hat{\boldsymbol{a}} = \left[a_x, a_y, a_z\right]^T = [k_d v_{B,x} + w_{v,x}, \quad k_d v_{B,y} + w_{v,y}, \quad \tilde{a}_z - b_{a,z} + w_{a,z}]^T, \tag{2.2}$$

where $\boldsymbol{w}_{\boldsymbol{v}}$ denotes the noise of the linear drag model.

2.2.3. STATE PROPAGATION

The estimator back-end is a simplified variant of the EKF-based back-end of the robocentric VIO [20]. It estimates the relative pose between the current body frame and the local frame of reference. The EKF state vector is defined as

$$\boldsymbol{x} := [{}^{R_k}\boldsymbol{p}_G, \quad {}^{R_k}_{G}\boldsymbol{q}, \quad \boldsymbol{g}_{R_k}, {}^{B_t}\boldsymbol{p}_{R_k}, \quad {}^{B_t}_{R_k}\boldsymbol{q}, \quad \boldsymbol{v}_{B_t}, \quad \boldsymbol{b}_a, \quad \boldsymbol{b}_g, \quad k_d]. \tag{2.3}$$

B_t is the current body frame at time t. When a new keyframe is defined, the new reference frame R_{k+1} is set to be the same as the B_t at the time. R_k is the current reference frame. It is the kth reference frame since the estimator is initialized. G stands for the global frame. It is the first reference frame R_0, *i.e.*, the body frame when the first keyframe is captured after the initialization of the estimator. Note that G and W are not the same coordinate frame. They have the same origin point, but there is relative rotation between them. $\boldsymbol{R}({}^{G}_{W}\boldsymbol{q})$ can be calculated from $\boldsymbol{R}({}^{G}_{W}\boldsymbol{q}) \cdot \boldsymbol{R}({}^{R_k}_{G}\boldsymbol{q}) \cdot \boldsymbol{g}_{R_k} = [0,0,g]^T$. ${}^{R_k}\boldsymbol{p}_G$ is a translation vector pointing from the origin of G to the origin of R_k, expressed in R_k. It is about the global position of R_k. ${}^{B_t}\boldsymbol{p}_{R_k}$ is a translation vector pointing from the origin of R_k to the origin of B_t, expressed in B_t. It is about the local position of B_t relative to R_k. ${}^{R_k}_{G}\boldsymbol{q}$ is the Hamilton quaternion reflecting the relative rotation between G and R_k. ${}^{B_t}_{R_k}\boldsymbol{q}$ reflects the relative rotation between R_k and B_t. $\boldsymbol{g}_{R_k}$ indicates the gravity vector expressed in R_k. $\boldsymbol{v}_{B_t}$ is the translational velocity of the IMU expressed in B_t.

Eq. (2.4) shows the IMU-driven state dynamics ($\dot{\boldsymbol{x}}$). $\boldsymbol{w_p}$ is the process noise in position integration. $[\hat{\boldsymbol{\omega}}]_\times$ represents the skew-symmetric matrix associated with $\hat{\boldsymbol{\omega}}$. $\boldsymbol{R}({}^{B_t}_{R_k}\boldsymbol{q})$ is a transformation function from ${}^{B_t}_{R_k}\boldsymbol{q}$ to *SO3* rotation matrix that maps a vector expressed in B_t to its expression in R_k. $\otimes$ denotes quaternion product. $\boldsymbol{E}_z$ is a 3×3 diagonal matrix with $[0,0,1]$ as its diagonal elements. $\boldsymbol{E}_{x,y}$ is a 3×3 diagonal matrix with $[1,1,0]$ as its diagonal elements. We utilize the techniques introduced in [21] for quaternion-related calculation.

$$
\begin{aligned}
{}^{B_t}\dot{\boldsymbol{p}}_{R_k} &= -[\hat{\boldsymbol{\omega}}]_\times \cdot {}^{B_t}\boldsymbol{p}_{R_k} + \boldsymbol{v}_{B_t} + \boldsymbol{w_p}, \\
\dot{\boldsymbol{v}}_{B_t} &= -[\hat{\boldsymbol{\omega}}]_\times \cdot \boldsymbol{v}_{B_t} + \boldsymbol{R}({}^{B_t}_{R_k}\boldsymbol{q})^T \cdot \boldsymbol{g}_{R_k} + \boldsymbol{E}_{x,y}(k_d \boldsymbol{v}_{B_t} + \boldsymbol{w_v}) + \boldsymbol{E}_z \hat{\boldsymbol{a}}, \\
{}^{B_t}_{R_k}\dot{\boldsymbol{q}} &= \frac{1}{2}\,{}^{B_t}_{R_k}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \hat{\boldsymbol{\omega}} \end{bmatrix}, \\
\dot{\boldsymbol{b}}_a &= \boldsymbol{w_{b_a}}, \quad \dot{\boldsymbol{b}}_g = \boldsymbol{w_{b_g}}.
\end{aligned}
\tag{2.4}
$$

2.2.4. ACCELERATION MEASUREMENT UPDATE

Drag-induced acceleration in the horizontal body plane (the plane that contains the x-axis and y-axis of B) can be measured by accelerometer measurements along the x-axis and y-axis of B. Accelerometer measurements correct state propagation through the following measurement update equations of the EKF.

$$
\begin{aligned}
z_{a,x} &= k_d\, v_{B,x} + b_{a,x} + w_{a,x}, \\
z_{a,y} &= k_d\, v_{B,y} + b_{a,y} + w_{a,y}.
\end{aligned}
\tag{2.5}
$$

The accelerometer measurement update steps along with the model-aided inertial ego-motion state propagation. The estimated horizontal states ($\boldsymbol{g}_{R_k}$, $v_{B,x}$, and $v_{B,y}$) do not drift over time. But they are noisy and negatively affected by the accelerometer bias $\boldsymbol{b}_a$. The performance of model-aided inertial ego-motion estimation is omitted in this chapter. An interested reader can refer to [15]. We use the drift-free roll $\hat{\phi}$ and pitch $\hat{\theta}$ angles to simplify the visual measurement processing, as introduced in Section 2.3. $\hat{\phi}$ and $\hat{\theta}$ of the current frame are calculated from $\boldsymbol{R}({}^{B_t}_{R_k}\boldsymbol{q})^T \cdot \boldsymbol{g}_{R_k} = R(\hat{\phi}) \cdot R(\hat{\theta}) \cdot [0,0,g]^T$.

2.2.5. RELATIVE VISUAL MEASUREMENT UPDATE

The *Visual Processing* module outputs the visual measurements of the relative pose with respect to the keyframe, $\hat{\psi}_{key}$ and ${}^{H,key}\hat{\boldsymbol{t}}$, to be introduced in Section 2.3. $\hat{\psi}_{key}$ reflects the visual measurement of the 1-dimension rotation between the current heading frame and the keyframe heading frame. ${}^{H,key}\hat{\boldsymbol{t}}$ is the visual measurement of the camera translation direction expressed in the keyframe heading frame. The EKF uses these measurements in the visual measurement update through Eq. (2.6).

$$\begin{aligned} \boldsymbol{z_r} &= {}^{B_t}_{R_k}\boldsymbol{q} + \boldsymbol{w_r}, \\ \boldsymbol{z_t} &= \frac{\boldsymbol{t}_{BC} + {}^{B_t}\boldsymbol{p}_{R_k} - \boldsymbol{R}({}^{B_t}_{R_k}\boldsymbol{q})^T \cdot \boldsymbol{t}_{BC}}{\|\boldsymbol{t}_{BC} + {}^{B_t}\boldsymbol{p}_{R_k} - \boldsymbol{R}({}^{B_t}_{R_k}\boldsymbol{q})^T \cdot \boldsymbol{t}_{BC}\|} + \boldsymbol{w_t}. \end{aligned} \tag{2.6}$$

$\boldsymbol{z_r}$ is the rotation between the current body frame and the keyframe body frame. It is calculated from $\hat{\psi}_{key}$ together with the roll and pitch angles of the current frame $\hat{\phi}_{curr.}$, $\hat{\theta}_{curr.}$ and the keyframe $\hat{\phi}_{key}$, $\hat{\theta}_{key}$. $\boldsymbol{z_t}$ is the direction of the translational motion from the camera position at the keyframe to the current camera position, expressed in the current body frame. It is obtained by rotating ${}^{H,key}\hat{\boldsymbol{t}}$ to get its expression in the current body frame via $\hat{\phi}_{curr.}$, $\hat{\theta}_{curr.}$, and $\hat{\psi}_{key,post.}$. $\boldsymbol{z_t}$ is a unit vector, so the propagated camera translation is normalized accordingly. $\boldsymbol{t}_{BC}$ is the translation vector points from the IMU to the camera, expressed in B. In the implementation, we neglect the correlation of the elements in the three axes of the visual measurements. Thus the measurement noise matrices $\boldsymbol{R}_{\boldsymbol{w_t}}$ and $\boldsymbol{R}_{\boldsymbol{w_r}}$ are diagonal matrices.

The rotation update using $\boldsymbol{z_r}$ takes place first. And then, translation update uses $\boldsymbol{z_t}$. The purpose of this two-step updating is to benefit the calculation of the visual translation measurement ${}^{H,key}\hat{\boldsymbol{t}}$ by the more accurate *a posteriori* relative yaw angle $\hat{\psi}_{key,post.}$, as introduced in Section 2.3. $\hat{\psi}_{key,post.}$ is calculated from the *a posteriori* ${}^{R_k}_{G}\hat{\boldsymbol{q}}$ and $\hat{\boldsymbol{g}}_{R_k}$ that has been updated by $\boldsymbol{z_r}$.

2.2.6. COMPOSITION AND RESETTING FOR NEW KEYFRAME

When a new keyframe is defined, the *a posterior* relative pose estimation ${}^{B_t}\hat{\boldsymbol{p}}_{R_k}$ and ${}^{B_t}_{R_k}\hat{\boldsymbol{q}}$ are composed to the global pose, as shown in Eq. (2.7).

$$\begin{aligned} {}^{R_{k+1}}_{G}\boldsymbol{q} &= {}^{B_t}_{R_k}\hat{\boldsymbol{q}} \otimes {}^{R_k}_{G}\boldsymbol{q}, \\ {}^{R_{k+1}}\boldsymbol{p}_G &= \boldsymbol{R}({}^{B_t}_{R_k}\hat{\boldsymbol{q}})^T \cdot {}^{R_k}\boldsymbol{p}_G + {}^{B_t}\hat{\boldsymbol{p}}_{R_k} \end{aligned} \tag{2.7}$$

The current body frame B_t becomes the new reference frame R_{k+1}. The expression of the gravity vector $\boldsymbol{g}_{R_{k+1}}$ in R_{k+1} is calculated as $\boldsymbol{g}_{R_{k+1}} = \boldsymbol{R}({}^{B_t}_{R_k}\hat{\boldsymbol{q}})^T \cdot \hat{\boldsymbol{g}}_{R_k}$. ${}^{B_t}\boldsymbol{p}_{R_{k+1}}$ and ${}^{B_t}_{R_{k+1}}\boldsymbol{q}$ are set to a zero vector and a unit quaternion whose vector part is a zero vector, respectively. Their corresponding elements in the covariance matrix are set to zeros too.

2.3. VISUAL RELATIVE POSE ESTIMATION

In this section, we describe the proposed method for estimating the yaw angle and the direction of translation relative to the keyframe. The relative yaw and translation are calculated separately, as shown in Fig. 2.1.

2.3.1. KEYFRAME-BASED FEATURE TRACKING

We follow the same idea of keyframe-based feature detection and tracking strategy as [9, 17]. Kanade-Lucas-Tomasi (KLT) [22] is utilized as the feature tracker to continuously track FAST features [23] between frames. If the number of points tracked in the current frame falls below a threshold, the current frame is defined as the new keyframe. A fixed number of uniformly distributed FAST points are detected in a new keyframe by evenly splitting the image into several regions while keeping the same number of good features in each region. These newly detected points are then added to a database for tracking in the coming frames.

2.3.2. LINEAR RELATIVE YAW CALCULATION

Based on the epipolar geometry of a pair of corresponding points, the linear 8-point algorithm [24] calculates the essential matrix which helps extracting the 3D relative rotation and the 2D direction of translation. If two relative orientation angles are known, the remaining angle and the direction of translation can be calculated linearly with 5-point pairs [25]. Slightly different from [17], our linear 5-point algorithm projects point coordinates on a unit sphere in the heading frame H.

The pixel location of a feature point is first undistorted and normalized using the distortion parameters and the intrinsic matrix of the camera. We then obtain the homogeneous coordinates of the feature points expressed in the camera frame, ${}^{C}\tilde{\boldsymbol{p}}_{key}$ and ${}^{C}\tilde{\boldsymbol{p}}_{curr.}$. The epipolar constraint is given by

$$ {}^{C}\tilde{\boldsymbol{p}}_{curr.}^{T} \lfloor {}^{C,curr.}\boldsymbol{t}_{\times} \rfloor \boldsymbol{R}_{C,key}^{C,curr.} {}^{C}\tilde{\boldsymbol{p}}_{key} = 0 \tag{2.8} $$

where $\lfloor {}^{C,curr.}\boldsymbol{t}_{\times} \rfloor$ is the skew-symmetric matrix of the translation vector ${}^{C,curr.}\boldsymbol{t}_{key}$ that points from the origin of the current camera frame to the origin of the keyframe camera frame, expressed in current camera frame.

Using the camera extrinsic rotation matrix $\boldsymbol{R}_{C}^{B}$ and the estimated roll $\hat{\phi}$ and pitch $\hat{\theta}$ angles of the keyframe and the current frame, we can express the feature point coordinates in the heading frame H, as

$$ \begin{aligned} {}^{H}\hat{\boldsymbol{p}}_{key} &= \boldsymbol{R}_{\hat{\theta},key}^{T} \boldsymbol{R}_{\hat{\phi},key}^{T} \boldsymbol{R}_{C}^{B} {}^{C}\tilde{\boldsymbol{p}}_{key}, \\ {}^{H}\hat{\boldsymbol{p}}_{curr.} &= \boldsymbol{R}_{\hat{\theta},curr.}^{T} \boldsymbol{R}_{\hat{\phi},curr.}^{T} \boldsymbol{R}_{C}^{B} {}^{C}\tilde{\boldsymbol{p}}_{curr.}. \end{aligned} \tag{2.9} $$

${}^{H}\hat{\boldsymbol{p}}_{key}$ and ${}^{H}\hat{\boldsymbol{p}}_{curr.}$ are normalized to unit vectors ${}^{H}\overline{\boldsymbol{p}}_{key}$ and ${}^{H}\overline{\boldsymbol{p}}_{curr.}$, which are equivalent to the 3D coordinates of the feature points projected onto the unit sphere. They describe the directions of the feature points in the heading frame H, so we refer to them as *point vectors*. The epipolar constraint equation in H is given by

$$ {}^{H}\overline{\boldsymbol{p}}_{curr.}^{T} \lfloor {}^{H,curr.}\boldsymbol{t}_{\times} \rfloor \boldsymbol{R}_{H,key}^{H,curr.} {}^{H}\overline{\boldsymbol{p}}_{key} = 0 \tag{2.10} $$

where $\boldsymbol{R}_{H,key}^{H,curr.}$ is the relative rotation between two heading frames, which equals to $\boldsymbol{R}_{\psi_{key}}$. In Eq. (2.10), the essential matrix $\boldsymbol{E}_{H} = \lfloor {}^{H,curr.}\boldsymbol{t}_{\times} \rfloor \boldsymbol{R}_{\psi_{key}}$ has six entries defined up-to-scale, which can be linearly solved with a minimum of five points [25]. The visual measurement of relative yaw $\hat{\psi}_{key}$ is then calculated from $\boldsymbol{E}_{H}$.

Both [9] and [17] reject outliers in the tracked points using the prior relative pose based on state propagation. They apply a threshold on the 2D distance from each tracked point to the epipolar line or the residuals of the epipolar constraint equation. This scheme is faster than iterative random sample consensus (RANSAC) outlier rejection. However, aggressive maneuvers may cause fewer feature points to be detected and correctly tracked. Due to the biases of the IMU measurements and the simplified drag model, the propagated relative translational motion would diverge if without enough vision updates. Consequently, the prior relative pose is not accurate enough for outlier rejection. Therefore, we employ RANSAC in the linear 5-point algorithm to calculate the relative yaw angle. Setting the number of random trials to a reasonably small number limits the required computation cost.

2.3.3. LINEAR RELATIVE TRANSLATION DIRECTION CALCULATION

During implementing the 5-point algorithm and testing it on EuRoC, it was observed that the translation direction vectors calculated together with relative yaw were noisy. The standard deviation of the direction error can be more than 30 degrees compared to the ground-truth motion direction. Instead of using this translational direction in the EKF update, we take the relative yaw angle as a known value and re-calculate the translation vector ${}^{H,key}\hat{\boldsymbol{t}}$. As introduced before, the visual translation measurement update happens after the rotation update. So we can calculate ${}^{H,key}\hat{\boldsymbol{t}}$ after the rotation update and make use of the updated *a posteriori* relative yaw angle $\hat{\psi}_{key,post.}$. This is achieved by rotating ${}^{H}\overline{\boldsymbol{p}}_{curr.}$ from the current heading frame to keyframe heading frame by $\boldsymbol{R}(\hat{\psi}_{key,post.})$. Then we get the epipolar constraint equation in the keyframe heading frame as

$$ {}^{H,key}\overline{\boldsymbol{p}}_{curr.}^{T} \lfloor {}^{H,key}\boldsymbol{t}_{\times} \rfloor \, {}^{H,key}\overline{\boldsymbol{p}}_{key} = 0. \tag{2.11} $$

There are only three up-to-scale entries in the translation-only essential matrix $\boldsymbol{E}_{H,key} = \lfloor {}^{H,key}\boldsymbol{t}_{\times} \rfloor$ in Eq. (2.11). Therefore, ${}^{H,key}\hat{\boldsymbol{t}}$ can be solved by a minimum of two point correspondences, as in [26].

We solve ${}^{H,key}\hat{\boldsymbol{t}}$ from the inlier point pairs of the 5-point RANSAC. In order to calculate translation, corresponding point vector pairs with big enough angles are necessary. After rotating the inlier point vectors into the keyframe heading frame, the point vector pairs are checked. Only pairs with angles beyond a certain threshold are used. In our implementation, the threshold is 5 degrees. Note that Eq. (2.11) has two mirrored solutions. The true direction can be determined by triangulating features and choosing the solution with all the features in front of the camera. To be more efficient, we use the propagated ${}^{B_t}\boldsymbol{p}_{R_k}$ to determine the direction of ${}^{H,key}\hat{\boldsymbol{t}}$. The solution that has a smaller angle with the expression of ${}^{B_t}\boldsymbol{p}_{R_k}$ in the heading frame is selected.

2.4. EXPERIMENTAL RESULTS

We evaluate the proposed ego-motion estimator on the widely recognized EuRoC MAV dataset. This dataset is comprised of multiple sequences with varying lighting conditions, and a maximum flight speed of 2.3 m/s. The proposed approach uses synchronized left-camera images and IMU measurements.

2.4.1. DATA PRE-PROCESSING

In the EuRoC dataset, the IMU does not coincide with the forward-right-down body frame definition. As a result, the accelerometer's x-axis and y-axis do not directly measure the drag force in the body horizontal plane. A body-IMU rotation matrix is needed to transfer the accelerometer's measurements to the body frame. We assume that the coordinate frame of the reflective markers mounted on the MAV approximately coincides with the body frame. Because the origin of our body frame is the same as the IMU frame, we use the sensor rotational extrinsic data to rotate IMU measurements and ground truth to the body frame.

The proposed estimator is initialized just before taking off to comply with the drag force model, which only stands when the MAV is in flight. Visual updating starts after the second keyframe is captured. The initial value of k_d was set to -0.2 for all the sequences. This value is close to the least-square solution calculated by the ground-truth velocity of the *Machine Hall 05* (*MH05*) sequence.

2.4.2. RESULTS AND DISCUSSION

The main evaluation results of the proposed estimator in terms of time efficiency and accuracy are listed in Table 2.1. The C++ code of this work is implemented based on the open-sourced code of [27]. The proposed estimator is compared with a state-of-the-art (SOTA) VIO MSCKF [5] implemented by the same open-sourced repository [27]. The top group of Table 2.1 (from ① to ③) shows the test results of the proposed estimator with three different settings in the vision front-end. The bottom group (④ and ⑤)) shows the outputs of MSCKF in two settings. Here we only compare with MSCKF since the gap in accuracy is big. The comparison of MSCKF with other VIO solutions can be found in [10].

The time consumption measurements shown in the second and the third columns of Table 2.1 were collected during the tests on *Machine Hall 05* sequence of the EuRoC dataset, running on a laptop computer. The root-mean-square error (RMSE) of absolute translation errors (ATEs) is the metric of estimation accuracy. The calculation is conducted by [28]. The alignment of the estimated trajectory and the ground-truth trajectory has 4 degrees of freedom (yaw and 3-d translation). The RMSEs of ATEs are shown in the eleven columns from the right of Table 2.1. The eleven columns correspond to the eleven flight sequences of the EuRoC dataset (*Vicon Room 101 to 103, 201 to 203*, and *Machine Hall 01 to 05*). The sequence names are abbreviated in Table 2.1.

For ① and ②, 30 feature points are detected in a keyframe, and a new keyframe is defined when the number of inlier points is below 10. For ③, 250 feature points are detected in a keyframe, and a new keyframe is defined when there are fewer than 60 inliers. ① corresponds to the estimator that uses images that are downsampled to half of the original resolution. And it does not have histogram equalization as image preprocessing. The rest settings (② and ③) and MSCKF (④ and ⑤) use images with original resolution and preprocessed by histogram equalization. Thus the vision processing time consumption of ① is much lower than ② while sacrificing the accuracy. It is an extreme case that minimizes visual processing. In the 5-point RANSAC of ①, ②, and ③, six point pairs are selected as a sample. The total number of random trials is set to six. If the highest inlier rate is more than 60%, this sample's inlier point pairs are used to calculate the essential matrix.

Table 2.1: Accuracy and time efficiency of the proposed method (top group, rows 1 to 3) compared with the baseline method (bottom group, rows 4 and 5). The name of the EuRoC dataset flight sequences (*Vicon Room 101 to 103, 201 to 203*, and *Machine Hall 01 to 05*) are abbreviated to fit the page width. "V" stands for *Vicon Room*, and "MH" stands for *Machine Hall*. Data below the sequence names shows the root-mean-square errors (RMSE) of the absolute translation errors (ATEs) of the estimated trajectories. **Bold** represents the overall best and underline represents the best of the proposed method.

ID	ATT[1]	AVT[2]	NFP[3]	V11	V12	V13	V21	V22	V23	MH1	MH2	MH3	MH4	MH5
①	**4.41**	**3.11**	30	5.12	1.19	8.04	4.21	2.11	5.90	4.70	13.4	12.0	2.37	2.76
②	7.25	5.81	30	1.26	1.17	3.11	2.38	1.48	2.91	3.60	13.2	11.2	2.12	1.56
③	12.9	11.5	250	2.04	1.76	5.37	4.56	2.80	7.76	13.5	1.67	**10.4**	2.37	1.35
④	7.93	3.82	51	0.16	0.13	0.12	245	0.13	**0.16**	38.0	5.20	130	2.35	**1.00**
⑤	17.4	7.58	199	**0.09**	**0.09**	**0.11**	**0.12**	**0.10**	0.20	**0.34**	**0.24**	58.5	**0.65**	1.54

[1] Average total time (ATT) consumption (millisecond) of processing after receiving a new frame.

[2] Average vision-related time (AVT) consumption (millisecond) in processing a single frame, including feature tracking, feature detection, and outlier rejection of feature points.

[3] Number of feature points (NFP). For the proposed approach (top group), the shown number is the number of feature points to detect in a new keyframe. For MSCKF (bottom group), the shown number is the average number of tracked points in each frame.

Comparing ② and ③, it is obvious that more feature points do not necessarily improve the accuracy of the proposed estimator. This is different from MSCKF. As shown by ④ and ⑤, ⑤ detects and tracks more points and has higher accuracy. The reason is that the vision update of MSCKF is triggered when a point loses tracking or leaves the field of view. Thus a bigger number of points means more times of updating. But for the proposed estimator, vision update happens for every frame and the visual measurement is calculated from all points tracked in a frame. In this case, fewer points do not necessarily lead to a less accurate essential matrix. On the contrary, a big number of tracked points with tracking noise would deteriorate the accuracy. So the proposed estimator is more suitable for applications where the computational power for visual processing is very limited and only a small number of points can be detected and tracked.

For ②, the estimated trajectories of *MH02* and *MH03* are less accurate than other sequences. The trajectory errors are mainly caused by the big estimation errors at the beginning of the sequences. For *MH02*, the ATE of the second half of the estimated trajectory is 3.243. For *MH03*, after deleting 20% of the estimated trajectory from the beginning of the sequence, the ATE of the rest part of the trajectory is 2.979. From Fig. 2.4, we can see that the estimation converges to be accurate after around 30 seconds since taking off. So in real-world applications, it is better to have safety measures for the possible big drifts at the beginning of a flight, for example, a space without obstacles close by.

From Table 2.1, we notice that, in the sequences *V201*, *MH01*, and *MH03*, MSCKF ④ drifts after very slow motion, which leads to big trajectory errors. But in general, the proposed approach is less accurate than MSCKF. Despite that, as shown in Fig. 2.3, Fig.

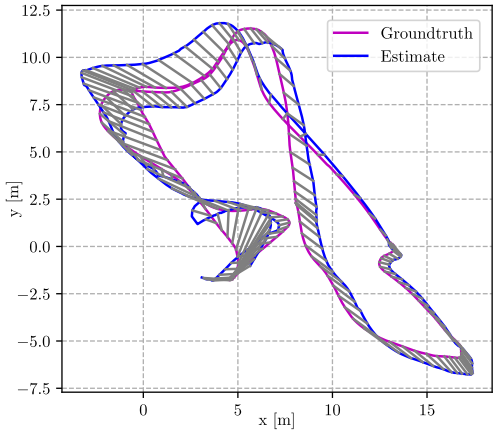
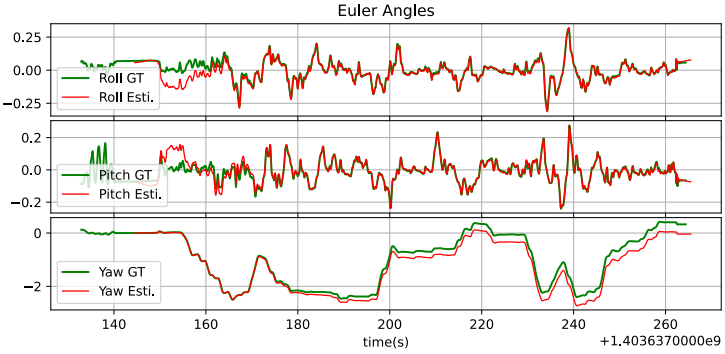
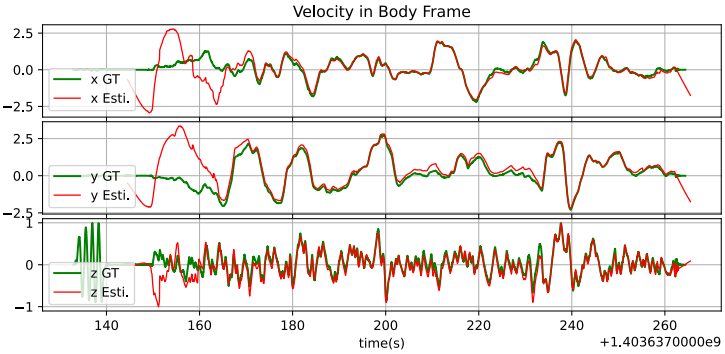


Figure 2.3: Estimated trajectory of *MH05* sequence by the proposed estimator ② in Table 2.1.

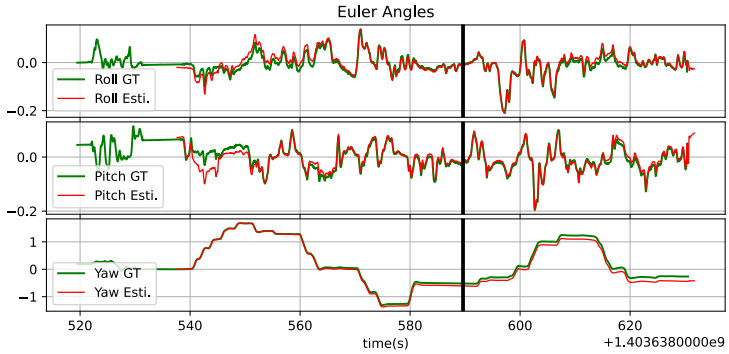


(a) Euler Angles Estimation (rad).

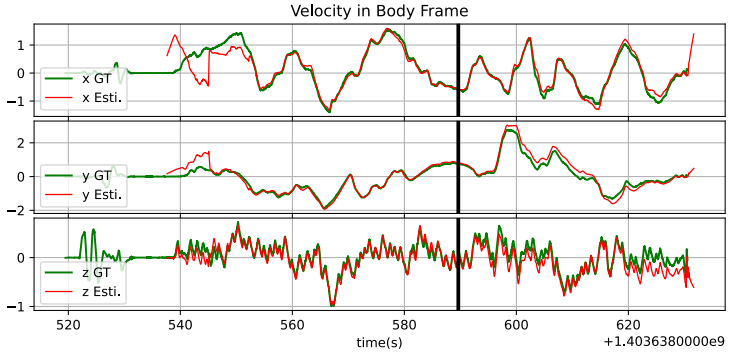


(b) Velocity Estimation Expressed in MAV Body Frame (m/s).

Figure 2.4: Estimated attitude and velocity of *MH03* sequence by the proposed estimator ① in Table 2.1.



(a) Euler Angles Estimation (rad).



(b) Velocity Estimation Expressed in MAV Body Frame (m/s).

Figure 2.5: Estimated attitude and velocity of *MH05* sequence by the proposed estimator ② in Table 2.1. Camera images are no longer used by the estimator after ~590 seconds.

2.4, and Fig. 2.5, the accuracy is acceptable for short-term ego-motion estimation.

The advantages of the proposed estimator are time efficiency and robustness. As Table 2.1 shows, ① and ② consume less time than MSCKF ④. Another fact worth mentioning is that we observed in experiments that our 5-point RANSAC implementation runs slower than the OpenCV 8-point RANSAC adopted by MSCKF. It can be attributed to the highly optimized OpenCV library function. There can be a space to further shorten the time consumption by optimizing our implementation. As for robustness, the proposed estimator does not have big drifts as ④ when testing on the sequences *V201*, *MH01*, and *MH03*. As the proposed estimator does not map the environment, sudden loss tracking of many or all points does not require re-initializing the system. In addition, the proposed estimator maintains accuracy even if there is not visual measurement anymore. As shown in Fig. 2.5, we cut off the camera image stream at the 52nd second after taking off, and the accuracy basically maintains. Obvious estimation errors are only observed in the velocity estimation of the z-axis in the last ~12 seconds of flight.

2.5. CONCLUSION AND FUTURE WORK

In this chapter, we present a visual-inertial ego-motion estimation approach for multirotor MAVs. Its high time efficiency derives from a vision frond-end that minimizes visual processing and a simple EKF-based back-end that performs loosely-coupled visual-inertial fusion. A linear drag model is utilized, and it leads to bounded errors in the estimation of the horizontal components of attitude and velocity. The estimation accuracy of the proposed approach is worse than the state-of-the-art approach in comparison but it can be enough for applications that require short-term ego-motion estimation. Given its high time efficiency, it is more friendly to lightweight multirotor MAVs with limited processing power onboard.

There is an opportunity to deploy the proposed ego-motion estimator in autonomous drone racing. Its small requirement for processing power benefits other algorithms running onboard and it has acceptable drifts in estimating flight trajectories. The racing gates can be detected and taken as stationary landmarks in ego-motion estimation and correct the accumulated error of the proposed estimator. The estimation accuracy rarely drops in short term when the vision information becomes absent, implying that the estimator can stay functional when there is no valid feature point due to the motion blur in agile maneuvers.

REFERENCES

[1] S. Li, E. van der Horst, P. Duernay, C. De Wagter, and G. C. H. E. de Croon, *Visual model-predictive localization for computationally efficient autonomous racing of a 72-gram drone,* (2019), arXiv: 1905.10110.

[2] K. N. McGuire, C. De Wagter, K. Tuyls, H. J. Kappen, and G. C. H. E. de Croon, *Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment,* Science Robotics **4**, 1 (2019).

[3] G. C. H. E. de Croon, *Monocular distance estimation with optical flow maneuvers and efference copies: A stability-based strategy,* Bioinspiration & Biomimetics **11**, 1 (2016).

[4] S. H. Lee and G. de Croon, *Stability-based scale estimation for monocular slam,* IEEE Robotics and Automation Letters **3**, 780 (2018).

[5] M. Li and A. I. Mourikis, *High-precision, consistent ekf-based visual-inertial odometry,* The International Journal of Robotics Research **32**, 690 (2013).

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

[7] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, *Keyframe-based visual-inertial SLAM using nonlinear optimization,* Proceedings of Robotis Science and Systems (2013).

[8] 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 **36**, 1053 (2017).

[9] D. Abeywardena, S. Huang, B. Barnes, G. Dissanayake, and S. Kodagoda, *Fast, on-board, model-aided visual-inertial odometry system for quadrotor micro aerial vehicles,* in *2016 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2016) pp. 1530–1537.

[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)* (IEEE, 2018) pp. 2502–2509.

[11] C. Forster, M. Pizzoli, and D. Scaramuzza, *Svo: Fast semi-direct monocular visual odometry,* in *2014 IEEE international conference on robotics and automation (ICRA)* (IEEE, 2014) pp. 15–22.

[12] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2013) pp. 3923–3929.

[13] S. Weiss and R. Siegwart, *Real-time metric state estimation for modular vision-inertial systems,* in *IEEE International Conference on Robotics and Automation* (2011) pp. 4531–4537.

[14] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, *Autonomous, vision-based flight and live dense 3D mapping with a quadrotor micro aerial vehicle,* Journal of Field Robotics **33**, 431 (2016).

[15] D. Abeywardena, S. Kodagoda, G. Dissanayake, and R. Munasinghe, *Improved state estimation in quadrotor mavs: A novel drift-free velocity estimator,* IEEE Robotics & Automation Magazine **20**, 32 (2013).

[16] M. Faessler, A. Franchi, and D. Scaramuzza, *Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories,* IEEE Robotics and Automation Letters **3**, 620 (2017).

[17] J. Svacha, G. Loianno, and V. Kumar, *Inertial yaw-independent velocity and attitude estimation for high-speed quadrotor flight,* IEEE Robotics and Automation Letters **4**, 1109 (2019).

[18] 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 **35**, 1157 (2016).

[19] M. Rigter, B. Morrell, R. G. Reid, G. B. Merewether, T. Tzanetos, V. Rajur, K. Wong, and L. H. Matthies, *An autonomous quadrotor system for robust high-speed flight through cluttered environments without GPS,* in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2019) pp. 5227–5234.

[20] Z. Huai and G. Huang, *Robocentric visual-inertial odometry,* The International Journal of Robotics Research **41**, 667 (2022).

[21] J. Sola, *Quaternion kinematics for the error-state kalman filter,* arXiv preprint arXiv:1711.02508 (2017).

[22] C. Tomasi and T. Kanade, *Detection and tracking of point features,* International Journal of Computer Vision Technical Report (1991).

[23] E. Rosten and T. Drummond, *Machine learning for high-speed corner detection,* in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9* (Springer, 2006) pp. 430–443.

[24] H. C. Longuet-Higgins, *A computer algorithm for reconstructing a scene from two projections,* Nature **293**, 133 (1981).

[25] F. Fraundorfer, P. Tanskanen, and M. Pollefeys, *A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles,* in *European Conference on Computer Vision* (Springer, 2010) pp. 269–282.

[26] C. Troiani, A. Martinelli, C. Laugier, and D. Scaramuzza, *2-point-based outlier rejection for camera-imu systems with applications to micro aerial vehicles,* in *2014 IEEE international conference on robotics and automation (ICRA)* (IEEE, 2014) pp. 5530–5536.

[27] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, *Openvins: A research platform for visual-inertial estimation,* in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2020) pp. 4666–4672.

[28] Z. Zhang and D. Scaramuzza, *A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry,* in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2018) pp. 7244–7251.

3

CNN-BASED EGO-MOTION ESTIMATION FOR FAST MAV MANEUVERS

In the field of visual ego-motion estimation for Micro Air Vehicles (MAVs), fast maneuvers stay challenging mainly because of the big visual disparity and motion blur. In the pursuit of higher robustness, we study convolutional neural networks (CNNs) that predict the relative pose between subsequent images from a fast-moving monocular camera facing a planar scene. Aided by the Inertial Measurement Unit (IMU), we mainly focus on translational motion. The networks we study have similar small model sizes (around 1.35MB) and high inference speeds (around 10 milliseconds on a mobile GPU). Images for training and testing have realistic motion blur. Starting from a network framework that iteratively warps the first image to match the second with cascaded network blocks, we study different network architectures and training strategies. Simulated datasets and a self-collected MAV flight dataset are used for evaluation. The proposed setup shows better accuracy over existing networks and traditional feature-point-based methods during fast maneuvers. Moreover, self-supervised learning outperforms supervised learning. Videos and open-sourced code are available at `https://github.com/tudelft/PoseNet_Planar`.

Parts of this chapter [1] have been published in 2021 IEEE International Conference on Robotics and Automation (ICRA).

3.1. INTRODUCTION

Indoor flight of Micro Air Vehicles (MAVs) is an attractive but challenging task. Towards the goal of autonomy, robust state estimation is one of the most essential modules of the MAV's flight control system. A camera captures rich information in a big field of view. Since being small and power-efficient, it is an ideal onboard sensor [2]. Its combination with the high-sample-frequency IMU is not only suitable for environment perception but also for real-time ego-motion estimation. Visual [3, 4] and visual-inertial [5–9] odometry (VO/VIO) systems contribute to MAVs' autonomy in generic environments by achieving real-time efficiency on onboard processors with decent accuracy.

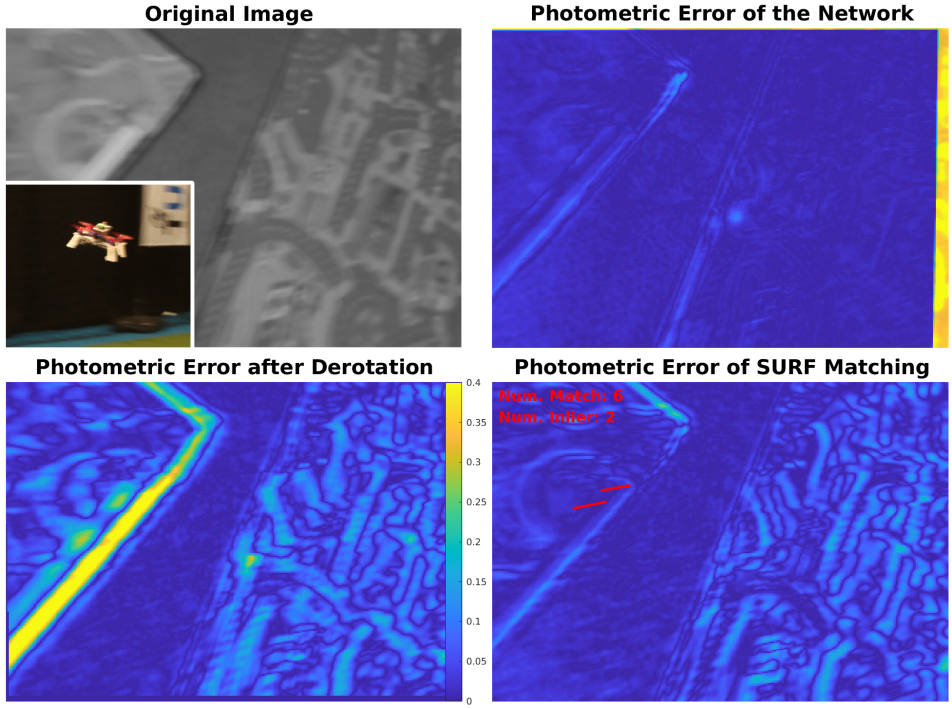


Figure 3.1: We study CNNs for visual ego-motion estimation, using blurry gray-scale images captured by the downward-facing camera of fast-moving MAVs. The derotated image pair has big photometric errors. After being warped by the network's prediction of the relative pose, only small photometric errors can be found around edges. The proposed networks better cope with fast motion than traditional feature-based methods.

Being constrained by the limited battery life, increasing flight speed is a direct way to enlarge an MAV's operation range and efficiency. However, it also introduces challenges for perception, and notably VO/VIO. Detection and tracking of handcrafted interest-point-based features [10–12] is the standard in state-of-the-art VIO systems [7–9]. However, such systems lack robustness in the presence of motion blur occurring during fast maneuvers. Robust Visual Inertial Odometry (ROVIO) [6] directly uses photometric errors of multilevel image patches around FAST feature points [11] to be more robust against image blur than

point features, partly because the texture of the tracked image patch is taken into account. However, Foehn *et al.* point out that, at larger speeds the state estimation of ROVIO suffers from drift [13]. When the speed gets larger, feature points and patches can move out of the camera's field of view sooner. We believe that the bigger visual disparities between images and the consequent lower number of frames in which features can be tracked is another adverse condition besides motion blur. Since ROVIO takes features' 3-dimensional (3-d) positions as states of an extended Kalman filter (EKF), having fewer visual observations decreases the accuracy. Other VIO systems such as [5, 7] that estimate feature positions by multiple observations can also suffer from high-speed motion [14].

CNNs are state of the art in many computer vision tasks and are promising for VO as well. Various networks have been proposed to estimate the pose change (rotation and translation) between two or more subsequent views. There are not only supervised pose networks trained by limited ground truth [15–17] but also self-supervised ones trained together with other networks including a depth estimation network [18–22]. Evaluated by the KITTI dataset [23], these networks obtain highly accurate performances and rival VO [4] with a traditional vision method [12].

There are also pose networks considering the application to an MAV's ego-motion estimation. For example, in [17], a recurrent CNN is trained on the EuRoC MAV dataset [24] to regress the 6 degree-of-freedom (6DoF) motion. The network manages to learn the more complex (compared with a car) MAV's dynamics but the accuracy is limited by the small amount of training data. Differently, PRGFlow [25] focuses on the essential function of estimating the 3-d translational velocity of the MAV with a downward-facing camera, assuming a planar ground. Aided by the attitude estimated by IMU measurements, via image warping, the task is simplified to the pixel-level similarity transformation estimation. Although PRGFlow thoroughly studied CNN-based ego-motion estimation, the focus is on the low-speed flight (about 0.5m/s on average), with motion blur lacking from the artificially generated training images.

Hence, it is currently still an open question of how good CNNs perform during fast maneuvers. To gain insight into this matter, in this chapter we study networks predicting 3-d relative translation of MAVs in fast maneuvers with a downward-facing camera. The networks are trained and tested on images with significant motion blur and big visual disparities. Our main contributions are that we: (1) Extend and further improve the performance of the network framework proposed in [25] to fit fast maneuvers, and (2) Investigate how well the networks can deal with faster motion in comparison with traditional feature-point-based methods. According to our knowledge, this is the first work showing networks' superior performance in fast motion when traditional feature-point-based methods have high failure rates.

3.2. METHODOLOGY

3.2.1. HOMOGRAPHY TRANSFORMATION

As shown in Eq. (3.1), for a fixed point laying on a plane observed by two cameras, it has been proven in [26] that the projective coordinates $\boldsymbol{x_1}$, $\boldsymbol{x}_2$ of the same point in the camera frames are related by the homography matrix $\boldsymbol{H}$ that depends only on the 6-d relative pose of the cameras and the unit normal vector of the plane $\boldsymbol{n}$. $\boldsymbol{R}$ denotes the rotation

matrix between the camera frames and $\boldsymbol{t}$ denotes the translation vector expressed in the second camera's frame pointing from the second camera to the first one. The scalar d is the distance from the first camera to the plane.

$$\boldsymbol{x}_2 = \boldsymbol{H}\boldsymbol{x}_1,\ \boldsymbol{H} = \boldsymbol{R} + \frac{\boldsymbol{t}\boldsymbol{n}^T}{d} \tag{3.1}$$

Here we define the coordinate system whose x-axis points to the north, y-axis to the east, and z-axis to the gravity direction as the world frame. The plane that the downward-facing camera observes is assumed to be orthogonal to the gravity vector. The attitude of the camera relative to the world frame can be estimated by an IMU, then the information remaining unknown in the homography matrix is the ratio of the translation vector $\boldsymbol{t}$ to the distance to the plane d. Here we refer to it as the distance-scaled relative translation vector. This vector together with the flight height that is available from a downward-facing rangefinder can determine the metric average translational velocity of the MAV during the camera's sample interval. Here we refer to it as the distance-scaled translational velocity vector.

PRGFlow warps both the images to make the image planes parallel to the ground using the absolute attitude estimated by the IMU. The distance-scaled relative translation then can be determined by the similarity transformation between the image pair. Networks are trained to predict the 3 parameters (2-d translation, zoom-in/out) reflecting the relative location of pixels. However, when the roll or pitch angle of the MAV is big, which is often the case in fast maneuvers, the camera would have big tilt angles relative to the plane's normal vector. So warping the image pair like PRGFlow can cause big black boundaries and thus lose many pixels. It then requires pre-processing moving the pixels back inside the image frame and the corresponding post-processing for calculating the pose from the similarity transformation.

To avoid the above-mentioned processings, our networks predict the distance-scaled relative translation vector expressed in the camera frame directly from images that have (non-zero) tilt angles, requiring input images to have the identical intrinsic parameters as the training set. Only one image needs to be warped by the relative rotation. Tilt angles are available from an IMU but we additionally explore networks predicting them in Subsection 3.3.4.

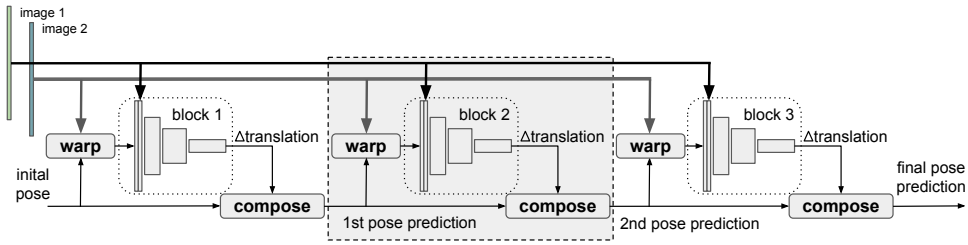


Figure 3.2: An ICSTN-based network with 3 blocks for relative pose prediction. The dashed line frame indicates the basic functional unit that can be sequentially stacked one or multiple times. The dotted frame indicates a network block that takes (downsampled) concatenated images as input and predicts the 3-d distance-scaled relative translation.

3.2.2. CASCADED NETWORK BLOCKS CONNECTED BY IMAGE WARPING

Sanket *et al.* adopt the inverse compositional spatial transformer networks (ICSTN) [27] as the framework of their networks [25]. The ICSTN has multiple network blocks that predict the image deformation that benefits the final goal. Based on the homography transformation, a new image can be synthesized by warping the original image using the method proposed in [28]. As shown in Fig. 3.2, a network block is made up of multiple convolutional layers followed by a fully-connected layer regressing the translation. With multiple cascaded network blocks, each block takes the concatenated original image 1 and the image 2 warped by the newest pose prediction as input and combine its output into the pose prediction. As the pose prediction is refined by more blocks, there is less relative motion between the concatenated images. Each block predicts a part of the total relative translation, making the problem more tractable. The network can also make use of an initial guess of the relative pose, which can be available from the IMU integration or the MAV's dynamic model. PRGFlow has compared network architectures inside one block. Focusing on fast maneuvers, we study higher-level architectures applying to the pyramidal images and feature maps to enlarge the receptive field which is important for dealing with the big visual disparities.

The loss functions of the networks are the mean of the Charbonnier [29] loss of the predicted 3-d translation's error in supervised learning and the mean of the Charbonnier loss of the valid pixels' photometric error in self-supervised learning. For data augmentation, we feed the network with image pairs concatenated in both orders to perform bidirectional training.

We implement the networks in Python 3.6.9 with the Pytorch [30] 1.1.0 library. The Adam optimizer [31] with $\beta = (0.9, 0.999)$ is utilized during the 25 training epochs. The batch size is 16. The initial learning rate is 0.0002 and it is divided by 2 after 5, 10, 15, and 20 epochs. The weights of convolutional layers are initialized by Glorot initialization [32] with a gain of 1. The weights of fully-connected layers are initialized by the (Pytorch default) uniform distribution $U(-\sqrt{k}, \sqrt{k})$ where k is the multiplicative inverse of the number of input features.

3.2.3. DATASET GENERATION

We use the Microsoft COCO dataset [33] as the source of a large variety of textures to generate a big number of image pairs thanks to the homography transformation. A source image is treated as a plane above which a simulated camera is moving. For one plane, one image pair is generated. Costante *et al.* and Kendall *et al.* test their pose estimation networks with respectively artificial Gaussian blur [15] and motion blur [34] added to images. Their blur is uniform over the whole image and thus not ego-motion-related. In order to obtain realistic blur that is caused by the camera's motion within the exposure duration, we simulate a moving camera whose time step for the kinetic integration is 0.1 millisecond (ms) and the exposure duration is 10ms. In each integration step during the exposure, an image is sampled from the homography transformation of the plane. The blurry image is the average of the 100 sampled images. The poses of both the exposure starting step and ending step of an image are recorded. Except for Subsection 3.3.3, the pose of the starting step is used as the ground truth in supervised learning. 30 frames per second (fps) are recoreded to simulate a common global shutter camera. The images are in grayscale

with the resolution of 320×224 pixels. The intrinsics of the simulated camera are $f_x = 160$, $f_y = 160$, $c_x = 160$, $c_y = 112$.

Figure 3.3: A blurry image pair. Starting from the left: (1) the first image, (2) the derotated second image, and (3) the second image.

The initial poses of the kinetic integrations uniformly distribute within the flight envelope of a normal quadrotor MAV. The uniformly randomly generated translational velocity and rotational velocity stay constant during the kinetic integration. The camera's distance-scaled translational velocity vector's components along the x-axis and y-axis of the world frame range from -7.5 to 7.5. The range of the component along the z-axis is from -3.75 to 3.75. Angular velocity vector's components along the x-axis and y-axis of the camera frame range from -180 to 180 degrees per second. The range of the component along the z-axis is from -90 to 90 degrees per second. Initial roll and pitch angles range from -25 to 25 degrees. Since we record the poses at the start and the end of exposure duration, the motion flow that causes blur can be calculated. Over the dataset, the average motion flow of all the pixels in an image has mean values of 6.6 and 6.2 pixels in the x-axis and y-axis, respectively. The maximum motion flow of all the pixels in an image has mean values of 13.4 (x-axis) and 11.9 (y-axis) pixels. The above data shows that our dataset involves a big range of motion and significant motion blur. After removing hundreds of images with little texture, there are 82,172 training samples, 9,948 validation samples (for validating the model after each epoch during training), and 30,565 testing samples.

Table 3.1: ICSTN-based networks with different numbers of blocks.

Network	Num. Blocks	Num. Conv./ Kernel/ Stride	Num. Params	FPS	RF	Inlier Rate(%)	EPE's Standard Deviations (**1e-3**) (end-point loss / multi-stage losses)	Medians of EPE's Absolute Values (**1e-3**) (end-point loss / multi-stage losses)
1 [18]	1	8/ 7,5/ 2.2	1.583M	215	263	90.14	(13.44,13.88,19.23)	(7.83,8.11,11.93)
2 [35]	1	18/ 3,3/ 2,2	1.441M	105	759	91.77	(6.70,6.94,9.63)	(3.85,4.03,6.12)
3 [36]	1	21/ 3,3/ 2,2	1.477M	103	975	92.35	(7.32,7.58,10.57)	(4.33,4.48,6.66)
4 (ours)	2	9/ 7,5/ 2,2	1.385M	104	647	89.92	(3.35,3.30,3.91) / (2.71,2.64,3.28)	(1.89,1.89,2.61) / (1.49,1.49,2.18)
5 (ours)	3	5/ 7,5/ 2,2	1.367M	101	71	87.59	(3.33,3.15,4.29) / (2.28,2.22,3.08)	(1.91,1.83,2.90) / (1.24,1.22,2.11)
6 (ours)	3	5/ 7,5/ 4,2	1.252M	101	135	92.18	**(2.06,2.05,2.90)**	**(1.16,1.13,2.00)**
7 (ours)	4	3/ 7,5/ 4,4	1.421M	95	55	85.48	(3.58,3.51,4.75) / (2.20,2.19,3.06)	(2.20,2.20,3.20) / (1.28,1.26,2.13)
8 (ours)	4	3/ 9,5/ 8,4	1.276M	95	105	87.35	(2.10,2.09,3.01)	(1.25,1.22,2.09)

Table 3.2: ICSTN-based networks using pyramidal images or feature maps.

Num. Pyramid	Num. Layers/ Kernel/ Stride	Num. Params	FPS (intrpl. / avg. pooling)	Receptive Field	EPE's Standard Deviations (**1e-3**) (intrpl. / avg. pooling)
3	3/ 7,5/ 2,2; 4/ 7,5/ 2,2; 5/ 7,5/ 2,2	1.388M	103 / 108	23×4; 39×2; 71	(2.13,2.05,2.93) / (2.11,2.09,2.91)
3	3/ 7,5/ 4,2; 4/ 7,5/ 4,2; 5/ 7,5/ 4,2	1.272M	104 / 111	39×4; 71×2; 135	(2.11,2.09,2.94) / (2.09,2.06,2.94)
3	4/ 7,5/ 2,2; 4/ 7,5/ 4,2; 4/ 7,5/ 4,4	1.316M	104 / 109	39×4; 71×2; 119	(2.07,2.06,2.93) / (2.07,2.07,2.90)
4	2/ 7,5/ 2,2; 2/ 7,5/ 4,2; 3/ 7,5/ 4,2; 3/ 7,5/ 4,4	1.386M	93 / 98	15×8; 23×4; 39×2; 55	(2.03,2.02,2.87) / (2.02,2.02,**2.85**)
3	[FPE: 3/ 7,5/ 2,2] + [2; 3; 4]	1.455M	100	71; 71; 71	(2.18,2.12,3.12)
3	[FPE: 3/ 7,5/ 4,2] + [2; 3; 4]	1.340M	100	135; 135; 135	(**2.00**,**1.97**,3.08)

Table 3.3: Comparison between supervised and self-supervised learning.

Networks	Inlier Rate(%) (supervised / self-supervised)	EPE's Standard Deviations (**1e-4**) (supervised / self-supervised)	Medians of EPE's Absolute Values (**1e-4**) (supervised / self-supervised)
Table 3.1 6	92.18 / 91.41	(20.62, 20.46, 28.99) / (19.16, 19.29, 28.36)	(11.62, 11.29, 20.01) / (10.71, 10.60, 19.54)
Table 3.1 6*	75.20 / 70.84	(5.30, 4.95, 7.04) / (4.16, 3.82, 5.54)	(3.15, 3.15, 4.55) / (2.40, 2.15, 3.47)
Table 3.2 3(p)*	73.20 / 69.91	(5.49, 4.93, 7.02) / (4.20, 3.79, 5.52)	(3.08, 2.91, 4.42) / (2.36, 2.16, 3.47)
Table 3.2 4(i)*	71.43 / 67.29	(4.49, 4.23, 5.94) / **(3.08, 2.89, 3.90)**	(2.82, 2.64, 3.93) / **(1.91, 1.80, 2.63)**

3.3. NETWORKS

3.3.1. ICSTN-BASED NETWORKS

In this subsection, we study ICSTN-based networks with different numbers of blocks. Blocks of one multi-block network have identical architecture. In the 3rd column of Table 3.1, "Num. Conv." is the abbreviation of the number of convolutional layers. The kernel sizes and the strides of the first and second convolutional layers are also listed. Deeper layers have kernel sizes of 3 and strides of 2 or 1. Networks' inference speeds are indicated by fps when running on an NVIDIA Jetson TX2 with Ubuntu 18.04.3 LTS and Cuda V10.0.326 in the MAXP_CORE_ARM power mode. "RF" denotes the receptive field of the fully-connected layer's input. We call the final prediction error the end-point error (EPE). The predicted translation is rotated into the world frame to calculate the 3-d EPE vector. 1e-3 means that the actual data equals the shown data multiplied by 10^{-3}.

EPE's standard deviation can reflect how noisy the predictions are but it is sensitive to outliers that can be caused by image pairs lacking texture or having duplicate textures. So we use a local outlier rejection function of MATLAB to remove outliers and keep the characteristic of local distribution. After respectively ascendingly sorted by the corresponded ground truth along each axis, the EPEs whose absolute values are more than 3 scaled median absolute deviations in a local window of size 1000 are rejected. A prediction is considered an outlier if its component along any axis is rejected. The 3 values inside the bracket separated by commas correspond to the data of the x-axis, y-axis, and z-axis. The medians of EPE's absolute values are calculated from all the predictions including outliers.

Networks with a single block are shown in the first 3 rows of Table 3.1. The 1st network is the pose estimation network proposed in [18]. The 2nd and 3rd networks respectively have skip connections [35, 36] for better performance in their deeper architectures. They have smaller model sizes, higher accuracy, but slower speed. The 2nd network with 18 convolutional layers and densely connected architecture has the highest accuracy. In the case of multiple blocks, for each block, there is an image warping operation that has unneglectable time consuming. Since a network's inference speed is required to be around 100fps, the total number of layers in the whole network decreases when the number of blocks increases. The accuracy gets worse when there are more than 3 blocks mainly because the blocks are too shallow and the total capacity of the whole network decreases. As shown by the 6th and 8th networks, bigger strides lead to smaller resolution of the last feature map and thus fewer parameters in the fully-connected layer. Besides, it increases the receptive field. Since our dataset has image pairs with big visual disparities, a bigger receptive field can capture more feature correspondences and improve the accuracy.

For the networks with multiple blocks, instead of only using the loss of the final prediction (end-point loss) in training [25], We used a weighted sum of the losses of every prediction after every block (multi-stage losses) for backpropagation. The loss weight distributions of blocks are respectively [0.3,0.7], [0.2,0.3,0.5], and [0.1,0.2,0.3,0.4] for networks with 2, 3, and 4 blocks. The accuracy is compared by the 4th, 5th, and 7th networks of Table 3.1. Multi-stage losses produce higher accuracy. For the 6th and 8th networks, we only show the results of multi-stage losses. All the networks in the rest part of this chapter are trained with their multi-stage losses.

3.3.2. PYRAMIDAL IMAGES AND FEATURE MAPS IN ICSTN

From Table 3.1, we find that a bigger receptive field can benefit accuracy. When the kernel size and stride keep the same, another way to increase the receptive field is using pyramidal images. Networks using pyramidal images or feature maps with lower resolution are shown in Table 3.2. The downsampled image at each pyramid level has half the size of the image of its adjacent lower level. So the lowest-resolution image of the network that has 4 pyramids has one-eighth the width and height of the original image. The number of network blocks is the same as the number of pyramids. The first pose prediction block uses the images at the highest pyramid level with the lowest resolution. The predicted pose is used to warp the original image. Then the warped image is downsampled to the next lower pyramid level and input to the next network block. For image downsampling, we compared bilinear image interpolation [28] and average pooling. They have similar accuracy, but average pooling is faster in our Pytorch implementation.

Since the EPE's standard deviation is enough to reflect the accuracy of the network predictions, the medians of EPE's absolute values are not shown in Table 3.2. Comparing the 1st network of Table 3.2 with the 5th of Table 3.1, with the same kernel size and stride, the pyramidal network that has fewer layers achieves higher inference speed and accuracy, thanks to the bigger receptive fields of the first two blocks. Comparing the 2nd network of Table 3.2 with the 6th network of Table 3.1, the pyramidal network has slightly lower accuracy. We think it is because when the receptive fields are big enough, the pyramidal version receives less information due to the downsampling. The 8th network of Table 3.1 has a big receptive field. Also with 4 blocks, the 4th network of Table 3.2 has decreasing receptive fields with the increasing of image resolution. Although 3 out of 4 blocks have smaller receptive fields than the 8th network of Table 3.1, this 4-stage coarse-to-fine refinement gets better accuracy. The 2nd and 3rd networks of Table 3.2 have the same total number of layers. The 3rd one having a deeper block at the lowest resolution achieves slightly higher accuracy.

The pyramidal feature maps network is based on the feature pyramid extractor (FPE) inspired by the PWC-Net [37]. The results are shown in the last 2 rows of Table 3.2. The general principle is extracting multiple feature maps at different resolutions (pyramid levels) of each image respectively by the same convolutional feature extractor network. One of the feature maps is warped and then concatenated along the channel dimension with the other feature map of the same size. The concatenated feature maps are the input of the pose prediction blocks. The networks we design have 3 levels of pyramidal feature maps and 3 pose prediction blocks that have 2, 3, and 4 convolutional layers respectively. The FPE network at the last row of Table 3.3 has the highest accuracy in the x-axis and y-axis.

3.3.3. SELF-SUPERVISED LEARNING

Self-supervised learning is based on the photometric error between the image warped by the predicted relative pose and the other image. We use a mask to not count the photometric errors of the pixels whose locations to interpolate lie outside the image frame. By the results shown in the 1st row of Table 3.3, we notice that self-supervised learning with a basic photometric loss gets better accuracy than supervised learning. The reason behind it is worth further studying. For now, we think it is mainly because the target relative poses used in supervised learning are calculated from the poses at the starting time points of the

image exposure. While the simulated camera keeps moving within the exposure duration, motion blur appears and the image gets a different appearance from the start of exposure, and thus there will be small photometric errors between the images warped by the target relative pose. This means the network is trained to regress to a target not perfectly matching the feature correspondences. This discrepancy can "confuse" the network. While in the case of self-supervised learning, the network tries to minimize the photometric error affected by the blur and is more likely to converge to the "accurate" relative pose that best matches the feature correspondence. When we evaluate the self-supervised network, we use the poses at the start of exposure as the ground truth, to which the network does not learn to converge. But the effect of it is smaller than the "confusion" induced by the discrepancy.

To verify the hypothesis above, we take the average of the poses at the start and at the end of exposure as reference pose for a blurry image and calculate the target relative pose from it. The results are marked with an asterisk and shown in Table 3.3 from the 2nd to the 4th row. "Table 3.2 3(p)" denotes the average pooling version of the 3rd network of Table 3.2. Similarly, the "(i)" denotes the bilinear interpolation version.

From the results of the testing set shown in Table 3.3, one can notice that all the self-supervised networks are more accurate. Besides, they are also slightly more accurate on the training set. As for the supervised networks, training with the new target pose (2nd row) has much higher accuracy compared with the old target pose calculated from the poses at the start of exposure (1st row). The inlier rates drop when we use the new target pose. The reason is that the errors of the image pairs having less texture are more likely to be outliers because their neighbors have smaller errors. Obviously, the new target pose matches the feature correspondence better and acts as better supervision. But still, the remaining small discrepancy makes it less good than self-supervised networks. So we believe that self-supervised learning is a better choice for blurry image pairs that have unknown relative pose perfectly matching the feature correspondences. This also provides us with the hypothesis that taking the non-neglectable exposure duration of an image into account may benefit ego-motion estimation.

3.3.4. NETWORKS FOR TILT ANGLE PREDICTION

It is known that one can estimate the tilt of the camera relative to the plane in the view from the optical flow field [38]. Since tilt is a property of the flow field and hence affects both images, it cannot be estimated iteratively by our ICSTN-based framework that warps only one image (Fig. 3.2). For this preliminary investigation, we employ a single deep network block to predict tilt angles from a pair of derotated images, supervised by ground truth. Shown in Table 3.4, the best network's EPE's standard deviation is around 4 degrees for both angles. Although the prediction is noisy, it may serve as an unbiased absolute information source of attitude.

3.4. EVALUATION

The 4th network of Table 3.3 is chosen for evaluation and comparison to traditional feature-based methods. Note that the network is trained only with the simulation dataset described in Subsection 3.2.3 without any fine-tuning to highlight the generalizability. We

Table 3.4: Networks predicting tilt angles.

Networks	Inlier Rate(%)	EPE's Std. Dev. (radian, **1e-2**)	Medians of EPE's ABS (radian, **1e-2**)
Table 3.1 - 2	97.13	**(7.72, 6.57)**	**(5.06, 4.30)**
Table 3.1 - 3	96.91	(8.71, 7.50)	(5.75, 4.87)

use MATLAB functions for traditional feature detection and matching. More feature points can be detected by tuning the parameters of the functions. Here we only show the results of the default parameters. 50 uniformly distributed ones are selected when there are enough detected features. The translation is obtained by calculating the similarity matrix (with known in-plane rotation, 3 DoF left) based on linear least squares and random sample consensus (RANSAC).

3.4.1. SIMULATED DATASET

We generate a dataset of 5000 image pairs with different exposure duration (ranges from 0.2ms to 20ms) and random distance-scaled velocity vectors that have the same norm ($\|\boldsymbol{v}\|/d = 5$) to compare the performance of the network and feature-based methods with increasing motion blur. Another dataset of 5000 sharp image pairs with different distance-scaled velocity vectors (same range as the training set) and the same exposure duration (0.2ms) is generated to study the effect of visual disparity. All the image pairs have the random attitude and zero angular rates.

The norms of the error vectors of the estimated distance-scaled translations and their local standard deviations are shown in Fig. 3.4. We use linear fitting to show their trends. The local standard deviations are calculated with the window size of 10% of the total number of inliers. For feature-based methods, if there are less than 2 inlier matchings in RANSAC, this pair is treated as an outlier. For the estimated pose, we apply the same local outlier rejection as Section 3.3 with the window size of 500. The final inlier rates of the network, SURF [39], ORB [12], and FAST [11] are shown in Fig. 3.4. The network has the highest inlier rates partly because it does not rely on the number of matches so it can perform prediction on every image pair. Fig. 3.4 shows that the network is most accurate with both datasets. Its performance is barely affected by the growing motion blur while feature-based methods more or less provide more noisy results. For the increasing disparity, the network is also least affected.

3.4.2. FLIGHT DATASET

To obtain sensor data in real-world flights, an MYNT EYE D1000-120 visual-inertial sensor is downward-facing mounted on an Eachine Wizard X220 FPV Racing Drone that carries an NVIDIA Jetson TX2. IMU measurements (200Hz) and monocular gray-scale images (30fps) with an exposure duration of 20ms are collected. The images are undistorted and transformed to have the same size and intrinsic matrix as the training set. The top left of Fig. 3.1 shows an example. The camera's attitude is estimated by the Madgwick filter [40] using the IMU measurements. The ground-truth velocity is obtained from the OptiTrack motion tracking system at 120Hz. The first column of Table 3.5 shows the average and

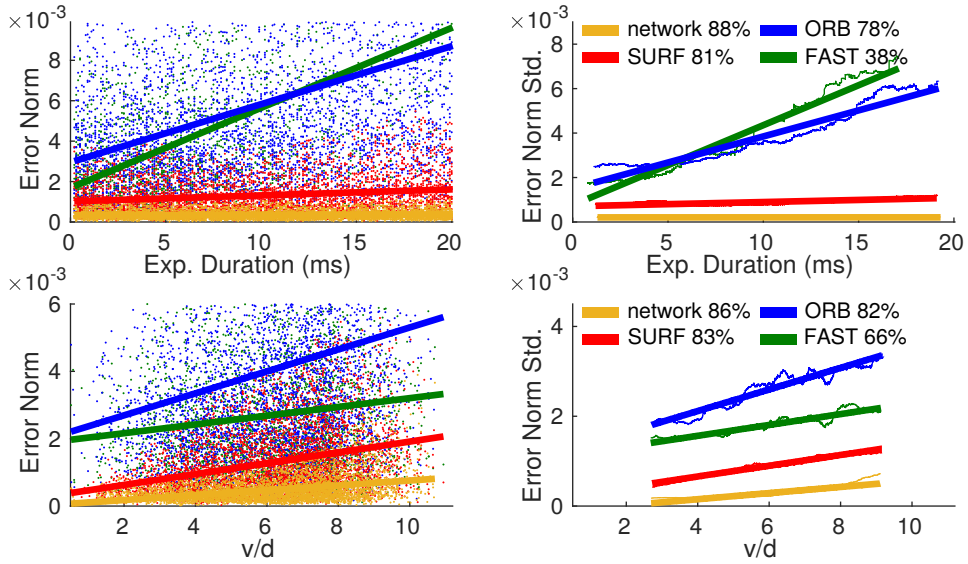


Figure 3.4: Comparison between feature-based methods and the network. The top row shows how their accuracy changes with the amount of motion blur. The bottom row shows the effects of the increasing visual disparity. "v/d" denotes the norm of the distance-scaled velocity of the simulated camera.

maximum distance-scaled translational velocity of 4 one-minute flights.

Table 3.5: Network and SURF evaluated by flight dataset.

Sequence	RMSE (**1e-2**): network / SURF (orig.) / SURF (hist. equal.)
1(0.3,1.3)	(2.10, 2.17, 2.01) / (1.95, 2.17, 2.40) / (**1.90**, **2.05**, **1.77**)
2(0.6,2.5)	(4.21, 4.57, 3.65) / (4.03, **4.35**, 3.70) / (**3.91**, 4.36, **2.96**)
3(1.2,3.2)	(5.44, 5.87, 5.03) / (5.75, 5.48, 6.36) / (**5.20**, **5.25**, **4.34**)
4(1.4,3.9)	(10.2, 9.48, 10.0) / (15.1, 11.5, 28.3) / (**10.0**, **8.91**, **8.65**)

The results of the fastest flight are shown in Fig. 3.5. SURF's result is noisy in some parts of the flight mainly because of the big motion blur and scenes lacking texture. For 8.6% of image pairs, SURF has less than 2 inlier matchings. We use zero vectors to show its results in this case. For the other 3 slower flights, SURF has enough matches all the time. The root mean square errors (RMSEs) of the distance-scaled velocity vector's components along the world frame's 3 axes are shown in Table 3.5. When using original images, the network outperforms SURF more in faster flights where fewer points are detected. In histogram equalized images, more SURF points are detected and the accuracy is slightly higher than the network. The network performs better on original images than histogram equalized images since the images in the training set are without pre-processing.

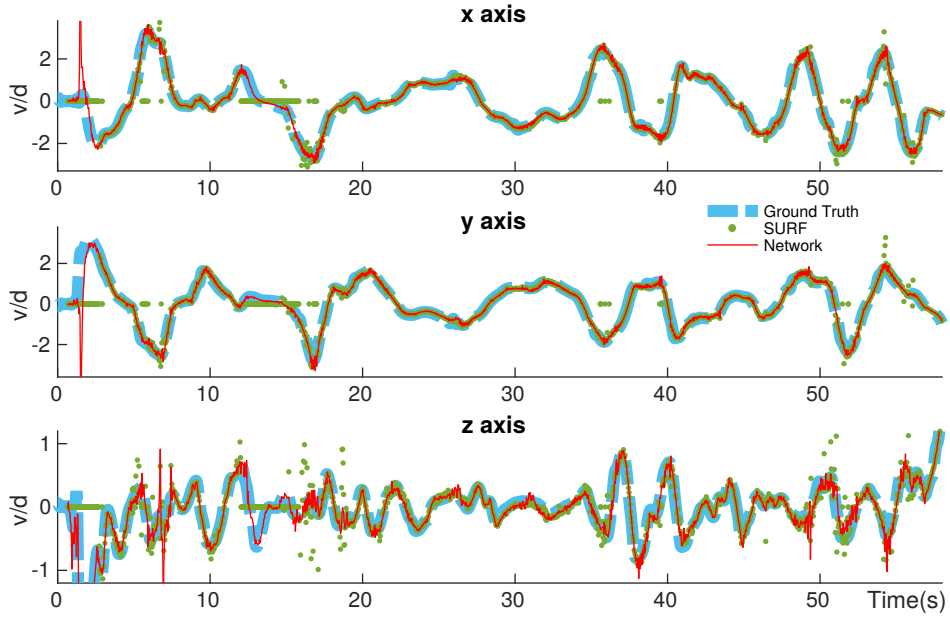


Figure 3.5: The ratio of velocity to height (v/d) of the number 4 flight, expressed in the world frame. Both methods use original images.

3.5. CONCLUSION

In this chapter, we have shown that CNNs are suitable for ego-motion estimation of fast-moving MAVs equipped with a downward-facing camera. When flying fast, both motion blur and the visual disparity between subsequent images increase, which is handled better by a network than by traditional feature-based methods. Our investigation into the training of an ICSTN-based network shows that (1) it is better to take all blocks' prediction errors into account, (2) a larger receptive field that can be achieved by pyramidal images allows to estimate larger motions, (3) self-supervised learning based on the photometric error leads to better performance.

3.6. APPENDIX

3.6.1. NETWORKS WITH SHARING PARAMETERS AMONG BLOCKS

Sharing parameters among the blocks of an ICSTN-based network is a way to widen the network (have more convolution kernels in each layer) without enlarging the model size. We train the 6th network of Table 3.1 and its variants by self-supervised learning. The results are shown in Table 3.6. Besides sharing all the parameters we also try only sharing the fully-connected layer (FC).

The results show that sharing all the parameters and keeping the width of the network significantly reduce model size but hurt the accuracy a lot. The wider network having a slightly bigger model size fails to outperform the origin, either, shown in the 2nd row. Besides, it is slower (96Hz) than the original (101Hz). Sharing the fully-connected layer

reduces the model size a little at the cost of the slight deterioration of accuracy. An explanation to the results is that, when blocks have different parameters, the first block is trained to better handle bigger disparities and the last block focuses more on the smaller ones. Although sharing parameters widens the network, its enhancement to the model capacity of the block is less than the negative effects of weakening its specialization.

Table 3.6: Networks with shared parameters.

Share Params	Num. Params (original / wider)	EPE's Approx. Normal Distr.: std(**1e-4**) (original / wider)
None	1.252M	(4.59, **4.04**, **5.94**)
All	0.417M / 1.259M	(8.68, 8.42, 12.05) / (6.28, 5.96, 8.55)
FC	1.221M	(**4.44**, 4.11, 6.02)

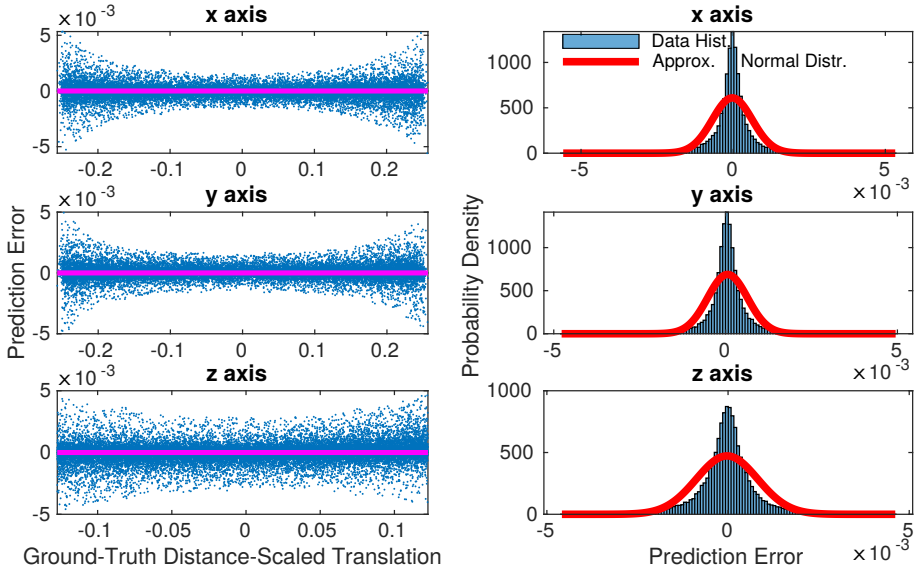


Figure 3.6: Error distribution of the 6th network of Table 3.1 trained in the self-supervised manner.

3.6.2. ERROR DISTRIBUTION OF NETWORK'S PREDICTION

In order to better illustrate the performance of the networks, Fig. 3.6 shows the error distribution of the 6th network in Table 3.1 trained in a self-supervised manner. Other networks' error distribution figures have similar shapes. After the outlier rejection described in Section 3.2, its inlier rate is 85.2%. From the left graph of Fig. 3.6, one can notice that the errors lie approximately unbiasedly close to zero and the predictions in the z-axis are noisier than the other 2 axes. For all the networks in this chapter the z-axis has worse predictions. This is also the case for most networks in [25]. It is possible that a standard

CNN's ability to estimate scale variations is fundamentally limited (cf. [41, 42]). A deeper analysis of this issue is required.

Another phenomena worth noticing is that the network has noisier predictions with bigger translations. It is also shown in Fig. 3.4, the uncertainty of prediction grows with the amount of motion. And the prediction error does not perfectly normally distribute, as shown in the right graph of Fig. 3.6. Predicting the uncertainty of the pose prediction will be studied in future works.

3.6.3. PUBLIC HIGH-SPEED FLIGHT DATASET AND PRIOR POSE

Table 3.7: Evaluation by a public dataset of fast MAV flight.

Sequence	RMSE (**1e-1**): Prior Pose Input / Zero Input
2 (6.97m/s)	(**10.76**, **11.81**, **7.53**) / (14.48, 12.63, 8.70)
4 (6.55m/s)	(**9.94**, **14.64**, **6.80**) / (12.16, 14.77, 8.40)
9 (11.23m/s)	(**9.70**, **12.78**, **10.55**) / (21.15, 15.07, 15.77)
12 (4.33m/s)	(**7.74**, **8.44**, **6.30**) / (9.38, 8.85, 6.48)
13 (7.92m/s)	(**9.86**, **19.08**, **6.56**) / (14.18, 19.13, 8.50)
14 (9.54m/s)	(**33.06**, 22.27, **17.08**) / (36.75, **21.21**, 21.37)

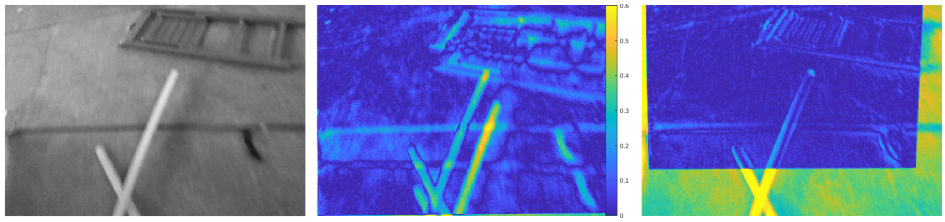


Figure 3.7: The network's performance when the scene is not a perfectly planar surface. The left image is the first image. The middle one shows the photometric error after derotation and the right one shows the photometric error after the second image is warped by the network prediction.

We evaluate the 4th network of Table 3.3 with the 6 indoor 45-degree downward-facing flight sequences that have public ground truth from the UZH-FPV [43] dataset. For this dataset, the distance to the ground is unknown. So we manually set the initial distance at the starting point of the ground truth data to make the RMSE smaller. The peak speed of each sequence and the networks' RMSEs are shown in Table 3.7. Fig. 3.7 shows that the network's performance when the scene is not a perfectly planar surface. Despite the network is disturbed by the low objects lying on the ground because of their rich visual textures, it still outputs relatively accurate predictions. There are other scenes that are not inside a single plane, such as high obstacles and strings. But since most of the time the ground plane takes up the majority of the image, the network has reasonable predictions. Another source of inaccuracy is that the attitude estimation from the Madgwick filter is less accurate because of the big acceleration during fast maneuvers.

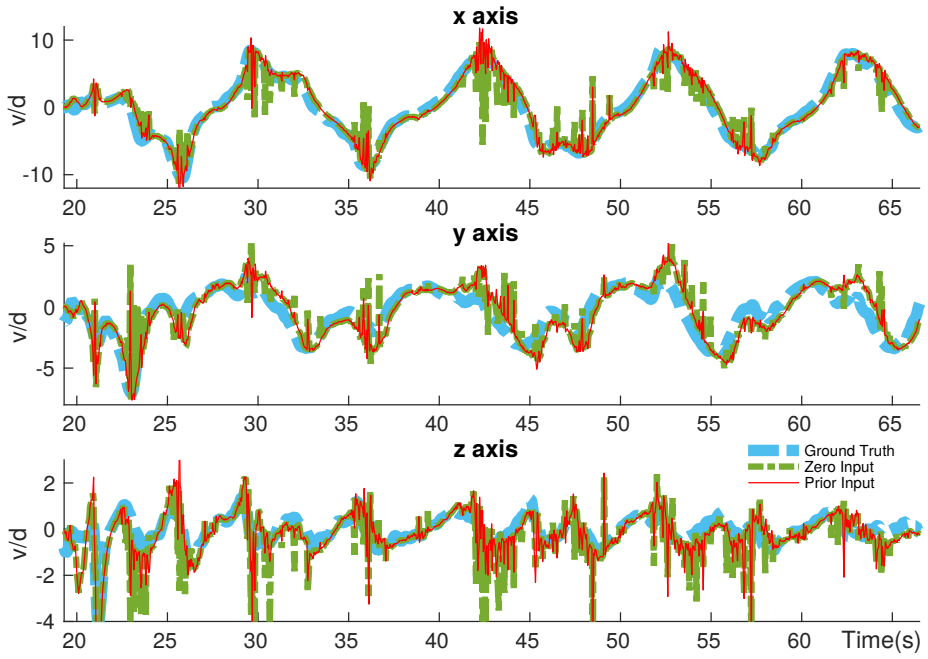


Figure 3.8: The ratio of velocity to height expressed in the world frame. The network's results with and without the prior pose are compared using the number 2 indoor 45-degree downward-facing sequence of the UZH-FPV dataset.

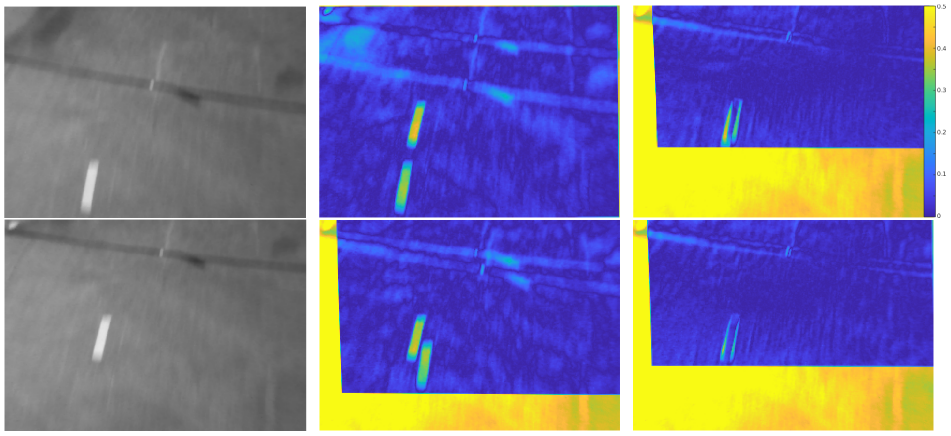


Figure 3.9: Comparison between with and without the prior pose when the visual disparity is big. The left column shows the image pair; In the middle column, the upper image shows the photometric error after derotation and the lower one shows the photometric error of warping by the network prediction when a zero vector is the initial guess of translation; The upper image of the right column shows the photometric error of warping by the prior pose. And the lower one shows the photometric error of warping by network prediction when the prior pose is the input initial pose.

In the previous parts of this chapter, the initial pose (shown in Fig. 3.2) of the network only contains the relative rotation. The translation is set to a zero vector. The network needs to deal with the whole visual disparity caused by translational motion. Because of the inertia of MAV, its translational velocity cannot change much during the sampling interval of the camera. So the translation vectors of temporally adjacent image pairs are similar. We add the predicted translation of the previous image pair as prior information to the initial pose and call it the prior pose. In this case, the disparity of the image pair warped by the initial pose gets smaller.

With the prior pose, the accuracy increases significantly and the predictions are less noisy, as shown in Table 3.7 and Fig. 3.8. We think the reason is that the visual disparities of the image pairs in the UZH-FPV dataset are big enough for the network's accuracy to decrease. Take the number 9 sequence as an example, the average absolute values of the difference between the network's input initial translation and output predicted translation in 3 axes significantly drop from 1.36e-1, 9.51e-2, and 6.46e-2 to 2.14e-2, 1.19e-2, and 1.50e-2 after using the prior pose. This means the network faces much smaller disparities. From Fig. 3.9 one can clearly see the big disparity when the MAV flies at around 9.2m/s close to the ground. Without the prior pose, the network decreases the disparity a lot but still not enough. By contrast, the disparities decrease a lot already after warping by the prior pose, which is easier for the network. The prior pose can be more accurate when other information sources of ego-motion (IMU, dynamics model, etc.) are available. The prior pose makes the network less demanded by big disparities, and thus makes it possible to reduce receptive fields and model sizes of the networks.

3.6.4. CNN-BASED VIO FOR REAL-TIME FEEDBACK CONTROL

For autonomous feedback control of an MAV using only onboard sensors and processors, we implement a CNN-based VIO for high-frequency ego-motion estimation. It is expanded from an EKF-based inertial attitude and velocity estimator [44] that also utilizes the linear drag model of quadrotor MAV. The network (4th of Table 3.3) predictions and the LiDAR measurements are utilized in the measurement update of EKF.

As shown in Eq. 3.1, the translational vector is scaled by the height of the first image. This is also the case in the training of the networks. As for the VIO, we take the current image as the first image of the network input, and the previous image as the second. So the predicted distance-scaled translation is scaled by the current height. This avoids rerunning the EKF from the previous image's time on. As the time interval of the image pair (Δt) is short, we assume the average velocity during Δt approximately equals the instantaneous velocity of the current image. Then the measurement equation of the network prediction can be formulated as Eq. (3.2), where h denotes the height of the camera, ${}^{c}\mathrm{v}$ denotes the velocity vector expressed in the camera frame, and ${}^{c}\mathrm{t}_{net,k}$ denotes the distance-scaled translation vector of the image pair predicted by the network. The prior pose (${}^{c}\mathrm{t}_{net,prior}$) is calculated from the estimated states. We input it to the network as the initial pose to reduce both amount and range of the motion that the network deals with. So we can ignore the varying error distribution of the network predictions over motion as shown in Subsection 3.6.2, and assume that the measurement noise of the network prediction ($\boldsymbol{n}_{net}$) is approximately Gaussian and the noise covariance matrix ($\boldsymbol{R}$) is constant.

$$ {}^{c}\mathrm{t}_{net,prior} = \frac{\Delta t \cdot {}^{c}\mathrm{v}_{k|k-1}}{h_{k|k-1}},\ {}^{c}\mathrm{t}_{net,k} = {}^{c}\mathrm{t}_{net,prior} + \boldsymbol{n}_{net} \tag{3.2} $$

The CNN-based VIO is implemented in C++ and communicates with the controller via Robot Operating System* (ROS). In order to run the network implemented and trained in Python within the VIO, we use TorchScript and LibTorch from the PyTorch C++ API†. TorchScript generates the traced network model that can be loaded and run in C++ by LibTorch functions.

In flight, the average time cost of the network (4th of Table 3.3) inference is around 12.8 milliseconds on the GPU of an NVIDIA Jetson TX2 ‡. Its MAXP_CORE_ARM power mode shows the highest inference speed of our network implementation. The camera's exposure duration is set to 10ms. Although the images look a little dark for bare eyes, the effect on the network's performance can be ignored. This highlights the network's generalizability. For control, a basic proportional-integral-derivative (PID)-based position and velocity controller runs on the TX2 as a ROS node. A Betaflight§ flight controller is in charge of attitude control and connected with the TX2 via a universal asynchronous receiver-transmitter (UART).

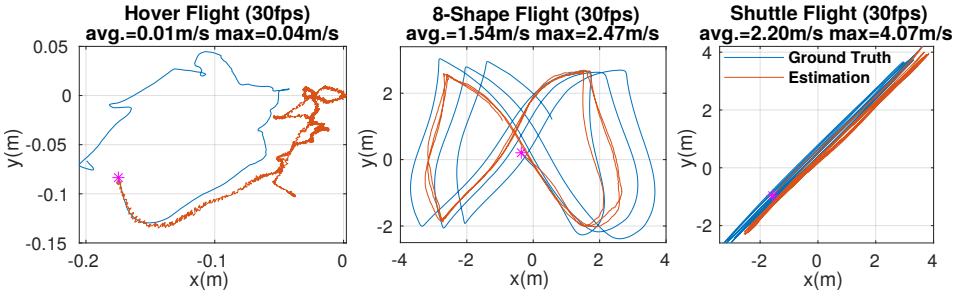


Figure 3.10: The ground-truth and estimated trajectories of hover flight (left), 8-shape flight (middle), and shuttle flight (right).

The MAV performs autonomous hover flight, eight-shape flight with changing heading and height, and high-speed shuttle flight between two waypoints, under using state estimation from the CNN-based VIO. The estimated and ground-truth trajectories for one-minute flights and their average and maximum speed are shown in Fig. 3.10. The link to the flight video is shown in Subsection 3.6.5. As far as we know, this is the first time that the pose estimation network's competence in autonomous feedback control of an MAV is demonstrated. Note that the VIO shown here is basic. Unlike VIO solutions performing mapping, our VIO has no global correction. The network only predicts the relative pose of adjacent images. The trajectory is purely integrated from the network-corrected velocity estimation and thus suffers from drift over time. Taking the average velocity during camera sample interval as the instantaneous velocity and the constant noise covariance

*http://wiki.ros.org/melodic
†https://pytorch.org/docs/stable/cpp_index.html
‡https://developer.nvidia.com/embedded/jetson-tx2
§https://betaflight.com/

matrix of network prediction are two other sources of inaccuracy. There is space for further improvements.

3.6.5. SUPPLEMENTARY MATERIALS

The links to the videos demonstrating the network's performance and autonomous flights are `https://youtu.be/BMdh6dmLgrM` and `https://youtu.be/Uz9pNpn94jU`. The code developed for this work is open-source at `https://github.com/tudelft/PoseNet_Planar`.

REFERENCES

[1] Y. Xu and G. C. de Croon, *Cnn-based ego-motion estimation for fast mav maneuvers,* in *2021 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2021) pp. 7606–7612.

[2] G. De Croon and C. De Wagter, *Challenges of autonomous flight in indoor environments,* in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2018) pp. 1003–1009.

[3] C. Forster, M. Pizzoli, and D. Scaramuzza, *Svo: Fast semi-direct monocular visual odometry,* in *2014 IEEE international conference on robotics and automation (ICRA)* (IEEE, 2014) pp. 15–22.

[4] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, *Orb-slam: a versatile and accurate monocular slam system,* IEEE transactions on robotics **31**, 1147 (2015).

[5] M. Li and A. I. Mourikis, *High-precision, consistent ekf-based visual-inertial odometry,* The International Journal of Robotics Research **32**, 690 (2013).

[6] 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)* (IEEE, 2015) pp. 298–304.

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

[8] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, *Robust stereo visual inertial odometry for fast autonomous flight,* IEEE Robotics and Automation Letters **3**, 965 (2018).

[9] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, *Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam,* arXiv preprint arXiv:2007.11898 (2020).

[10] J. Shi *et al.*, *Good features to track,* in *1994 Proceedings of IEEE conference on computer vision and pattern recognition* (IEEE, 1994) pp. 593–600.

[11] M. Trajković and M. Hedley, *Fast corner detection,* Image and vision computing **16**, 75 (1998).

[12] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, *Orb: An efficient alternative to sift or surf,* in *2011 International conference on computer vision* (Ieee, 2011) pp. 2564–2571.

[13] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, *Alphapilot: Autonomous drone racing,* arXiv preprint arXiv:2005.12813 (2020).

[14] S. Zhong and P. Chirarattananon, *Direct visual-inertial ego-motion estimation via iterated extended kalman filter,* IEEE Robotics and Automation Letters **5**, 1476 (2020).

[15] 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**, 18 (2015).

[16] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu, *Relative camera pose estimation using convolutional neural networks,* in *International Conference on Advanced Concepts for Intelligent Vision Systems* (Springer, 2017) pp. 675–687.

[17] S. Wang, R. Clark, H. Wen, and N. Trigoni, *End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks,* The International Journal of Robotics Research **37**, 513 (2018).

[18] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, *Unsupervised learning of depth and ego-motion from video,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017) pp. 1851–1858.

[19] R. Li, S. Wang, Z. Long, and D. Gu, *Undeepvo: Monocular visual odometry through unsupervised deep learning,* in *2018 IEEE international conference on robotics and automation (ICRA)* (IEEE, 2018) pp. 7286–7291.

[20] A. Ranjan, V. Jampani, L. Balles, K. Kim, D. Sun, J. Wulff, and M. J. Black, *Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019) pp. 12240–12249.

[21] Y. Chen, C. Schmid, and C. Sminchisescu, *Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera,* in *Proceedings of the IEEE International Conference on Computer Vision* (2019) pp. 7063–7072.

[22] J. Bian, Z. Li, N. Wang, H. Zhan, C. Shen, M.-M. Cheng, and I. Reid, *Unsupervised scale-consistent depth and ego-motion learning from monocular video,* Advances in neural information processing systems **32** (2019).

[23] A. Geiger, P. Lenz, and R. Urtasun, *Are we ready for autonomous driving? the kitti vision benchmark suite,* in *2012 IEEE conference on computer vision and pattern recognition* (IEEE, 2012) pp. 3354–3361.

[24] 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 **35**, 1157 (2016).

[25] N. J. Sanket, C. D. Singh, C. Fermüller, and Y. Aloimonos, *Prgflow: Benchmarking swap-aware unified deep visual inertial odometry,* arXiv preprint arXiv:2006.06753 (2020).

[26] O. D. Faugeras and F. Lustman, *Motion and structure from motion in a piecewise planar environment,* International Journal of Pattern Recognition and Artificial Intelligence **2**, 485 (1988).

[27] C.-H. Lin and S. Lucey, *Inverse compositional spatial transformer networks,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017) pp. 2568–2576.

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

[29] D. Sun, S. Roth, and M. J. Black, *A quantitative analysis of current practices in optical flow estimation and the principles behind them,* International Journal of Computer Vision **106**, 115 (2014).

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, *Pytorch: An imperative style, high-performance deep learning library,* in *Advances in neural information processing systems* (2019) pp. 8026–8037.

[31] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization,* arXiv preprint arXiv:1412.6980 (2014).

[32] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks,* in *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010) pp. 249–256.

[33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, *Microsoft coco: Common objects in context,* in *European conference on computer vision* (Springer, 2014) pp. 740–755.

[34] 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.

[35] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, *Densely connected convolutional networks,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017) pp. 4700–4708.

[36] K. He, X. Zhang, S. Ren, and J. Sun, *Identity mappings in deep residual networks,* in *European conference on computer vision* (Springer, 2016) pp. 630–645.

[37] 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.

[38] G. De Croon, H. Ho, C. De Wagter, E. Van Kampen, B. Remes, and Q. Chu, *Optic-flow based slope estimation for autonomous landing,* International Journal of Micro Air Vehicles **5**, 287 (2013).

[39] H. Bay, T. Tuytelaars, and L. Van Gool, *Surf: Speeded up robust features,* in *European conference on computer vision* (Springer, 2006) pp. 404–417.

[40] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, *Estimation of imu and marg orientation using a gradient descent algorithm,* in *2011 IEEE international conference on rehabilitation robotics* (IEEE, 2011) pp. 1–7.

[41] Y. Xu, T. Xiao, J. Zhang, K. Yang, and Z. Zhang, *Scale-invariant convolutional neural networks,* arXiv preprint arXiv:1411.6369 (2014).

[42] N. Van Noord and E. Postma, *Learning scale-variant and scale-invariant features for deep image classification,* Pattern Recognition **61**, 583 (2017).

[43] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, *Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset,* in *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019) pp. 6713–6719.

[44] D. Abeywardena, S. Kodagoda, G. Dissanayake, and R. Munasinghe, *Improved state estimation in quadrotor mavs: A novel drift-free velocity estimator,* IEEE Robotics & Automation Magazine **20**, 32 (2013).

4

CUAHN-VIO: CONTENT-AND-UNCERTAINTY-AWARE HOMOGRAPHY NETWORK FOR VISUAL-INERTIAL ODOMETRY

Learning-based visual ego-motion estimation is promising yet not ready for navigating agile mobile robots in the real world. In this chapter, we propose CUAHN-VIO, a robust and efficient monocular visual-inertial odometry (VIO) designed for micro aerial vehicles (MAVs) equipped with a downward-facing camera. The vision front-end is a content-and-uncertainty-aware homography network (CUAHN). Content awareness measures the robustness of the network towards non-homography image content, e.g. 3-dimensional objects lying on a planar surface. Uncertainty awareness refers that the network not only predicts the homography transformation but also estimates the prediction uncertainty. The training is self-supervised, so that it does not require ground truth that is often difficult to obtain. The network has good generalization that enables "plug-and-play" deployment in new environments without fine-tuning. A lightweight extended Kalman filter (EKF) serves as the VIO back-end and utilizes the mean prediction and variance estimation from the network for visual measurement updates. CUAHN-VIO is evaluated on a high-speed public dataset and shows rivaling accuracy to state-of-the-art (SOTA) VIO approaches. Thanks to the robustness to motion blur, low network inference time (~23ms), and stable processing latency (~26ms), CUAHN-VIO successfully runs onboard an Nvidia Jetson TX2 embedded processor to navigate a fast autonomous MAV.

Parts of this chapter [1] have been submitted to the International Journal of Robotics Research.

4.1. INTRODUCTION

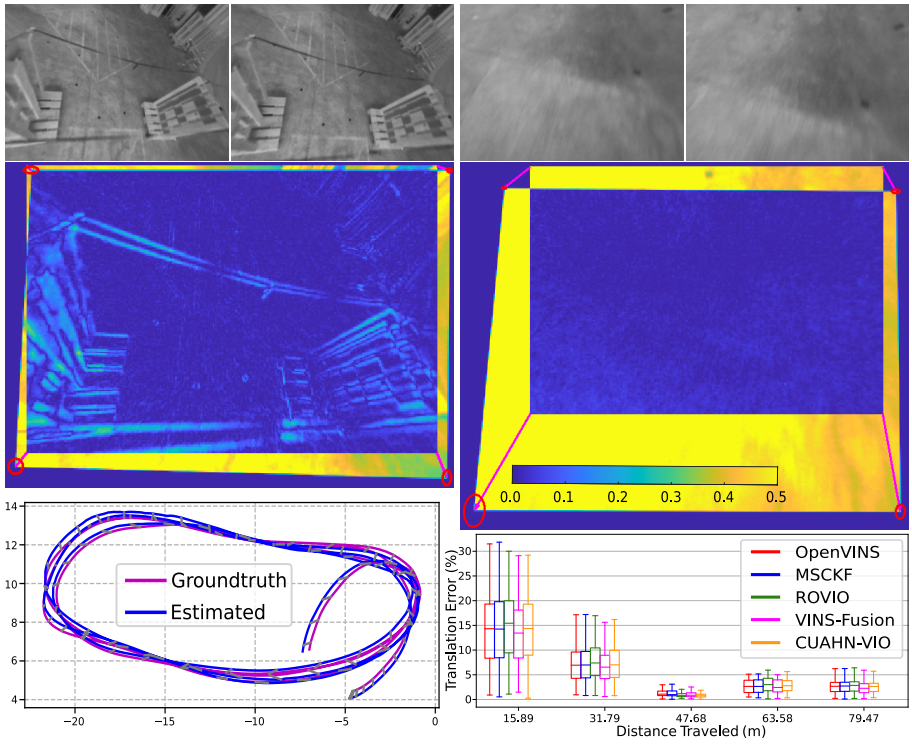


Figure 4.1: Visualized outputs of CUAHN-VIO evaluated on Seq. 13 of the UZH-FPV dataset [2]. CUAHN is robust to sparse non-planar objects and motion blur. Example image pairs are respectively shown on the left and right of the first row. The colormaps (2nd row) show the photometric error between the current image and the previous image warped according to the homography transformation predicted by the network. The arrows in pink are the network-predicted optical flow vectors of the four corner pixels. The red ellipses are the 95% confidence ellipses of the endpoint distributions of the optical flow vectors. They are plotted according to the uncertainty estimation from the network. The trajectory plot in the bottom left aligns and compares the trajectory estimated by CUAHN-VIO with the ground truth. The boxplot in the bottom right shows the relative translation errors of different sub-trajectory lengths. CUAHN-VIO rivals SOTA approaches.

Thanks to the rapid development of computer vision and state estimation techniques, VIO has become a trustworthy component of autonomous robots, such as MAVs. It expands the application scope of MAVs to GPS-denied environments such as indoor spaces. Monocular VIO is attractive to MAVs because it only requires an inertial measurement unit (IMU) and a single camera.

Traditional monocular VIO is built upon projective geometry. Feature-based approaches [3–8] detect and track handcrafted feature points along image frames. Direct approaches [9–11] directly utilize the photometric intensities of pixels. Hybrid approaches [12, 13] combine both. Although such approaches has been widely recognized, their vision front-

ends have inherent defects. They are often affected by disadvantageous and hard-to-model environmental factors, such as motion blur, varying illumination, and textureless regions.

An alternative is learning to predict camera ego-motion by a deep neural network (DNN). As observed in [14–19], DNNs better cope with visually degraded conditions than their handcrafted counterparts. Often referred to as *PoseNet*, the DNN regresses to the six-degrees-of-freedom (6-DoF) relative pose, *i.e.* 3-DoF rotation and 3-DoF translation, between temporally consecutive camera views. The network input is a concatenation of images or an optical flow map, where the camera ego-motion is encoded. In supervised learning, *PoseNet* learns translational motion with metric scale from ground-truth labels [17–22]. But the labels are often expensive to obtain and thus limit the amount of training data. Alternatively, self-supervised learning can be conducted by involving co-training with another network that predicts a pixel-wise depth map [15, 23–30]. The training loss derives from the difference between the actually captured image and the "virtual" one synthesized by image warping according to the predicted relative pose and depth.

When training with monocular videos, translation and depth are scaled mutually to best explain the visual correspondences within the input images. Since there is no constraint on the scales in the loss function, as pointed out in [27], networks not only suffer from scale ambiguity but also have scale-inconsistent predictions over different video snippets. Metric scale can be learned from calibrated stereo images [28, 29] or videos with synchronized IMU data streams [15]. But both methods raise higher demands on training data.

Besides the issue of scale discussed above, we believe that learning-based ego-motion estimation has three major challenges on the road to being trusted in deployment onboard MAVs. The first one is the network generalization capacity. Obviously, the requirement of fine-tuning in every new environment is a fatal barrier to wider application. However, to the best of our knowledge, only three works [14, 16, 20] demonstrated cross-dataset generalization. All of them utilized large datasets synthesized in simulation. In most works, the networks are trained and tested on the same dataset. The most popular is KITTI [31], a car dataset with 3-DoF motion. When it comes to a smaller number of training samples and more difficult motion patterns, networks [15, 18, 19, 21, 22] show worse accuracy than traditional approaches on EuRoC [32], an indoor MAV flight dataset with 6-DoF motion.

The second challenge is network prediction uncertainty. It is typical that most deep learning application works purely pursue prediction accuracy on certain datasets. It is not enough because we lack knowledge of the mechanisms of DNNs and thus highly inaccurate predictions may appear, especially when the input sample is outside the training distribution or distorted by noise. Such outliers can cause a big drift in ego-motion estimation and mislead the robot. Uncertainty estimation can remedy this problem. For example, estimating the uncertainties of each network prediction and using them within the bundle adjustment (BA) back-end lead to better accuracy than constant hand-tuned uncertainty [18, 29].

The last challenge is high computation time. The causes are, for instance, the network being too deep [18], combining multiple networks that together are very large [30], or using an expensive intermediate representation such as a dense optical flow map [14, 17].

Works [14, 17, 18, 30] reported network inference time of more than 40ms measured on Nvidia GPUs designed for desktop computers.

In this chapter, we propose CUAHN-VIO that overcomes the three challenges to a large extent. Instead of *PoseNet*, the vision front-end is a network that predicts the planar homography transformation. It is a pixel-level task and thus generalizes better across cameras with different intrinsics. The network input is a pair of temporally consecutive images captured by a downward-facing camera mounted on an MAV. We show cross-dataset evaluation and real-world flight experiments without any fine-tuning to demonstrate the decent generalization of the network. The network prediction uncertainty is estimated with minor extra computation. It strengthens the system's robustness toward outlier predictions and contributes significantly to VIO accuracy. In terms of inference time, CUAHN-VIO runs faster than 30 frame-per-second (fps) onboard an Nvidia Jetson TX2 mobile processor. Its robustness toward high-speed motion is highlighted in a comparative experiment with a traditional VIO approach.

In the previous chapter, we observed that a network with cascaded architecture better copes with motion blur than handcrafted visual feature points when applied to predicting 3-DoF translational motion. Motivated by this observation, this chapter aims to establish a complete VIO system with a network-based vision front-end to pursue accurate ego-motion estimation in agile maneuvers of MAVs. The network, CUAHN, is more capable than the translation networks in [16], featured by that CUAHN predicts homography transformation that encodes 6-DoF camera motion, and CUAHN estimates the prediction uncertainty.

The main contributions of this chapter can be summarized as:

- We propose a practical scheme of self-supervised training a homography network. It has high prediction accuracy, high-quality uncertainty estimation, and robustness toward sparse 3-dimensional (3-d) structures in view.
- We build a VIO system upon the network and an EKF-based back-end. The metric scale is maintained by the integration of IMU measurements. The network architecture of cascaded blocks makes full use of the EKF *a priori* state, contributing to both accuracy and efficiency.
- To the best of our knowledge, CUAHN-VIO is the first learning-based VIO that not only rivals SOTA approaches in both accuracy and efficiency but also has "plug-and-play" generality and convenience for robot navigation in the real world.

4.2. RELATED WORKS

4.2.1. LEARNING-BASED VISUAL EGO-MOTION ESTIMATION

For *PoseNets* learning to predict 3-DoF translational motion in metric scale from monocular video [17, 18, 20, 28, 29], the scale in testing is recovered by the network's "memory" of the scene structure, *e.g.* the size of objects, in the training set. When testing in a new environment, the scale is possibly inaccurate because of the non-perfect generalization. An extreme case is a miniature park. A car model may be misidentified by the network as a real car that the network has seen in training. Consequently, a translation of a few centimeters may be mistaken for meters of motion. To avoid this problem, TartanVO [14]

recovers up-to-scale translational motion by constraining the normalized translation vector in training. The network is trained on a large-scale dataset of simulation environments and generalizes well to real-world datasets. A potential problem is that the normalized translation is indefinite when the camera is close to stationary or in pure rotation. In the evaluation of [14], the scale of predicted translation is recovered by ground truth metric scale. So the potential bad effect cannot be observed. Another drawback of this approach is that calculating the optical flow map that is the input of *PoseNet* is computationally heavy.

When IMU measurements are fed along with video, two separate subnetworks can be respectively in charge of visual and inertial processing at different sensor rates and output two intermediate tensors. And another subnetwork takes the concatenated intermediate tensors as input to perform sensor fusion and pose prediction [19, 21]. This setup has a principle-level generalization issue. Networks for IMU processing and sensor fusion implicitly "remember" the sensor setup of the training set, *e.g.* bias and noise characteristics of IMU and the extrinsics between IMU and camera. Generalizing to a new sensor setup is difficult. IMU data is low-dimensional and has well-understood models that are grounded in physics. Practising this idea, an end-to-end supervised learning scheme for a loosely-coupled VIO is proposed in [22]. Its back-end is a differentiable EKF whose states are propagated by integrating IMU measurements.

For self-supervised learning, SfMLearner [23] firstly proposed to simultaneously train two networks that respectively predict $T_{t\to s}$ and D_t. $T_{t\to s}$ is the relative pose between source image I_s and target image I_t. D_t is the pixel-wise depth map of I_t. An image $\widetilde{I}_s$ can be synthesized by warping I_s according to the 2-d projections of the 3-d point cloud established from D_t on the image plane of I_s located at $T_{t\to s}$. Based on the assumption that the pixels in consecutive images corresponding to the same point in the scene have the same intensity, the supervision signal derives from the photometric difference between $\widetilde{I}_s$ and I_t. We call it reprojection-based loss for simplicity. This scheme was further developed by also predicting D_s and punishing the 3-d geometric inconsistency between D_t and D_s [26, 27].

Also using reprojection-based loss, SelfVIO [30] performs self-supervised learning of a depth network and three subnetworks for pose prediction. They have the same functions as the subnetworks of supervised-learning VIO [19] and [21]. IMU measurements bring in motion information with metric scale, however, as pointed out in [15], the IMU processing network has no knowledge of the physical model of IMU and the reprojection-based loss does not account for scale. So the metric scale of the IMU measurements is transformed by the trained network and thus predictions still have no metric scale. Extended from [22], the *PoseNet* prediction in [15] is also fused with the IMU-propagated *a priori* states by an EKF. The refined *a posteriori* ego-motion and the output of a depth network together minimize the self-supervised reprojection-based loss in training. The metric scale is obtained by explicitly integrating IMU measurement according to its physical model. But the authors used a 7-DoF similarity transformation (*Sim3*) for trajectory alignment to quantify the VIO accuracy. So we do not know how well the scale of their VIO output matches the metric scale.

4.2.2. NETWORK UNCERTAINTY ESTIMATION IN COMPUTER VISION

According to the taxonomy of [33], for a deep network model, there are two major types of uncertainty that can be modeled. *Aleatoric* uncertainty captures noise inherent in the network input. It can be learned from the real data distribution by the network [34]. The loss function is the negative log-likelihood (NLL) loss. We referred to it as predictive uncertainty to emphasize the way by which it is obtained.

Epistemic uncertainty reflects the ignorance about the *perfect* model that maps clean noiseless input to the desired output. It can be explained away given enough training data. Bayesian neural networks [35] model the trainable network parameters as distributions instead of deterministic values to explain the *Epistemic* uncertainty in the parameters. Since exact Bayesian inference is computationally intractable for DNNs [36, 37], practical strategies of approximate inference were developed such as ensembles of DNNs (deep ensembles) [36] and Monte Carlo Dropout (MC-Dropout) [38]. Given a certain input, these methods estimate the distribution of the network prediction by combining the multiple outputs of an empirically sampled subset of all the possible network instances.

The models of deep ensembles are different point estimates (instead of distribution) of model parameters. They are trained independently to de-correlate their predictions. Ensemble members can be trained on different randomly sampled subsets of the entire training set, referred to as bootstrapping. To approximate a similar effect in a computationally more efficient way, MC-Dropout requires only a single network model trained with dropout, while also deploying dropout during inference, such that multiple independent models are randomly sampled via multiple forward passes. *Epistemic* uncertainty is referred to as empirical uncertainty in this chapter to highlight its acquisition approach.

The above introduced uncertainty estimation approaches have been applied to computer vision tasks. Predictive uncertainty has been proven effective in the prediction of object pose [39], camera ego-motion [15, 18, 22, 40, 41], monocular depth [33, 40, 42], optical flow [43], semantic segmentation[33], and image classification [36]. For empirical uncertainty, deep ensembles were evaluated in image classification [36], optical flow [43], and monocular depth [42]. Likewise, MC-Dropout was adopted in networks for optical flow [43], monocular depth [33, 42], semantic segmentation [33], and camera pose regression [44].

Our purpose in studying network uncertainty estimation is for a better knowledge of visual measurement to benefit Bayesian state estimation. With the similar aim, Kaufmann *et al.* [39] fuse the network-predicted gate pose and its uncertainty with outputs of a VIO system by an EKF. The purpose is to compensate for the gate displacement and the accumulating error of VIO in autonomous drone racing. Embedded in a traditional VO, D3VO [29] leverages the predictive uncertainty. The uncertainty map of photometric matching acts as the weights of the photometric energy in the BA back-end. The relative pose network of D3VO has no uncertainty estimation, so the weights in the optimization of pose energy are set as constant. In [18], six predictive standard deviations of 6-DoF relative pose are used in BA that optimizes a pose graph and achieve higher accuracy than constant hand-tuned covariance. Differently, Li *et al.* [22] proposed to learn predictive uncertainty through the Bayesian nature of a differentiable EKF instead of the widely used NLL loss. The supervision signal is the gradient flow coming from the *a posteriori* ego-motion that is a function of the measurement noise covariance matrix $\boldsymbol{R}$ in EKF updating. Since the

a posteriori states are functions of the whole filter, the learning of $\boldsymbol{R}$ is implicitly affected by the EKF hyperparameters, *e.g.* the process noise covariance matrix $\boldsymbol{Q}$, which poses a potential of overfitting.

Most works adopt uncertainty estimation in supervised learning. About self-supervised learning, Poggi *et al.* [42] made a step in the field of monocular depth. A strategy called Self-Teaching was proposed to decouple depth from pose. The network that outputs predictive uncertainty is trained by the NLL loss and supervised by the outputs of an already trained depth network with the same architecture. The self-supervised EKF-based VIO [15] learns predictive uncertainty of relative camera pose from the error of *a posteriori* ego-motion, same as the supervised VIO [22]. Because the current network prediction affects the later *a posteriori* states, the network is supposed to adjust the current covariance prediction according to the error of *a posteriori* ego-motion in the future. So sequential training data having enough length is required.

4.2.3. DEEP PLANAR HOMOGRAPHY

When a camera films a 3-d point on a planar surface from different poses, the 2-d projections of this point on the image planes can be mapped by a planar homography transformation. It is a function of the ego-motion of the camera and thus useful for a VIO system. It can be inferred by a DNN from an input image pair. Both supervised [45–47] and self-supervised [48, 49] learning schemes have been proposed.

A planar homography transformation can be based on visual correspondences between the image pair. Multiple cascaded network blocks can predict the transformation parameters incrementally [16, 46, 47]. In this scheme, image warping and synthesizing operation is inserted between every two adjacent blocks. After the inference of each block, an image is synthesized by warping the original one using bilinear interpolation [50] according to the prediction(s) of the previous block(s). The next block infers from the synthesized image and the other image. Between them, there are supposed to be fewer visual disparities than the original image pair. In this way, each block predicts a part of the total transformation. Compared with a single deep network, this strategy can lead to higher accuracy and less difficulty in training thanks to the involvement of geometric knowledge and shallower architectures of network blocks.

In many applications, it is not the case that all the visual correspondences can be explained by homography transformation. Masking out the non-homography pixels, *e.g.* the ones filming 3-d structures or dynamic objects, has the goal of boosting accuracy. In [47], a convolutional decoder is added to the homography network for mask prediction. Two masks for the input image pair are predicted together and then concatenated with the images. The concatenation is the input to the next cascaded network block. In this work, mask prediction is learned from the ground truth labels. Instead, Zhang *et al.* [49] implicitly learn the mask in a self-supervised way. An extra subnetwork predicts a mask for each input image. And then the mask is multiplied with the feature map of the image. The idea behind this is that the mask can weigh down the influence of non-homography pixels. The homography network infers from the mask-weighted feature maps for the transformation that is constrained by the self-supervised loss function.

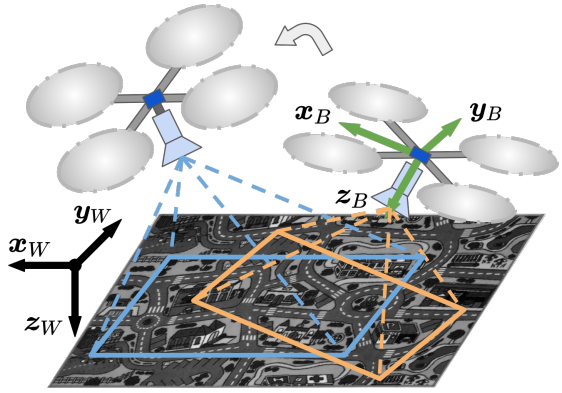


Figure 4.2: An overview illustration of the application scenario, sensor setup, and coordinate definition. W stands for world frame and B stands for body frame, *i.e.*, IMU frame.

4.3. SYSTEM OVERVIEW

Fig. 4.2 illustrates that CUAHN-VIO applies to MAVs that are equipped with an IMU and a downward-facing monocular camera. From a pair of temporally consecutive images, the vision front-end, a DNN (Fig. 4.5), predicts the planar homography transformation and the uncertainty. They are utilized in updating the EKF back-end as shown in Fig. 4.3.

Learning-based VIO approaches [15, 19, 21, 22, 30] perform end-to-end learning, *i.e.*, the ego-motion inferred from both inertial and visual measurements is under constraint in the loss function. The whole VIO system is obtained from a single training attempt. But there are disadvantages. First, videos with synchronized IMU streams are required in training. They are expensive to collect, which places a barrier to enlarging the training set. Besides, extra work is required to obtain the initial poses of data sequences in self-supervised learning [15]. Second, although training the VIO submodules together contributes to in-domain accuracy, the VIO system can overfit the sensor setup of the training set.

By contrast, the DNN of CUAHN-VIO is trained alone, totally decoupled from the VIO system. The benefit is the better generalization capacity. The network has no requirement for camera intrinsics or the camera-IMU extrinsics. Changes to the sensor set only require modifying the back-end parameters without any change in the network. Besides, we do not require sequential training data. A large number of easy-to-obtain simulation image pairs (Subsection 4.4.1) enable the network to generalize to real-world scenes without any fine-tuning.

In the context of no ground-truth label, our approach requires training two networks. They are the student network acting as the VIO front-end and the teacher network. The teacher network has more layers than the student network to gain more accuracy. It is trained by a self-supervised loss function based on photometric matching (Subsection 4.4.2). Content-aware pixel-wise masks are predicted to mitigate the negative impacts of the pixels whose photometric error cannot be reduced by a better homography transformation (Subsection 4.4.3). The teacher network is required because its mean value predictions of homography transformations are needed by the student network as targets

to learn predictive uncertainty by the NLL loss (Subsection 4.5.2). The student network estimates empirical uncertainty by deep ensembles or MC-Dropout (Subsection 4.5.3). Uncertainty estimation turns out to be important in improving the VIO accuracy.

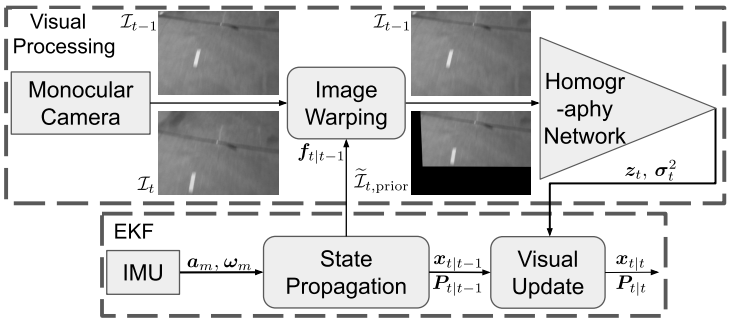


Figure 4.3: An overview data flow diagram of CUAHN-VIO.

The back-end of CUAHN-VIO is a simple extended Kalman filter (EKF), as shown in Fig. 4.3 and introduced in detail in Subsection 4.6.2. It is propagated by IMU integration that explicitly maintains the metric scale. The network-predicted homography transformation $\boldsymbol{z}_t$ and its uncertainty $\boldsymbol{\sigma}_t^2$ update the filter at the frame rate. The *a priori* homography transformation parameterized as four optical flow vectors $\boldsymbol{f}_{t|t-1}$ is utilized for pre-warping the current image I_t. The new image $\widetilde{I}_{t,\text{prior}}$ synthesized by warping is more similar to the previous image I_{t-1} unless the EKF totally diverges. The smaller visual disparities make the task of the network easier. This is especially helpful in fast flight when the optical flow is big. With this prior information, running fewer network blocks produces higher accuracy.

4.4. PLANAR HOMOGRAPHY NETWORK

4.4.1. DATASETS

The training dataset is the same as [16]. It is a big-scale (more than 80 thousand training samples) synthetic dataset with a wide variety of textures, realistic motion blur, and diverse motion patterns. It consists of independent image pairs with small baselines filming perfectly planar surfaces. We refer to it as the Basic Dataset in this chapter.

To involve non-planar and dynamic content, we collected a flight dataset by a MYNT EYE D1000-120 camera downward-facing mounted on a quadrotor MAV. It has 20 videos in which 44,837 image pairs were selected for training, 3,904 for validation, and 4,577 for testing. We put many objects of various heights on the floor that the camera filmed. Some of them moved due to the downwash from the MAV propellers. The ground-truth homography transformations were calculated from the camera poses measured by an OptiTrack motion capture system. Example images are shown in the left three columns of Fig. 4.6. This dataset is called the MYNT Dataset.

The inputs of all networks in this chapter are required to be undistorted grayscale images with the resolution of 320×224. There is no requirement on camera intrinsics.

4.4.2. SELF-SUPERVISED CASCADED NETWORK BLOCKS

When the network acts as a VIO vision front-end, its input images are temporally consecutive, so we refer to them as the previous image I_p and the current image I_c. The homography transformation is parameterized as four 2-d optical flow vectors in the image plane of I_c. They point from the image corners to the pixels corresponding to the corners of I_p, as illustrated in Fig. 4.4. For simplicity, they are called 8-d corner flow $\boldsymbol{f}$. This four-point parameterization of homography transformation is widely adopted by learning-based homography estimation [45–48] and it shows better performance than 3×3 homography matrix when using traditional methods [51].

As shown in Fig. 4.5, the proposed network has four cascaded blocks that are gradual in terms of the number of layers and the resolution of the input. The 1st block is the shallowest and its input is the most downsampled. The 4th block is the deepest and it infers from full-resolution images. An input tensor to a network block is made of two (downsampled) images concatenated along the channel dimension. The 1st block infers from the downsampled I_p and I_c and regresses to $\boldsymbol{f}_1$. The direct linear transformation (DLT*) solver calculates the homography matrix $\boldsymbol{H}_1$ from $\boldsymbol{f}_1$. The correspondence between the float pixel coordinate (u_c, v_c) in I_c and the integer pixel coordinate (u_p, v_p) in I_p is

$$\lambda[u_c, v_c, 1] = \boldsymbol{H}_1[u_p, v_p, 1]^T. \tag{4.1}$$

Homography transformation has 8 DoFs while homography matrix $\boldsymbol{H}$ has nine elements. λ is a scalar that makes the equation true when $\boldsymbol{H}$ is given. With (u_c, v_c), a new image can be synthesized by warping I_c using differentiable bilinear interpolation [50]. The synthesized image is referred to as $\widetilde{I}_{c,1}$, the warped I_c according to $\boldsymbol{f}_1$. $\widetilde{I}_{c,1}$ is then downsampled and concatenated with the downsampled I_p to form up the input tensor of the 2nd block. $\boldsymbol{f}_2$, the prediction of the 2nd block, is supposed to point from the corners of $\widetilde{I}_{c,1}$ to the pixels in $\widetilde{I}_{c,1}$ that have the same intensities as the corners of I_p. $\boldsymbol{H}_2$ is integrated with $\boldsymbol{H}_1$ by matrix multiplication to produce the updated $\boldsymbol{H}_{\text{integ.},2} = \boldsymbol{H}_1\boldsymbol{H}_2$. $\boldsymbol{H}_{\text{integ.},2}$ is used to warp I_c to synthesize $\widetilde{I}_{c,2}$, which is an input to the 3rd block. The same processes repeat for the 3rd and 4th blocks. The later warping is based on the refined $\boldsymbol{H}_{\text{integ.},i} = \boldsymbol{H}_1\boldsymbol{H}_2\cdots\boldsymbol{H}_i$. Thus there should be fewer discrepancies between the pair of (downsampled) images that are input to the next block. In this way, blocks running earlier are trained to capture bigger disparities and the later blocks are good at refining $\boldsymbol{H}_{\text{integ.},i}$ by inferring from the more-and-more similar images. The final prediction $\boldsymbol{f}_{\text{total}}$ is the total corner flow between I_c and I_p. It is obtained by

$$\lambda(\boldsymbol{f}_{\text{total},j} + \boldsymbol{c}_{j,I_c}) = \boldsymbol{H}_{\text{integ.},3}(\boldsymbol{f}_{4,j} + \boldsymbol{c}_{j,\widetilde{I}_{c,3}}), \tag{4.2}$$

where j indexes over the four corners and $\boldsymbol{c}_j$ is the corner pixel coordinate. $(\boldsymbol{f}_{4,j} + \boldsymbol{c}_{j,\widetilde{I}_{c,3}})$ is the predicted coordinate of the pixel in $\widetilde{I}_{c,3}$ corresponding to the jth corner of I_p. $(\boldsymbol{f}_{\text{total},j} + \boldsymbol{c}_{j,I_c})$ is the coordinate of a pixel in I_c that has the same intensity as the pixel at $(\boldsymbol{f}_{4,j} + \boldsymbol{c}_{j,\widetilde{I}_{c,3}})$ in $\widetilde{I}_{c,3}$. Here pixel coordinates are 3-d homogeneous coordinates.

*An asterisk indicates that further elaboration is available in Appendix. Applying to the whole chapter.

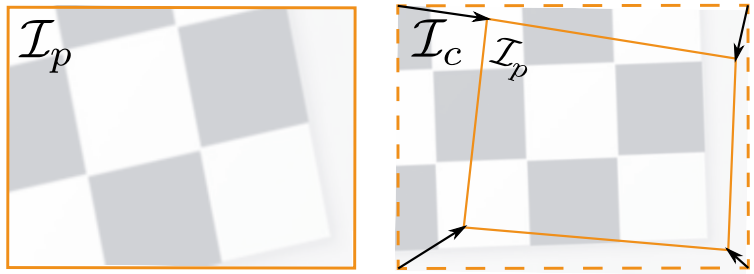


Figure 4.4: An example of 8-d corner flow $\boldsymbol{f}$. Images are adapted from [52].

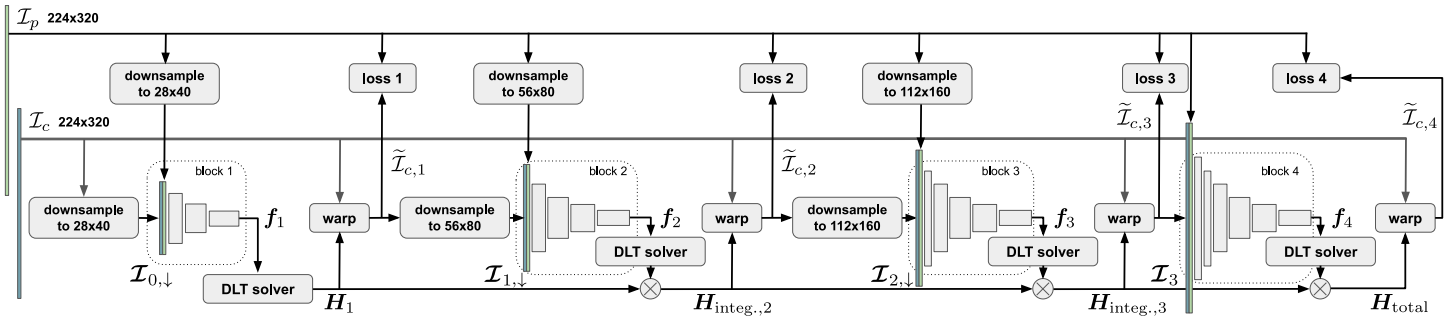


Figure 4.5: The architecture of cascaded network blocks for planar homography transformation prediction. The downward arrow in the foot marker of $\boldsymbol{I}_{i,\downarrow}$ indicates that it has been downsampled. Data flows correspond to training. In inference, loss terms are not calculated and there is no DLT solver for $\boldsymbol{f}_4$. The network output $\boldsymbol{f}_{\text{total}}$ is obtained by Eq. (4.2).

The similarity between $\widetilde{I}_{c,i}$ and I_p indicates how accurate is $\boldsymbol{H}_{\text{integ.},i}$. The loss function for self-supervised learning is given in Eq. (4.3) and Eq. (4.4).

$$L_i = \frac{1}{|V|} \sum_{k \in V} L(\boldsymbol{I}_{i,k}) \tag{4.3}$$

$$L(\boldsymbol{I}_{i,k}) = \frac{\alpha}{2}(1 - \text{SSIM}(\boldsymbol{I}_{i,k})) + (1-\alpha) \cdot |I_{p,k} - \widetilde{I}_{c,i,k}| \tag{4.4}$$

V denotes the set of all valid pixels excluding the ones sampled outside the border of I_c. It is a mask calculated from $\boldsymbol{H}_{\text{integ.},i}$. $\boldsymbol{I}_i$ denotes the concatenation of I_p and $\widetilde{I}_{c,i}$. Same as other works [29, 40, 42, 53], we involve both the $L1$ loss of pixel-wise photometric error and Structured Similarity Index Measure (SSIM) loss in the loss function of self-supervised learning, as shown in Eq. (4.4) where $\alpha = 0.85$. Multi-stage losses are calculated from each $\boldsymbol{I}_i$. Their weights are respectively 0.1, 0.2, 0.3, and 0.4 from earlier to later blocks.

The planar homography network is implemented* in a Python environment with PyTorch library and trained on the Basic Dataset by the self-supervised loss. We employed bidirectional training, *i.e.* concatenating an image pair in two opposite orders. The average error of the predicted $\boldsymbol{f}_{\text{total}}$ on the testing set of the Basic Dataset is 0.275 pixels. It is the average of the absolute values of elements of the 8-d error vector that represents the difference between the network prediction and ground truth, *i.e.* $\frac{1}{8}\sum_{j=1}^{4} |\boldsymbol{f}_{j,u} - \boldsymbol{f}_{j,u,\text{GT}}| + |\boldsymbol{f}_{j,v} - \boldsymbol{f}_{j,v,\text{GT}}|$. Note that this is different from the optical flow endpoint error (EPE) utilized by other works [45, 47, 49], which is the average $L2$ distance, *i.e.* $\frac{1}{4}\sum_{j=1}^{4} ||\boldsymbol{f}_j - \boldsymbol{f}_{j,\text{GT}}||_2$. The reason for element-wise averaging is that, as introduced later, the uncertainty of each element of $\boldsymbol{f}$ is estimated independently. The error-variance data pairs for evaluating uncertainty estimation are element-wise. We refer to the trained network as the Basic Model. Its average inference time cost of a single image pair is 28.20 ms in Python environment and 21.16 ms in C++*, measured on a TX2 processor in Max-P ARM power mode.

4.4.3. CONTENT-AWARE LEARNING

The major assumptions made in the self-supervised loss function Eq. (4.4) are that 1) the camera is facing a single perfectly planar surface, and 2) the overlapping content of both images meets the brightness consistency constraint. However, these assumptions can be easily violated in the real world by 3-d structures, moving objects, occlusions, and reflective materials. A straightforward idea is to learn a content-aware (CA) mask to down-weight the losses of pixels violating the assumptions. The mask is supposed to be learned without ground truth. It only acts on the loss function and thus is not required during testing.

To predict such a mask, 4th block is expanded to a UNet[54]-like architecture with skip connections. Its convolutional layers serve as the encoder part. The upsampling decoder part is added and connected to the last convolutional layer. A single mask is inferred from $\boldsymbol{I}_3$. The mask-involved loss function is applied to the final homography transformation prediction $\boldsymbol{H}_{\text{total}}$. The rest blocks keep their original architectures and $\boldsymbol{H}_{\text{integ.},i}$, $i \in \{1,2,3\}$ are still constrained by Eq. (4.4) without taking the mask into account, assuming that the ratio of assumption-violating pixels is not big enough to greatly deteriorate the supervision signal.

We compare two content-aware loss functions. The first one is proposed in [23]. The predicted mask is called the explainability map. Its elements E_k are bounded between zero and one by a Sigmoid activation. E_k indicates the network's belief in how much the assumptions are satisfied for the kth pixel of I_p. The pixel-wise loss is weighted by E_k as shown in Eq. (4.5). A regularization term $L_{\text{reg.}}(E_k)$ encourages non-zero E_k by minimizing the cross-entropy loss with 1.0 so as to prevent the network to minimize the loss by outputing small values for all E_k. If the network predicts the kth pixel to meet the assumptions well, the value of E_k would be close to 1.0 and $L(\boldsymbol{I}_{4,k})$ would be fully minimized. On the contrary, $L(\boldsymbol{I}_{4,k})$ would be ignored if E_k is close to zero.

$$L_{\text{CA,Exp.}} = \frac{1}{|V|} \sum_{k \in V} E_k \cdot L(\boldsymbol{I}_{4,k}) + \lambda_{\text{reg.}} \cdot L_{\text{reg.}}(E_k) \tag{4.5}$$

Another approach considers the content-aware mask and homography transformation as parameters of a Laplacian probability distribution. The nature of the mask is an uncertainty map. This approach was adopted for structure from motion [29, 40] and optical flow estimation [43]. Given the Laplacian probability density function (PDF)

$$p(x|\mu, b) = \frac{1}{2b} e^{\frac{-|x-\mu|}{b}}, \tag{4.6}$$

since we use $L1$ loss in Eq. (4.4), the term $|x-\mu|$ can be replaced by photometric matching loss $L(\boldsymbol{I}_{4,k})$ calculated from the homography prediction. The parameter b in Eq. (4.6) is related to the variance $\sigma^2 = 2b^2$ of the Laplacian distribution. The predicted mask is made of the b_k that corresponds to the kth pixel of the photometric matching map $\boldsymbol{I}_4$. The learning objective is to maximize the PDF, *i.e.*, minimize the NLL loss

$$L_{\text{CA,Lap.}} = \frac{1}{|V|} \sum_{k \in V} \frac{L(\boldsymbol{I}_{4,k})}{b_k} + \log b_k. \tag{4.7}$$

b_k can be understood intuitively as the uncertainty of the indirectly predicted $L(\boldsymbol{I}_{4,k})$, *i.e.* photometric matching uncertainty. From the perspective of the uncertainty of network prediction, b_k encodes the predictive uncertainty induced by the content-related observation noise. If pixel k potentially violates the assumptions and is too difficult for photometric matching, Eq. (4.7) allows the learning process to increase the value of b_k to down-weight $L(\boldsymbol{I}_{4,k})$ and reduce the overall loss. $\log b_k$ prevents b_k to overgrow.

Table 4.1: Comparison of content-aware homography networks (CAHN).

Setups	No Mask	Lap.	Exp. ($\lambda_{\text{reg.}} = 10^{-3}$)	Exp. (2×10^{-3})	Exp. (3×10^{-3})
Avg. E_k	-	-	0.383	0.640	0.779
Avg. Error (pixel) ↓	0.8768	0.8477	0.8483	0.8471	0.8487

The content-aware networks are trained and tested on the MYNT Dataset. Except for the randomly initialized mask prediction decoder, all parameters are initialized by the Basic Model. The average error of the predicted $\boldsymbol{f}_{\text{total, CA}}$ of each setup is shown in the 3rd

row of Table 4.1. The 2nd column is the baseline network without mask prediction and trained by the original loss function in Eq. (4.3). The network of the 3rd column is trained by Eq. (4.7). The rest three columns correspond to networks trained by Eq. (4.5) with different hyperparameters $\lambda_{\text{reg.}}$. Their pixel-wise average values of the predicted explainability maps on the training set are listed in the 2nd row. A bigger value means that more photometric error is taken into account in the loss function.

It is noticeable that the value of explainability map is sensitive to the manually set $\lambda_{\text{reg.}}$, though the effect on prediction accuracy is small. Despite that 3-d structures distribute densely in the MYNT Dataset, the prediction accuracy of all setups is only slightly improved from the baseline. As shown in the 6th row of Fig. 4.6, the example photometric error maps of the baseline look very similar to the 2nd and 4th rows that are outputs of the content-aware networks. The reason behind the minor contribution of content-aware learning can be that the original loss function Eq. (4.3) minimizes the total photometric error, which drives the network to abandon the minority assumption-violating pixels.

As shown in Fig. 4.6, the uncertainty map (3rd row) is clear and corresponds well to the photometric error (2nd row) caused by non-homography image content. The explainability mask (5th row) is very noisy and prone to discount the textures because they cause bigger photometric errors than uniform regions. Besides, the sensitivity of the explainability mask toward $\lambda_{\text{reg.}}$ may induce extra work of parameter tuning. Therefore, we believe that the uncertainty map trained by Eq. (4.7) is the better choice for content-aware learning.

The 2nd and 3rd rows of Fig. 4.6 show the positive correlation between the predicted uncertainty maps and the photometric error maps. Photometric error can be caused by non-homography image content and inaccurate homography transformation. When obvious non-homography pixels exist, we can observe that most pixels with high predicted uncertainty fall on 3-d structures as shown in the three columns on the left. When the scene is perfectly planar, as shown in the rightmost column, the non-zero uncertainty predictions are totally caused by the homography prediction error. So content-aware mask is not ideal for semantic plane segmentation. Its only duty is to down-weight the non-homography pixels in training.

4.5. UNCERTAINTY ESTIMATION

In Subsection 4.2.2, we introduced practical approaches for estimating predictive uncertainty and empirical uncertainty. In this section, they are implemented for uncertainty estimation of the homography network. We gain the knowledge of their uncertainty estimation quality, effects on prediction accuracy, and additional time consumption. In this section, the uncertainty-aware homography networks (UAHN) are trained and evaluated on the Basic Dataset. Content-aware learning is not involved.

4.5.1. CONFIGURATIONS

The correlations between the uncertainty of network outputs are often neglected in practice. The covariances between the pixel-wise predictions are not considered in monocular depth and semantic segmentation [33, 40, 42]. The uncertainty of u and v components of an optical flow vector are separately estimated in [43]. As for pose prediction [15, 18, 22, 39], the six elements are modelled as independent of each other. In this work,

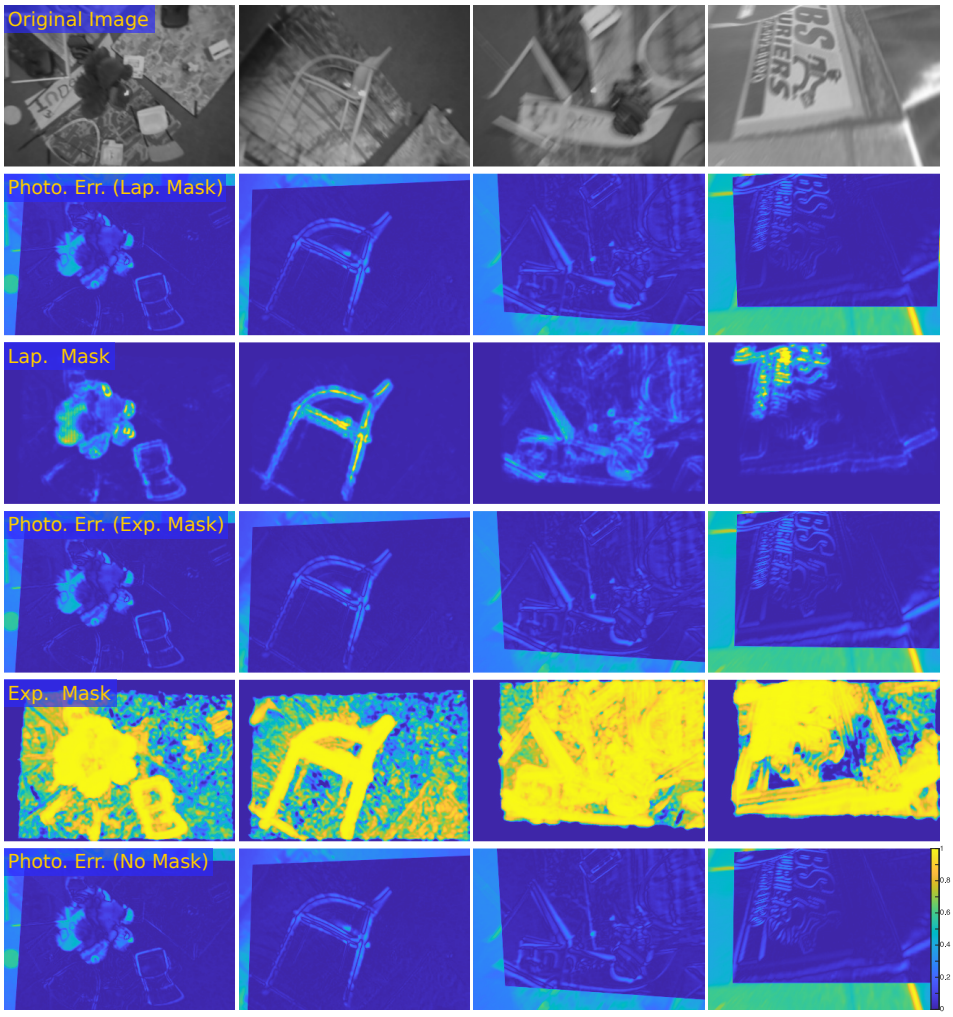


Figure 4.6: From the top row to the bottom row: original image, photometric error maps of the network trained by Eq. (4.7), uncertainty maps (b_k) in Eq. (4.7), photometric error maps of the network trained by Eq. (4.5), explainability maps (E_k) in Eq. (4.5) ($\lambda_{\text{reg.}}$ =2e-3), and photometric error maps of the baseline network. The photometric error map is made of $|I_{p,k} - \widetilde{I}_{c,k}|$, where $\widetilde{I}_c$ is the warped I_c according to the predicted $\boldsymbol{f}_{\text{total}}$ and k is the pixel index. Dark blue means a low error. The 5th row shows the maps of $1 - E_k$. Pixels having small weight in the loss are in yellow, consistent with other rows. The three columns on the left show example images from the MYNT Dataset. The object in the center of the leftmost column is an artificial plastic tree with leaves swaying due to the downwash. The rightmost column shows an image from the Basic Dataset.

we neglect the covariances as well and leave them for potential future works. A scalar variance is estimated for each element of the 8-d mean prediction of homography transformation $\boldsymbol{f}_{\text{total}}$.

As introduced before, our network has four cascaded blocks that infer from their own inputs. It is not necessary for all of them to estimated the prediction uncertainty. The uncertainty of $\boldsymbol{f}_4$ estimated by the last (4th) block is enough to obtain the uncertainty of $\boldsymbol{f}_{\text{total}}$ by the following way. The 4th block infers from $\widetilde{I}_{c,3}$ and I_p and outputs the mean prediction $\boldsymbol{f}_{4,j}$ and the variances $\sigma^2_{u,j}$, $\sigma^2_{v,j}$ of the endpoint of $\boldsymbol{f}_{4,j}$ in the 2-d image plane of $\widetilde{I}_{c,3}$. j indexes over the four corner pixels. $\boldsymbol{\Sigma}_{4,j}$ is a 3-by-3 diagonal matrix whose diagonal elements are $\sigma^2_{u,j}$, $\sigma^2_{v,j}$ and zero. The coordinate of the pixel in I_c that has the same intensity as the endpoint of $\boldsymbol{f}_{4,j}$ can be obtained by Eq. (4.2). Thus the variance of this pixel coordinate, *i.e.*, the variance $\boldsymbol{\Sigma}_{\text{total},j}$ of $\boldsymbol{f}_{\text{total},j}$, is calculated as

$$\lambda^2 \boldsymbol{\Sigma}_{\text{total},j} = \boldsymbol{H}_{\text{integ.},3} \cdot \boldsymbol{\Sigma}_{4,j} \cdot \boldsymbol{H}^T_{\text{integ.},3}. \tag{4.8}$$

Note that the λ here has the same value as the λ in Eq. (4.2). Based on the above explanation, the first three blocks of UAHN stay the same as the Basic Model. Only the 4th block is modified for uncertainty estimation.

$\boldsymbol{\Sigma}_{4,j}$ is a diagonal matrix but $\boldsymbol{\Sigma}_{\text{total}}$ has non-zero non-diagonal elements because of the matrix multiplication. These non-diagonal elements are two orders of magnitude smaller than the diagonal elements in general. So we neglect them and form up the error-variance pair for evaluation by the 2-d error of $\boldsymbol{f}_{\text{total},j}$ and the first two diagonal elements of $\boldsymbol{\Sigma}_{\text{total},j}$. In this way, a testing image pair has eight error-variance pairs.

Same as [42, 43], we adopt Area Under the Sparsification Error (AUSE) as a metric to evaluate the quality of uncertainty estimation. AUSE derives from the "sparsification plot" that reflects how well high errors and high uncertainty coincide. To form a sparsification plot, error-variance pairs are descendingly sorted according to variance. Pairs with the highest variances are removed gradually. If the variances and errors coincide well, the average error of the remaining pairs should decrease while we are removing the data. In contrast, there would be little change in the average error if the variance does not correlate with the error. The ideal sparsification *i.e., oracle sparsification*, is obtained by removing data pairs with the highest errors gradually. An example is the rightmost subplot of Fig. 4.8. The horizontal and vertical coordinates are respectively the ratio of removed data and the average error of remaining data.

The difference between the sparsification formed up by the estimated variances and the ideal sparsification reflects the quality of variance estimation. A sparsification error curve is calculated by subtracting the ideal sparsification from the estimated one. AUSE is the area of the region below the error curve. A lower AUSE means better variance estimation. In practice, to reduce the computation, data pairs are removed in batches. In this chapter, we remove ten pairs at each step to get a data point of the sparsification curve. An AUSE value shown later is the sum of the vertical axis coordinates of all the data points of the sparsification error curve.

AUSE only reflects the relative values among the variances without showing how well their values reflect the values of the errors. For example, for three errors, 1, 2, and 3, the corresponding variances estimated by two approaches are 0.1, 0.2 0.3, and 10, 20, 30, respectively. In this case, their sparsification plots are the same but obviously the former

approach underestimates and the latter overestimates the uncertainty. So we use another metric as a complement. It is the percentage of the testing errors falling into the three standard deviations (3σ) interval, abbreviated as "Inside Rate". A low Inside Rate means that the uncertainty is underestimated.

4.5.2. MODEL DISTILLATION FOR PREDICTIVE UNCERTAINTY

As introduced in Subsection 4.4.3, an additional decoder network predicts the photometric matching uncertainty per image pixel, with the purpose of content-aware learning. Here we shift the focus to the predictive uncertainty of the homography transformation parameterized as the 8-d corner flow $\boldsymbol{f}_4$. The approach proposed in [34] is adopted. A subnetwork of two fully-connected (FC) layers is added to the 4th block to infer the predictive uncertainty from input. It has the same architecture and input tensor as the layers predicting the mean values. The outputs are eight logarithmic variances, $\log\sigma^2_{u,j}$ and $\log\sigma^2_{v,j}$. The training loss is the NLL loss

$$L_{\text{Gaus.}} = \sum_{n=1}^{8} \frac{1}{2\sigma_n^2(\boldsymbol{I}_3)} ||t_n - \mu_n(\boldsymbol{I}_3)||^2 + \frac{1}{2}\log\sigma_n^2(\boldsymbol{I}_3). \tag{4.9}$$

t_n denotes the learning target of the mean value. μ_n and σ_n^2 are respectively the means and variances inferred from the input $\boldsymbol{I}_3$. n indexes over the elements of $\boldsymbol{f}_4$.

Since we aim to build a self-supervised pipeline, ground-truth t_n is not available. Inspired by the Self-Teach scheme proposed in [42], the pseudo label t_n can be generated by a network trained in self-supervised fashion. A student network predicting both mean and variance can be trained by Eq. (4.9) under the supervision of the trained teacher network that outputs only the mean predictions. The student is trained to imitate the teacher by outputting μ_n closer and closer to t_n. The predictive variance σ_n^2 learns to capture how good is the imitation. Thus σ_n^2 only reflects the imitation error $||t_n - \mu_n||^2$ instead of the true error of μ_n *w.r.t.* the ground truth.

The Basic Model has decent accuracy and thus is an option for the teacher network. We referred to it as Self-Teach, same as [42]. Besides, we propose an enlarged version of the Basic Model called the Master Model. It has six network blocks in total. The first three are the same as the Basic Model. The following three blocks have the same architecture as the 4th block of the Basic Model. They together can be treated as a more capable "last block". In the refining training of the Master Model, we initialized the last three blocks by the parameters of the 4th block of the Basic Model. A small improvement in accuracy was achieved and the final testing average error is 0.144 pixels, better than the Basic Model (0.275). The scheme using the Master Model as teacher is called Master-Teach.

Self-Teach and Master-Teach are compared in Table 4.2. To gain more insight, we also trained a student network supervised by ground-truth t_n, abbreviated as GT-Teach. $t_{n,\text{GT}}$, *i.e.* $\boldsymbol{f}_{4,\text{GT}}$, is calculated from $\boldsymbol{f}_{\text{total, GT}}$ and $\boldsymbol{H}_{\text{integ.,3}}$ that is predicted by the first three blocks. The 4th blocks of all the student networks in Table 4.2 are randomly initialized before training. When a training epoch has been finished, the average imitation error is calculated on the validation set. The set of network parameters achieving the smallest average imitation error are recorded for testing.

Table 4.2 shows that GT-Teach has the lowest prediction accuracy but the highest Inside Rate and average variance. The AUSE of Master-Teach is the lowest but its advantage

Table 4.2: Comparison of supervision signals for predictive uncertainty.

Supervision	Avg. Error (pixel) ↓	Avg. Imitation Error (pixel)	Avg. Var.* (pixel)	AUSE ↓	Inside Rate (%) (3σ) ↑
GT-Teach	0.454	0.489	16.66	370.0	**97.63**
Master-Teach	0.428	0.336	7.69	**357.3**	86.96
Self-Teach	**0.408**	0.208	1.83	565.9	79.08

* The networks predict rare unreasonably big variances. The averages are calculated after removing the 0.1% biggest values.

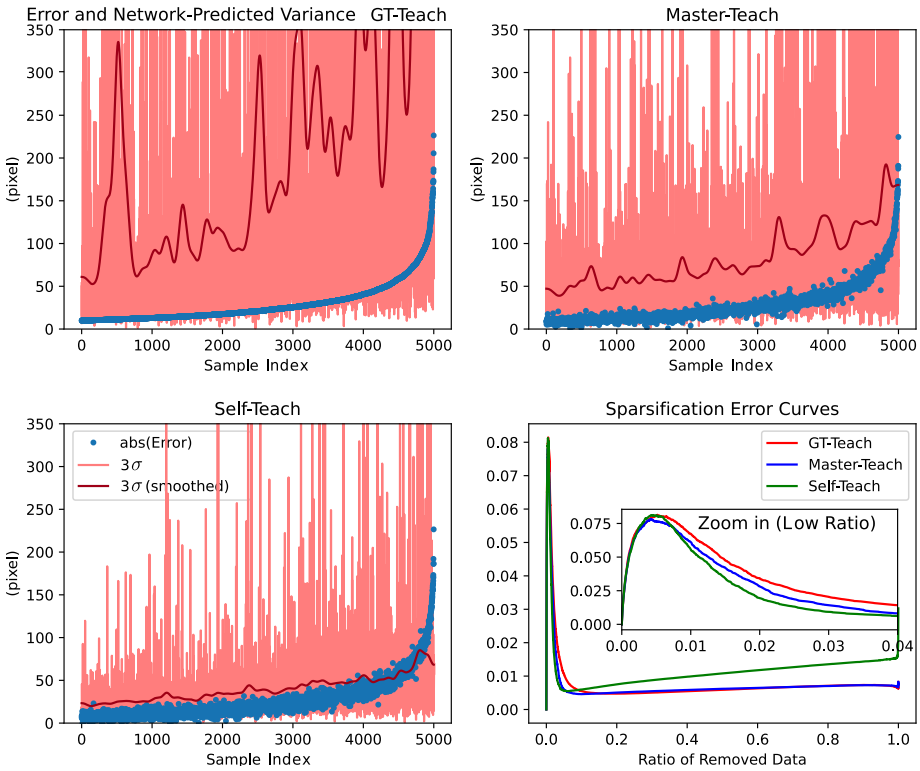


Figure 4.7: Comparison of different approaches to learning predictive uncertainty. The two subplots at the top and the bottom-left subplot: three times of the predictive standard deviations σ and the corresponding prediction errors (absolute values) of GT-Teach, Master-Teach, and Self-Teach, sorted according to the errors of GT-Teach. The 5,000 samples with the **biggest testing errors** are shown. Because 3σ is very noisy, a Gaussian filter is adopted to produce the smoothed curves (in dark red) that allow more intuitive views. Bottom-right subplot: comparison of the sparsification error curves of the three supervision schemes.

over GT-Teach is small. For all other metrics, Master-Teach achieved the middle places. The smallest imitation error and variance indicate that the student model imitates the teacher best in the Self-Teach scheme. But the AUSE and Inside Rate tell us the predictive uncertainty of Self-Teach is the poorest. The low Inside Rate and average variance show that the uncertainty is underestimated.

To visualize the comparison better, we plot the prediction errors and predictive variances in Fig. 4.7. For most testing samples, their error and predictive variance are both small. Here we show the 5,000 error-variance pairs with the biggest errors. Inaccurate predictions like them are dangerous for VIO if the corresponding high variances are not correctly predicted. The error-variance pairs from different supervision schemes are aligned by data indexes and sorted according to the errors of GT-Teach. In this way, the data points with the same index in the three subplots correspond to the same element of the corner flow of the same image pair. We can observe that the three schemes have similar errors for the same testing sample, consistent with the similar average errors in Table 4.2. 3σ grows with error as a general trend. 3σ of GT-Teach is the noisiest and biggest. In contrast, Self-Teach has the smallest σ that is sluggish toward the increasing error and tends to underestimate especially the big errors. The sparsification error curves shown in the rightmost subplot have peaks at a very low ratio, which means that, statistically, the quality of predictive variance is poor when the prediction error is big. For most of the testing data, predictive variance is effective, as evidenced by the low sparsification error curves.

In training, the prediction error of a student network is caused by two factors, the imitation error $||t_n - \mu_n||^2$ and the prediction error of the teacher. As mentioned before, σ_n^2 can only capture the imitation error. So when the imitation errors are big and the teacher errors are small, σ_n^2 well reflects the prediction errors. Conversely, when imitation error is small but the teacher predictions are inaccurate, σ_n^2 keeps a small value and becomes almost irrelevant to the student prediction error. Master-Teach and Self-Teach respectively correspond to the former and latter cases above. Thus Master-Teach produces better predictive uncertainty.

For a sparsification curve, when the ratio of removed data goes higher, remaining data pairs have smaller σ_n^2. As shown in the bottom-right subplot of Fig. 4.7, Self-Teach has an increasing sparsification error curve, which indicates that a smaller σ_n^2 coincides worse with the actual error. The reason for a small σ_n^2 can be that the student network is confident that its mean value prediction is close to the teacher network that supervised it in training. In this case, the student prediction error is close to the unknown teacher error that is not reflected by σ_n^2.

Fig. 4.8 diagrams how the predictive variances cover the prediction errors in a different view from Fig. 4.7. The left subplot shows around 90% of the testing data. Most data points fill up the area between -3σ and 3σ. The local Inside Rate is 86.02%. In the middle subplot, it is noticeable that the variances are much bigger for the remaining 10% of the data. Although the distribution of the errors becomes broader, but not as much as 3σ increases, *i.e.* the extent of overestimation grows with 3σ. The local Inside Rate is 95.38%. As observed in the sparsification plot (right subplot of Fig. 4.8), when the data pairs with the biggest predictive variances (less than 5% of the total) are removed, the average error drastically drops to less than 0.1 pixels. It indicates that, for most testing samples, the Master-Teach student network achieves high prediction accuracy. The uncommon outliers can

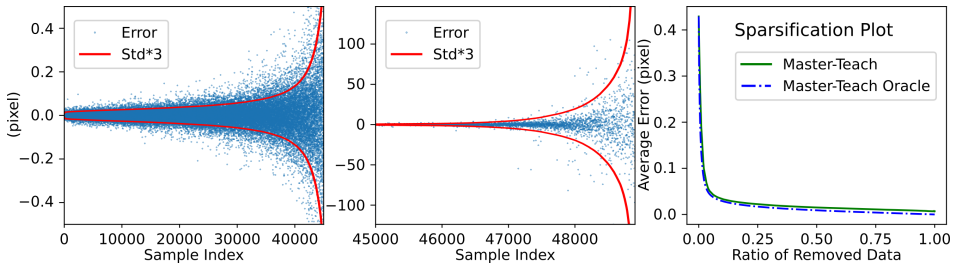


Figure 4.8: Predictive uncertainty of the randomly initialized Master-Teach model (1st row of Table 4.3). Error-variance pairs of the testing set are sorted according to the variances. To avoid overly dense data points in plots, we show one point every ten pairs. Small and big variances are shown respectively in the left and middle subplots with different ranges of y-axis for better visualization.

be revealed by the big predictive variances. The above results corroborate that the quality of Master-Teach predictive uncertainty is generally satisfactory. All the uncertainty-aware networks in the rest of this chapter are trained in Master-Teach scheme.

Table 4.3: Comparison of different initializations of network parameters.

Conv. Layers	FC Layers (mean value)	Avg. Error (pixel) ↓	Avg. Imitation Error (pixel)	Avg. Var. (pixel)	AUSE ↓	Inside Rate (%) (3σ) ↑
Random	Random	0.428	0.336	7.69	**357.3**	**86.96**
Basic	Random	0.376	0.282	5.98	400.0	57.05
Basic	Basic	**0.352**	0.258	4.37	387.9	62.65

Besides random initialization, the student network can be initialized with the trained parameters of the Basic Model. It is clearly shown in Table 4.3 that initializing both convolutional layers and FC mean prediction layers (3rd row) with the Basic Model is better than convolutional layers alone (2nd row). Random initialization (1st row) has the best predictive uncertainty and worst prediction accuracy. The high prediction accuracy of the 3rd row comes from the initial parameters. They also make the imitation error smaller. Thus the predictive variances are smaller and reflect the prediction errors less well, leading to worse predictive uncertainty.

4.5.3. EMPIRICAL UNCERTAINTY

Deep ensembles [36] and MC-Dropout [38] are implemented on the student networks with predictive uncertainty. They both require multiple forward passes to get the samples from the distributions of network parameters. The variance is calculated empirically from the outputs of the forward passes. Same as [33, 42, 43], we combine empirical and predictive uncertainty. The total variance σ_n^2 of the nth element of $\boldsymbol{f}_4$ is shown in Eq. (4.11) as the sum of the empirical variance of the mean value predictions $\mu_{m,n}$ and the average

of predictive variances $\sigma^2_{m,n}$. m indexes over the network model samples.

$$\mu_n = \frac{1}{M}\sum_{m=1}^{M}\mu_{m,n} \tag{4.10}$$

$$\begin{aligned}\sigma_n^2 =& \sigma^2_{n,\text{pred.}} + \sigma^2_{n,\text{emp.}} \\ \sigma^2_{\text{pred.}} = \frac{1}{M}\sum_{m=1}^{M}\sigma^2_{m,n},& \quad \sigma^2_{\text{emp.}} = \frac{1}{M}\sum_{m=1}^{M}(\mu_{m,n}-\mu_n)^2\end{aligned} \tag{4.11}$$

The idea of deep ensembles is to train M network models independently as the samples. Training the models with different bootstrapped subsets of the training data enhances independence. But meanwhile, less training data harms the prediction accuracy. We follow the practice of [36] that using the entire training set for every model, assuming random initialization along with random shuffling of training data produce sufficient independence. We trained eight independent models respectively for the 1st and 3rd initialization schemes in Table 4.3. An ensemble combines eight models at most because the increasing time consumption makes it impossible for real-time inferencing on a mobile processor.

For MC-Dropout, we implement two schemes and two dropout rates. One scheme randomly initializes all parameters and performs dropout before all layers. The other initializes the convolutional layers with the parameters of the trained Basic Model. Following the practice of [44], that is deploying dropout only before layers that are randomly initialized, dropout is only effective before FC layers.

The above-introduced schemes are compared in Fig. 4.9 by four metrics. We find that performing dropout before all layers leads to much worse accuracy and AUSE, besides long inference time. So it is eliminated without being shown. The top-left subplot shows that increasing the number of sampled network models only slightly improves prediction accuracy. While, the AUSE values shown in the top-right subplot vary with the number of samples significantly, especially for deep ensembles. The same trend is observed in Inside Rate (bottom-left subplot). An ensemble of three network models has significantly better uncertainty estimation than a single one. In contrast, more forward passes of MC-Dropout networks produce relatively smaller improvements. The higher dropout rate (10%) performs worse than the lower one (5%) in terms of both accuracy and AUSE, while better in Inside Rate.

As for the two initialization schemes, random initialization has better and bigger predictive uncertainty as shown before. Thanks to more randomness in network parameters, random initialization in theory has better and bigger empirical uncertainty as well and thus has better overall uncertainty estimation. As shown in Fig. 4.9, the two initialization schemes are respectively advantageous in prediction accuracy and uncertainty estimation quality.

For deep ensembles, the time consumption increases significantly with the number of sampled networks. We failed to find a way in our current implementation to speed up, though the samples of the 4th network block are independent and, in theory, can run in parallel. Due to the real-time requirement of VIO, we only consider the ensembles of less than three network samples. For MC-Dropout, instead of inferencing multiple times temporally serially, the intermediate tensors can be duplicated along the batch dimension

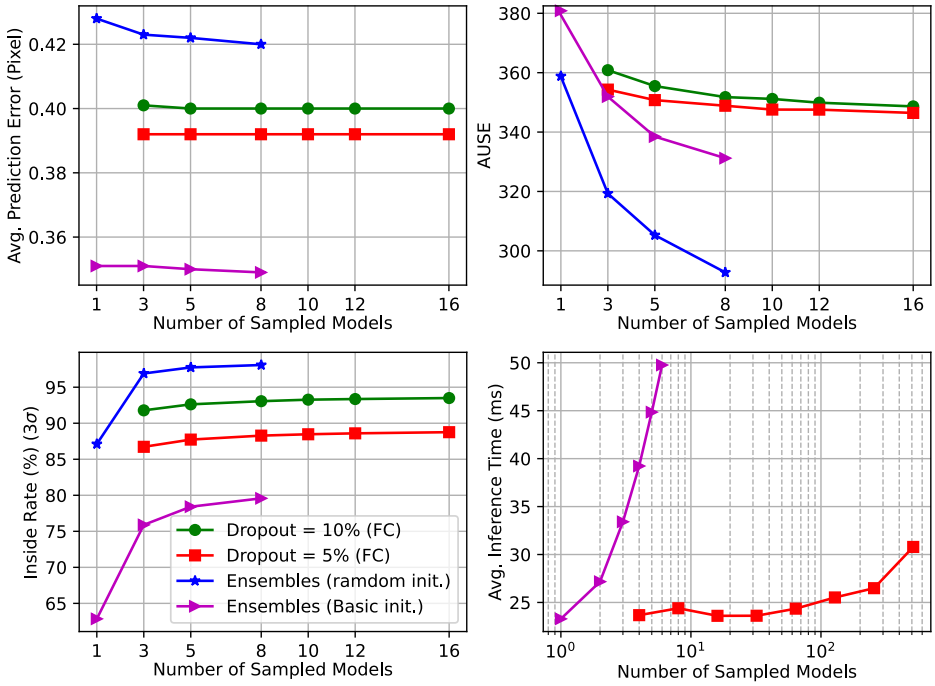


Figure 4.9: Comparison of approaches for empirical uncertainty in terms of prediction accuracy, uncertainty estimation quality (AUSE and Inside Rate), and inference time consumption. The x-axis for deep ensembles is the number of independent network models. For MC-Dropout, the x-axis is the number of forward passes. Deep ensembles indicated by blue stars and magenta triangles correspond respectively to the 1st and 3rd row of Table 4.3. The inference time was measured on a TX2 processor running network inference in a C++ environment. We show the inference time of one of the two networks using MC-Dropout because the other has the same in theory. Same for the deep ensembles.

before dropout to obtain the same effect. The time consumption increases insignificantly as shown in the bottom-right subplot of Fig. 4.9. Based on the overall consideration of the four metrics, we select three candidates that act as the VIO vision front-end and are compared in terms of the resulting VIO accuracy in Subsection 4.7.2. They are 1) the ensemble of randomly initialized models (indicated by the blue stars in Fig. 4.9), 2) the ensemble of models initialized by the Basic Model (magenta triangles), and 3) 5% dropout before FC layers with 16 forward passes (red squares).

Subsection 4.9.9 in the Appendix of this chapter shows the magnitudes of $\sigma^2_{\text{emp.}}$ and $\sigma^2_{\text{pred.}}$ and their correlation for interested readers.

4.6. VISUAL-INERTIAL ODOMETRY

4.6.1. HOMOGRAPHY-NETWORK-BASED VISION FRONT-END

We have introduced how to train CAHN (Section 4.4) and UAHN (Section 4.5). In the following, we describe the way of combining both to get a content-and-uncertainty-aware homography network (CUAHN), and how it acts as the vision front-end of a VIO system.

As discussed in the previous section, Master-Teach is a good choice for the student network to learn the predictive uncertainty. To gain higher accuracy through the robustness toward non-homography image content, we train the Master Model by the content-aware loss Eq. (4.7), different from the Master Model in Subsection 4.5.2 that minimizes the photometric error of all the pixels. Three upsampling decoders are respectively attached to the last three blocks to predict the content-related photometric matching uncertainty maps, as introduced in Subsection 4.4.3. The decoders share parameters. In this way, each of the last three blocks has its predicted uncertainty map and the photometric matching map obtained from the integrated homography transformation prediction $\boldsymbol{H}_{\text{integ.},i}$. The training loss is the sum of the content-aware losses of the three blocks.

It is important for the training set to have enough non-homography contents and also be generic. In this chapter, the six sequences with public available ground truth of UZH-FPV are used for evaluation. We take the 6,070 image pairs from the rest four sequences without ground truth for training and name them the UZH-FPV training set. Together with the generic Basic Dataset, the aggregated dataset is used for training CUAHN.

As introduced in Section 4.5, an uncertainty-aware network estimates the 8×8 covariance matrix $\boldsymbol{R}_{\text{net.}}$ of the corner flow prediction. Theoretically, it should be used directly as the measurement noise covariance matrix $\boldsymbol{R}_{\text{meas.}}$ in the measurement update of EKF. But because the performance of the EKF is under the effects of noise matrices, it is better to have the freedom of tuning the measurement noise. Thus we introduce a manually tuned scalar hyperparameter $k_{\text{var.}}$ to linearly scale $\boldsymbol{R}_{\text{net.}}$ as $\boldsymbol{R}_{\text{meas.}} = k_{\text{var.}} \cdot \boldsymbol{R}_{\text{net.}}$. In practice, it is easy to tune since the system is not very sensitive to $k_{\text{var.}}$.

4.6.2. EKF-BASED BACK-END

The VIO back-end is a simple and very efficient EKF. IMU measurements drive the state propagation and network outputs drive the visual measurement updates. To simplify the filter, we assume that a single plane is observed by the camera throughout the whole video and the plane is orthogonal to the gravity vector. These assumptions apply to many flight arenas, especially indoor environments. The origin of the world frame lies on the plane

and the z-axis is parallel to the gravity vector as shown in Fig. 4.2. The EKF state vector is defined as:

$$\boldsymbol{x} := \left[\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{v}, \boldsymbol{b}_a, \boldsymbol{b}_g, \boldsymbol{f}_j\right], \quad j \in \{ul, bl, br, ur\}. \tag{4.12}$$

$\boldsymbol{p}$ is the position of IMU relative to the origin of the world frame, expressed in the IMU frame. $\boldsymbol{q}$ is the Hamilton quaternion reflecting the relative rotation between the world frame and the IMU frame. $\boldsymbol{v}$ is the translational velocity of IMU expressed in the IMU frame. $\boldsymbol{b}_a$ and $\boldsymbol{b}_g$ are respectively the additive bias on accelerometer and gyroscope. $\boldsymbol{f}_j$ indicates the optical flow vector of the jth corner pixel between two consecutive frames. It is expressed in the current frame, pointing from the corner to the pixel that is supposed to have the same intensity as the corner of the previous frame. The foot markers of $\boldsymbol{f}_j$ are respectively the abbreviations of upper left, bottom left, bottom right, and upper right.

As shown in Eq. (4.13), IMU measurements are modelled as the sum of the desired actual value ($\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{\omega}}$), additive bias ($\boldsymbol{b}_a$ and $\boldsymbol{b}_g$), and white Gaussian noise ($\boldsymbol{w}_a$ and $\boldsymbol{w}_g$).

$$\boldsymbol{a}_m = \hat{\boldsymbol{a}} + \boldsymbol{b}_a + \boldsymbol{w}_a, \quad \boldsymbol{\omega}_m = \hat{\boldsymbol{\omega}} + \boldsymbol{b}_g + \boldsymbol{w}_g \tag{4.13}$$

The initialization of $\boldsymbol{q}$, $\boldsymbol{b}_a$, and $\boldsymbol{b}_g$ is based on the average IMU measurements within a period of time when the MAV stays stationary, implemented in the code of [7].

$$\begin{aligned}
\dot{\boldsymbol{p}} &= -[\hat{\boldsymbol{\omega}}]_\times \boldsymbol{p} + \boldsymbol{v} + \boldsymbol{w}_{\boldsymbol{p}}, \\
\dot{\boldsymbol{v}} &= -[\hat{\boldsymbol{\omega}}]_\times \boldsymbol{v} + \hat{\boldsymbol{a}} + \boldsymbol{R}(\boldsymbol{q})^{-1}\boldsymbol{g}, \\
\dot{\boldsymbol{q}} &= \frac{1}{2}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \hat{\boldsymbol{\omega}} \end{bmatrix}, \\
\dot{\boldsymbol{b}}_a &= \boldsymbol{w}_{\boldsymbol{b}_a}, \quad \dot{\boldsymbol{b}}_g = \boldsymbol{w}_{\boldsymbol{b}_g}, \\
\dot{\boldsymbol{f}}_j &= -(\boldsymbol{I} - (\boldsymbol{c}_j + \boldsymbol{f}_j)\boldsymbol{e}_z^T)\boldsymbol{H}(\boldsymbol{c}_j + \boldsymbol{f}_j)
\end{aligned} \tag{4.14}$$

Eq. (4.14) shows the IMU-driven state dynamics ($\dot{\boldsymbol{x}}$). $[\hat{\boldsymbol{\omega}}]_\times$ is the skew-symmetric matrix associated with $\hat{\boldsymbol{\omega}}$. $\boldsymbol{w}_{\boldsymbol{p}}$ is the process noise in position integration. $\boldsymbol{R}(\boldsymbol{q})$ is a transformation function from $\boldsymbol{q}$ to *SO3* rotation matrix that maps a vector expressed in the IMU frame to its expression in the world frame. $\boldsymbol{g} = [0, 0, g]^T$ is the gravity vector expressed in the world frame. $\otimes$ denotes quaternion product. We utilize the techniques introduced in [55] for quaternion-related calculation. The propagation of $\boldsymbol{f}_j$ is based on the continuous homography transformation. The formula derivation can be found in [52]. $\boldsymbol{c}_j$ stands for the 2-d coordinate of the jth corner pixel. It is a constant parameter calculated from the camera intrinsics. $\boldsymbol{I}$ is a 3×3 identity matrix. $\boldsymbol{e}_z = [0, 0, 1]^T$. In our implementation, $\boldsymbol{f}_j$ and $\boldsymbol{c}_j$ are homogeneous coordinates in the camera frame instead of pixel coordinates, which means that they are expressed on the $z = 1$ plane of the camera frame. So camera intrinsics are not needed in state propagation.

$\boldsymbol{H} \in \mathbb{R}^{3\times3}$ relates the camera motion to the optical flow $\dot{\boldsymbol{f}}_j$. It is known as the continuous homography matrix:

$$\boldsymbol{H} = [\hat{\boldsymbol{\omega}}_c]_\times + \frac{1}{d_c}\boldsymbol{v}_c\boldsymbol{\mu}_c^T \tag{4.15}$$

where

$$\begin{aligned}
\hat{\boldsymbol{\omega}}_c &= \boldsymbol{R}_{CI}\hat{\boldsymbol{\omega}}, \\
\boldsymbol{v}_c &= \boldsymbol{R}_{CI}(\boldsymbol{v} + [\hat{\boldsymbol{\omega}}]_\times \boldsymbol{t}_{IC}),
\end{aligned} \tag{4.16}$$

and

$$\begin{aligned} \boldsymbol{\mu}_c^T &= \boldsymbol{R}_{CI}\boldsymbol{R}^{-1}(\boldsymbol{q})\boldsymbol{e}_z, \\ d_c &= -\boldsymbol{e}_z^T\boldsymbol{R}(\boldsymbol{q})(\boldsymbol{p}+\boldsymbol{t}_{IC}). \end{aligned} \tag{4.17}$$

$\hat{\boldsymbol{\omega}}_c$ and $\boldsymbol{v}_c$ are respectively the angular and translational velocity vectors of the camera expressed in the camera frame. $\boldsymbol{\mu}_c$ is the normal vector of the plane expressed in the camera frame. Based on our assumption, it has the same direction as the gravity vector. d_c is the distance from the camera to the plane. We define the z-axis to be downward as shown in Fig. 4.2. In the cases where the z-axis points up, minus signs should be added to the right of the equal signs in Eq. (4.17).

The visual measurement of $\boldsymbol{f}_j$ is modelled as

$$\boldsymbol{z}_{j,t} = \boldsymbol{f}_{j,t|t-1} + \boldsymbol{w}_{j,t} \tag{4.18}$$

where $\boldsymbol{f}_{j,t|t-1}$ is the *a priori* estimation of $\boldsymbol{f}_j$ propagated by Eq. (4.14). $\boldsymbol{z}_{j,t}$ is the mean value prediction of the whole homography transformation from the network. When $\boldsymbol{f}_{t|t-1}$ is used for pre-warping as shown in Fig. 4.3, the network predicts a part of the transformation and $\boldsymbol{z}_{j,t}$ is the combination of the network prediction and $\boldsymbol{f}_{t|t-1}$. $\boldsymbol{w}_{j,t}$ is the measurement noise. The covariance matrix of $\boldsymbol{w}_{j,t}$ is $\boldsymbol{R}_{\text{meas.}} = k_{\text{var.}} \cdot \boldsymbol{R}_{\text{net.}}$. Note that network outputs are in pixels. So $\boldsymbol{z}_{j,t}$ and $\boldsymbol{R}_{\text{net.}}$ are required to be scaled by the camera intrinsics (focal length) to convert to the homogeneous coordinates in the camera frame, the coordinate system same as $\boldsymbol{f}_{j,t|t-1}$.

$\boldsymbol{f}_j$ is a temporary state reflecting the transformation between two consecutive frames. It has been propagating from zero since the acquisition of the last frame. When a new frame is available, the network inferences from the newest two frames and the difference between the propagated prior $\boldsymbol{f}_{j,t|t-1}$ and $\boldsymbol{z}_{j,t}$ acts as the measurement residual in EKF update. After updating, $\boldsymbol{f}_j$ and its corresponding elements in the covariance matrix of the state vector are reset to zeros.

4.7. EVALUATION

We first compare the proposed VIO with open-sourced SOTA VIO approaches, followed by an ablation study. Then, a generic and efficient variant UAHN-VIO is demonstrated competent for feedback control of autonomous MAV flight in an unseen test environment. Lastly, we compare CUAHN-VIO with a feature-point-based VIO approach MSCKF [4, 7], focusing on analyzing the processing latency and robustness toward fast motion.

The evaluation is mainly by means of the six indoor 45-degree downward-facing sequences trajectories from a public MAV dataset UZH-FPV [2]. It is known for its high flight speed and big optical flow. Another dataset is recorded in autonomous MAV flights by the same hardware as the MYNT Dataset. It features frequent significant motion blur and is utilized in robustness evaluation. KITTI [31] is wildly utilized by works of ego-motion estimation. While it does not suit this work because it is recorded by forward-facing cameras mounted on a car. The cameras captured rich 3-d content. By contrast, CUAHN-VIO is designed for a downward-facing camera mounted on an MAV and requires most of the scene in the field of view to be a single planar surface. EuRoC [32], a popular dataset captured by a forward-facing camera of an MAV, cannot be used in this work for the same reason.

As is common in VIO studies, the root-mean-square error (RMSE) of absolute translation errors (ATE) acts as the metric for VIO accuracy. We utilize an open-sourced tool [56] for the calculation. The estimated trajectory and the ground truth are aligned by the 4-DoF yaw-only rigid body transformation (1-DoF rotation and 3-DoF translation, *posyaw*) corresponding to the four unobservable DoFs for VIO [56]. It reveals how well the scale of the estimated trajectory matches the metric scale. The *Sim3* alignment widely used by other works cannot.

4.7.1. COMPARISON OF ACCURACY WITH SOTA VIO APPROACHES

Table 4.4: Comparison with SOTA VIO approaches. **Bold** represents the best and underline represents the best one in SOTA approaches.

VIO	RMSE (meter) of Absolute Translation Errors (ATE) ↓					
	Seq. 2	Seq. 4	Seq. 9	Seq. 12	Seq. 13	Seq. 14
UAHN (6-3)	**0.3371**	0.3139	0.3392	0.5837	0.4066	1.7905
CUAHN (6-4)	0.3479	0.3138	0.3214	0.5843	0.4095	1.7790
CUAHN (4-4)	0.3475	**0.2739**	**0.3200**	**0.5826**	**0.3985**	1.7903
OpenVINS [7]	0.3438	0.3937	0.3772	0.6252	0.5542	**1.7392**
LARVIO [8]	1.0584	0.8085	0.5069	0.8100	0.8370	2.0767
MSCKF [4, 7]	0.3718	0.3704	0.4189†	0.6347	0.5424	1.7405
ROVIO [12]	0.9175†	0.4233	0.7837†	0.6234	0.4217	1.8286†
VINS-Fusion [6]	0.4040	0.4533†	0.6439	0.6021	0.4544	1.7988†

† Without online calibration.

Based on the ablation study shown later, three setups of CUAHN-VIO (6-3, 6-4, and 4-4 in Table 4.6) are selected to compare with open-sourced SOTA VIO approaches in Table 4.4. The numbers in the 1st row are the sequence numbers. The maximum speeds of the sequences in meters per second (m/s) are shown in the brackets following the sequence numbers in the 1st row of Table 4.6. We used a laptop computer to run the VIO approaches to guarantee no frame was discarded because of slow processing. We tried to get as good as possible results from the SOTA approaches by tuning the parameters*, *e.g.* IMU noise density and the starting time of the data sequences. For approaches having the function of online calibration, we tried both with and without this function and put the better results in the table. ORB-SLAM3 [3] was also tried but it failed to initialize the map or keep tracking it on any sequence. For five sequences out of six, the smallest errors are achieved by UAHN or CUAHN. For Seq. 4 and 9, the advantage is relatively obvious. In general, the proposed VIO rivals the SOTA approaches.

We also compared with DRIOD-SLAM [57], an open-sourced learning-based VO that can run on the evaluation dataset without requiring retraining. Since it is a monocular VO system, the estimated trajectories do not have the metric scale. So we used the 7-DoF *Sim3* trajectory alignment for the comparison of DRIOD-SLAM and UAHN-VIO, as shown in Table 4.5. The preprocessing and frame rate of the input videos are the same for DRIOD-SLAM and UAHN. The global bundle adjustment of DRIOD-SLAM was disabled

Table 4.5: Comparison with a learning-based VO approach. The accuracy metric is based on *Sim3* trajectory alignment. **Bold** represents better.

VIO / VO	RMSE (meter) of Absolute Translation Errors (ATE) ↓					
	Seq. 2	Seq. 4	Seq. 9	Seq. 12	Seq. 13	Seq. 14
UAHN (6-3)	**0.3355**	**0.2941**	**0.3382**	**0.5800**	**0.4057**	**1.7672**
DRIOD-SLAM [57]	1.3882	1.1096	1.4307	7.3421	1.2826	4.3682

in our testing. Table 4.5 shows that UAHN has an advantage in accuracy over DRIOD-SLAM. Regarding time efficiency, we run DRIOD-SLAM on a partition of a multi-Instance Nvidia A100 GPU (four instances). The average processing frame rate is around 10 fps. The input videos have 30 fps, so DRIOD-SLAM did not run in real-time. As to be shown later, CUAHN-VIO ran in real-time on a small-size mobile GPU processor. Thus CUAHN-VIO outperforms DRIOD-SLAM in terms of time efficiency.

4.7.2. ABLATION STUDY

In this ablation study, we aim to gain insights into how the components and setups of CUAHN-VIO contribute to VIO accuracy. The VIO variants are shown in Table 4.6. The 2nd column shows the network acting as the vision front-end of a VIO variant. The 3rd column indicates the initialization method of a network. The last network block can be initialized randomly or by the Basic Model, as introduced in Subsection 4.5.2. The number of network blocks running in a VIO variant is shown in the 4th column. The 5th and 6th columns tell about the measurement covariance matrix $\boldsymbol{R}_{\text{meas.}}$ and the estimation method of empirical uncertainty.

VIO variants are divided into six groups according to the shared setups. The VIO variants in Group 1 have no uncertainty estimation, which means that $\boldsymbol{R}_{\text{meas.}}$ stays constant for all network predictions. The shown value in the 5th column is the identical diagonal element of $\boldsymbol{R}_{\text{meas.}}$. For Group 2 to Group 6, uncertainty estimation is available. The 5th column shows the $k_{\text{var.}}$. Most EKF parameters, *e.g.* $\boldsymbol{Q}$, stay fixed and are the same for all the VIO variants. $\boldsymbol{R}_{\text{meas.}}$ is the only manually-tuned parameter for different VIO variants. For each VIO variant, we run it on Seq. 2 several times to find the $\boldsymbol{R}_{\text{meas.}}$ or $k_{\text{var.}}$ that yields good accuracy. The same value is used for all sequences. In practice, we found that the ATE is not sensitive to $k_{\text{var.}}$. Increasing $k_{\text{var.}}$ can produce a little bit smoother estimated trajectory while slightly enlarging the ATE.

First, we look at the benefits of having predictive uncertainty. The VIO accuracy is improved substantially. This can be seen by comparing Group 1 (light yellow colored) with Group 2 and 3 (light blue colored).

Table 4.6: Evaluation of VIO variants on indoor 45-degree downward-facing sequences of the UZH-FPV dataset. **Bold** represents the best.

ID	Network	Initialization	Blocks	$\boldsymbol{R}_{\text{meas.}}$ / $k_{\text{var.}}$	Empirical Uncertainty	Avg. Time Cost*↓	RMSE (meter) of Absolute Translation Errors (ATE) ↓					
							2(6.97)	4(6.55)	9(11.23)	12(4.33)	13(7.92)	14(9.54)
1-1	Basic	rand.	4	125.0	None	20.755	5.3298	4.9616	2.6323	3.0957	2.5465	4.3532
1-2	CAHN	Basic	4	35.0	None	-	3.7962	5.5064	2.6658	3.3070	2.8477	7.8038
1-3	Basic	rand.	3	10.0	None	19.658	1.1526	1.0022	0.5777	0.9379	1.2572	**1.6455**
1-4	CAHN	Basic	3	1.5	None	-	1.1091	1.5402	0.4282	0.7613	0.7873	1.6754
2-1	UAHN	rand.	4	1.0	None	23.289	0.4033	0.5121	0.3529	0.5696	0.4171	1.7597
2-2	CUAHN	rand.	4	10.0	None	-	0.3747	0.3529	0.3249	0.6017	0.3884	1.7813
2-3	UAHN	rand.	3	0.5	None	22.043	0.4412	0.5435	0.3796	**0.5390**	0.4288	1.7786
2-4	UAHN+	rand.	3	10.0	None	-	0.4053	0.2965	**0.3145**	0.5518	0.4249	1.7886
2-5	CUAHN	rand.	3	10.0	None	-	0.3496	0.2930	0.3195	0.5954	0.3950	1.7869
3-1	UAHN	Basic	4	30.0	None	-	0.3628	0.3827	0.3779	0.5823	0.4290	1.7732
3-2	CUAHN	Basic	4	15.0	None	-	0.3601	0.3417	0.3225	0.5877	0.4125	1.7706
3-3	UAHN	Basic	3	30.0	None	-	0.3606	0.3614	0.3738	0.5866	0.4329	1.7821
3-4	CUAHN	Basic	3	20.0	None	-	0.3548	0.3144	0.3231	0.5879	0.4189	1.7753
4-1	UAHN	rand.	3	0.5	Ensem. (2†)	26.075	0.4664	0.4497	0.3494	0.5725	0.4156	1.7731
4-2	CUAHN	rand.	3	5.0	Ensem. (2†)	-	0.3493	0.2846	0.3206	0.5865	0.3937	1.7880
4-3	UAHN	rand.	3	0.5	Ensem. (3†)	31.519	0.4264	0.3863	0.3335	0.5749	0.4196	1.7720
4-4	CUAHN	rand.	3	5.0	Ensem. (3†)	-	0.3475	**0.2739**	0.3200	0.5826	0.3985	1.7903
5-1	UAHN	Basic	3	65.0	Ensem. (2†)	-	0.3575	0.3170	0.3825	0.6186	0.4047	1.8008
5-2	CUAHN	Basic	3	50.0	Ensem. (2†)	-	0.3445	0.3179	0.3477	0.6059	0.4035	1.7852
5-3	UAHN	Basic	3	50.0	Ensem. (3†)	-	0.3662	0.3126	0.4199	0.6148	0.4033	1.8052
5-4	CUAHN	Basic	3	30.0	Ensem. (3†)	-	0.3544	0.3072	0.3353	0.5992	0.4042	1.7811
6-1	UAHN	Basic‡	4	10.0	Drop. 5% (16†)	23.605	0.3577	0.3607	0.3623	0.5976	**0.3871**	1.7903
6-2	CUAHN	Basic‡	4	10.0	Drop. 5% (16†)	-	0.3906	0.3437	0.3356	0.6090	0.3945	1.7816

continued on next page

Table 4.6 – continued from previous page

ID	Network	Initialization	Blocks	$\boldsymbol{R}_{\text{meas.}}$ / $k_{\text{var.}}$	Empirical Uncertainty	Avg. Time Cost⋆↓	RMSE (meter) of Absolute Translation Errors (ATE) ↓ 2(6.97)	4(6.55)	9(11.23)	12(4.33)	13(7.92)	14(9.54)
6-3	UAHN	Basic‡	3	10.0	Drop. 5% (16†)	-	**0.3360**	0.2976	0.3761	0.5989	0.3970	1.8003
				5.0			0.3371	0.3139	0.3392	0.5837	0.4066	1.7905
6-4	CUAHN	Basic‡	3	10.0	Drop. 5% (16†)	-	0.3436	0.2915	0.3417	0.5959	0.4067	1.7854
				5.0			0.3479	0.3138	0.3214	0.5843	0.4095	1.7790
6-5	UAHN	Basic‡	2	5.0	Drop. 5% (16†)	19.419	0.3533	0.3015	0.3359	0.5935	0.3964	1.7881
6-6	CUAHN	Basic‡	2	5.0	Drop. 5% (16†)	-	0.3556	0.3044	0.3214	0.5842	0.4057	1.7759
6-7	UAHN	Basic‡	1	5.0	Drop. 5% (16†)	**17.455**	0.5862	0.3612	0.3887	0.6050	crash	6.1602
6-8	CUAHN	Basic‡	1	5.0	Drop. 5% (16†)	-	0.4185	0.3692	0.3432	0.6005	0.4430	8.8627

⋆ Average network inference time consumption, measured on a TX2 processor in Max-P ARM power mode. Networks with the same architectures were only measured once. For instance, the data of 3-2 is omitted since, theoretically, it should be the same as 2-1.

† The number of independent network models in an ensemble or the number of forward passes with dropout.

‡ Only to initialize the convolutional layers. The FC layers are randomly initialized and have dropout layers before them.

Second, we investigate whether it gives better results when empirical uncertainty is also estimated, by comparing Group 2 and 3 (light blue) with Group 4 to 6 (light green). Here, the differences are less pronounced. However, most lowest ATEs are in light green groups, which do have empirical uncertainty. Comparing deep ensembles (Group 4 and 5) with MC-Dropout (Group 6) does not lead to clear conclusions either. Given their similar accuracy, MC-Dropout is preferable due to its lower time consumption. Another phenomenon we observed but is not shown in the table is that more than 16 times of sampling of MC-Dropout produces no noticeable improvement.

Third, we evaluate the effects of content-aware learning. CAHN of Group 1 is initialized by the Basic Model and further trained on the UZH-FPV training set in the same way as the networks in Subsection 4.4.3. In other groups, CUAHN is trained with the aggregated dataset. Compared with UAHN which is trained with the Basic Dataset, CUAHN not only performs content-aware learning but also has seen more in-domain data, *i.e.* the UZH-FPV training set. To see how much content-aware learning alone helps, we trained a master network on the aggregated dataset. Eq. (4.3) instead of Eq. (4.7) is the loss function thus it does not conduct content-aware learning. The student network 2-4 is trained by this master network on the aggregated dataset. The plus sign indicates that it has the bigger training set than other UAHNs. Comparing 2-4 and 2-5 that are trained on the same dataset, 2-5 is trained with the content-aware loss while 2-4 is not. 2-5 wins on four sequences out of six. But, in general, the differences are small. We conclude that the contribution of content-aware learning is small, the same as what is observed in Subsection 4.4.3.

Fourth, we assess the effects of exploiting *a priori* homography for image pre-warping as shown in Fig. 4.3. In Table 4.6, except for the VIO variants with four network blocks, *a priori* homography is exploited for all other variants with less blocks. Running three blocks leads to comparable performance to four blocks with a small computational time gain (~1ms). Reducing the number of blocks further leads to additional time gains. But using only one block results in worse VIO accuracy, as shown in Group 6. Pre-warping with *a priori* homography facilitates VIO accuracy especially in high speed, as illustrated in Fig. 4.10. The 3rd row shows an example of high-speed flight. The 1st network block fails to predict the homography transformation well, which is not corrected by the subsequent blocks. Big estimated variances (top right of the rightmost image) indicate the network's low confidence in its prediction. The 4th row shows how the *a priori* homography initializes the image pair. They are close to good alignment and further refined by the network blocks.

Fifth, we study the influence of the initialization of the student network. Group 2 to Group 5 show no clear influence of this variable. This may mean that the trade-off between more accurate mean value prediction and better variance estimation is equitable and leads to similar VIO accuracy. We notice that the manually tuned parameter $k_{\text{var.}}$ is quite different between the initialization schemes. This tuning may be the partial cause of the similar accuracy. The higher values of $k_{\text{var.}}$ for the "Basic" initialization may compensate to a certain extent for the underestimation of uncertainty, although a simple scaling factor does not intrinsically improve the quality of uncertainty estimation. Since we think proper uncertainty estimation is one of the keys to good generalization and robustness, we have a light preference for random initialization.

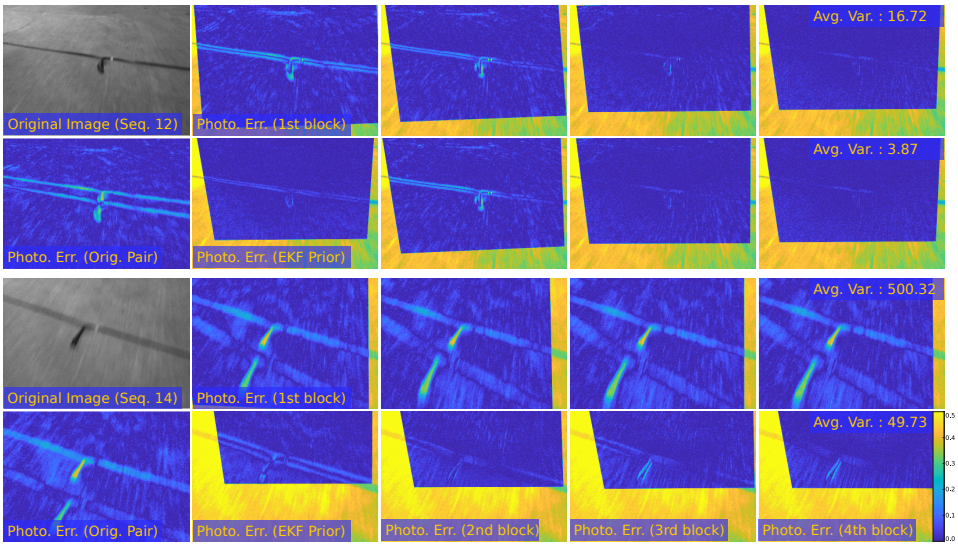


Figure 4.10: The top two rows show an example image pair captured at a relatively slow speed (Seq. 12 of UZH-FPV). The bottom two rows show a high-speed example (Seq. 14). The two image pairs film the same scene. The 1st column shows the original images and the original photometric error maps of the image pairs. The 2nd column shows the photometric error maps of $(\widetilde{I}_{t,1}, I_{t-1})$ or $(\widetilde{I}_{t,\text{prior}}, I_{t-1})$. The 3rd to 5th columns show the photometric error maps of $(\widetilde{I}_{t,i}, I_{t-1})$, i is the index of network block and ranges from 2 to 4. The performance of the network 6-2 in Table 4.6 is shown in the 1st and 3rd rows, and network 6-4 in the 2nd and 4th rows.

4.7.3. ONBOARD DEPLOYMENT FOR FEEDBACK CONTROL

UAHN-VIO, 6-5 of Table 4.6, is deployed onboard an autonomous MAV to produce odometry information required by the close-loop feedback control. Sparse non-planar objects are randomly laid on the planar floor of the flight arena. Images from this environment are not involved in network training. Thanks to the wide distribution of the Basic Dataset and the robustness toward non-planar content of the network as shown in Subsection 4.4.3, theoretically, UAHN-VIO should work in any environment requiring no fine-tuning.

The MAV for flight experiments is a quadrotor equipped with a TX2 processor. An MYNT-EYE visual-inertial sensor is mounted to the bottom of the MAV, downward-facing at 90 degrees with the propeller planes. The image stream and IMU measurement stream are published at 30Hz and 200Hz respectively. UAHN-VIO subscribes to sensor data and publishes the estimated attitude, velocity, and position once it has processed the latest image. So the odometry information has the same frequency as the image stream. We did not compensate for the processing latency* because it is low and stable, as introduced later. The flight controller is a basic proportional-integral-derivative (PID)-based position and velocity controller. It generates thrust and attitude control commands that are sent to Betaflight4 for low-level control.

We tested three kinds of flights, hover*, tracking a circle trajectory*, and shuttle flight between two waypoints. During autonomous flights, the sensor data was recorded for offline replay. During the two-waypoint shuttle flight, the controller was badly tuned on purpose to induce larger motions, resulting in a variety of captured images. The velocity and trajectory plots of the two-waypoint shuttle flight are shown in Fig. 4.13. The link to the flight video is in Subsection 4.9.16.

4.7.4. TIME EFFICIENCY AND PROCESSING LATENCY

We have shown the network inference time consumption in Table 4.6. In the following, we further discuss the detailed time consumption and processing latency of the whole VIO system. The 1st row of Table 4.7 shows the time-consumption-related indicators. We log the time consumption of the three main computing procedures (visual processing, IMU propagation, and EKF updating) of each frame. The mean and variance of the total time consumption are calculated. Besides, we compare the total time cost of processing each frame with the standard time interval of the 30Hz video (33.3ms). The 3rd column from the right shows the percentage of frames that take more than 33.3ms in all frames. In the implementation of both MSCKF and UAHN-VIO, only when a frame has been processed, the filter state at this timestamp is recorded and used to calculate ATE. The Average Processed Frame Rate (2nd column from the right) equals to the number of processed frames divided by the video duration in seconds. This is a metric for how many frames are skipped. The cause of skipping a frame is the limited fixed size of the image buffer. In the implementation, if the VIO processing is too slow and more than five images are waiting for processing in the buffer, the oldest one will be discarded to make room for the new image.

Table 4.7: Time consumption indicators measured on a TX2 processor processing Seq. 2 of UZH-FPV. **Bold** represents the best. Underline marks the frame rates and ATEs valid for accuracy evaluation.

VIO	Image Resolution (pixels)	Num. of Pts / Network Blocks	Histogram Equalization (HE)	Visual Processing Time (ms)	IMU Propagation Time (ms)	EKF Updating Time (ms)	Total Time Mean (ms) ↓	Total Time Variance (ms^2) ↓	Ratio of Long Processing Time (%) ↓	Avg. Processed Frame Rate (fps) ↑	RMSE (m) of ATE ↓
MSCKF	640×480	300	✓	38.37	19.38	8.43	66.19	3.95e4	66.30	18.06	0.3142
MSCKF	320×240	180	✓	24.47	2.29	6.04	32.80	471.61	25.94	23.78	0.4058
MSCKF	640×480	100	✓	26.14	2.47	3.94	32.57	549.43	29.88	23.14	0.3386
MSCKF	640×480	100	✗	22.85	2.09	3.80	28.74	460.15	24.93	24.24	0.3178
MSCKF	640×480	10	✓	15.81	1.66	1.28	18.75	233.38	12.21	26.23	0.4112
MSCKF	320×240	10	✓	**11.57**	1.78	1.48	**14.83**	157.05	7.01	26.23	0.6000
UAHN-VIO	320×224	2	✗	24.00	1.48	0.12	25.61	3.32	**0.44**	26.11	0.3544
UAHN-VIO	320×224	3	✗	27.30	**1.46**	**0.12**	28.89	**2.66**	0.78	26.11	**0.3380**

The MSCKF [4] in Table 4.7 is implemented by [7]. We choose it instead of other SOTA approaches because our C++ implementation is based on the open-sourced code of [7]. Thus the comparison is as fair as possible due to the similarities in code. Besides, the efficiency of MSCKF is also advantageous as a filter-based approach. We change the image resolution, the number of processed feature points, and histogram equalization of the MSCKF because all of them observably affect the time consumption. The data in Table 4.7 is measured on a TX2 processor in the power mode that the VIO approach runs faster. MSCKF computes everything with the CPU. It runs faster in the Max-N power mode. The network of UAHN-VIO runs on GPU and is faster in Max-P ARM mode. According to our observation, it applies to CUDA-accelerated DNNs implemented in PyTorch and LibTorch running on TX2 processors.

The MSCKF of the 2nd row of Table 4.7 has the same settings as the one in Table 4.4. When running on a TX2 processor, the average time cost of processing a frame is around two times the standard time interval. We reduce the number of feature points and downscale the images to lower the average time to just below 33.3ms, as shown in the 3rd and 4th rows. But the variance of the total time is still very big. 25% to 30% frames require a longer time than 33.3ms to process. And there are still frames skipped. The 4th and 5th rows tell us that the better robustness toward motion blur (to be introduced later) brought by histogram equalization comes at an expense of ~3.3ms extra processing time.

The number of feature points is further reduced to only ten as shown in the 6th and 7th rows to minimize the time cost. Only the bottom four rows of Table 4.7 manage to process all the frames. The subtle difference between 26.23 and 26.11 is caused by the different stop time of the two VIO approaches. The frame rates are less than 30 because irregular frame drops exist in the original 30-fps video of UZH-FPV. Comparing ATEs of trajectories at very different frequencies is not informative. So the ATEs in the top four rows are out of the discussion. Comparing the ATEs of UAHN-VIO in Table 4.7 and Table 4.4, they are almost the same on different machines. The accuracy of MSCKF processing ten points is only slightly worse than when processing 300 points as shown in Table 4.4.

It is clear that for both approaches in Table 4.7, visual processing takes most of the time. For MSCKF, the visual processing time is significantly affected by the number of feature points and image resolution. The variance of total time decreases with the mean value, but it is still very big compared with the ones of UAHN-VIO. Even if the number of points is only ten, there are still around 10% of frames whose processing cannot be finished before the next frame comes. In comparison, UAHN-VIO's variance of total time cost is very small, which indicates that the processing time is almost constant for every frame. The very few (less than 1%) frames that take longer processing time for UAHN-VIO are the first several of the video. The cause of it is likely to be library related, *i.e.*, the warm-up phase of the network object of LibTorch. Besides the stable network inference time, UAHN-VIO has low and constant state propagation and updating time cost because of its very simple filter design. Most computation is the network inference on the GPU. As a result, the CPU usage of UAHN-VIO is very low.

Fig. 4.11 shows the processing latency measured on a TX2 processor. It is the time gap between capturing a new image and updating the filter states according to the image. The latency of UAHN-VIO is very stable thanks to the image-content-independent network inference time cost and the simple filter design. The big variation in the latency of MSCKF

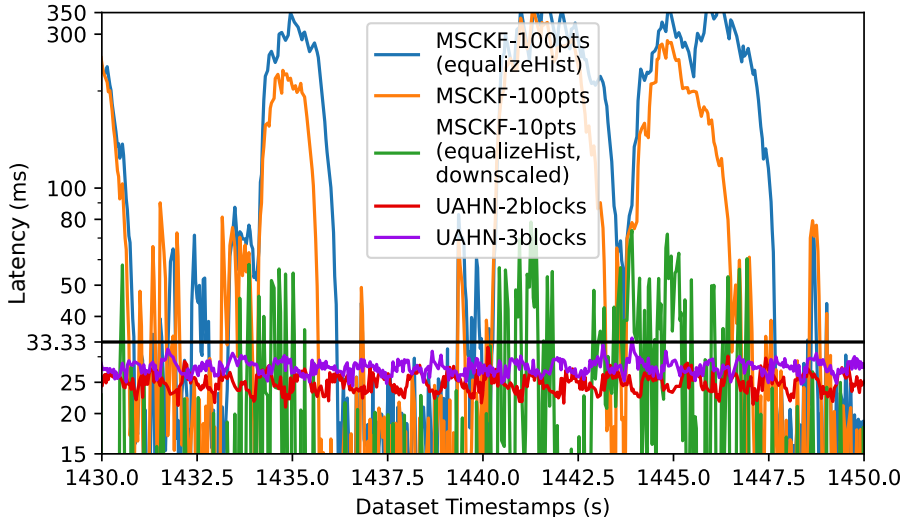


Figure 4.11: Processing latency within 20 seconds of Seq. 2, UZH-FPV dataset.

corresponds to the big variance of time consumption shown in Table 4.7. MSCKF using original images (yellow curve) has smaller latency than its peer that conducts histogram equalization (blue curve). When the number of points is reduced to only ten and the images are downscaled to half resolution (green curve), latency significantly decreases but is still noisy and often beyond 33.3 ms.

The above discussion about the processing time consumption of MSCKF only applies to the current CPU implementation. There are GPU-based implementations for handcrafted feature points such as [58], and learning-based feature points [59–61]. VIO approaches based on feature points adopting such techniques can achieve lower and scene-independent stable latency in their vision front-ends. But the complicated back-ends that utilize the pixel trajectories of the vanished points [4], BA [6], or iterative EKF [12] still require serial computing and the required CPU resources can be considerable and scene-dependent. As far as we know, most traditional VIO approaches only have CPU implementations. So before their GPU versions are widely recognized, CUAHN-VIO has an advantage in processing latency. Besides, it requires small CPU resources and thus allows the deployment of computationally heavy iterative planning and control approaches that run better on CPUs.

4.7.5. ROBUSTNESS TOWARD HIGH-SPEED FLIGHT

A bad effect of high-speed flight on VIO is the huge optical flow in the image plane, especially when the distance to the ground is small. Due to the fixed sample interval and non-neglectable exposure duration of a frame-based camera, visual disparities between consecutive images and motion blur correspondingly grow with optical flow. Regarding big visual disparities, in Fig 4.10, we show a failure case that is solved by the *a priori* homography. Confronting motion blur, in the following, we demonstrate the advantage of using a network as the vision front-end over processing handcrafted feature points. Same

as before, we compare UAHN-VIO and MSCKF. A big percentage of images captured during the two-waypoint shuttle flight (Fig. 4.13) have significant motion blur thus this sequence is used to evaluate the robustness toward blur. The quadrotor MAV maximized its tilt angle to speed up and down. When the speed reached a peak, the MAV rotated to slow down. In this case, the optical flow was caused together by the fastest translation and rotation thus it achieved a peak.

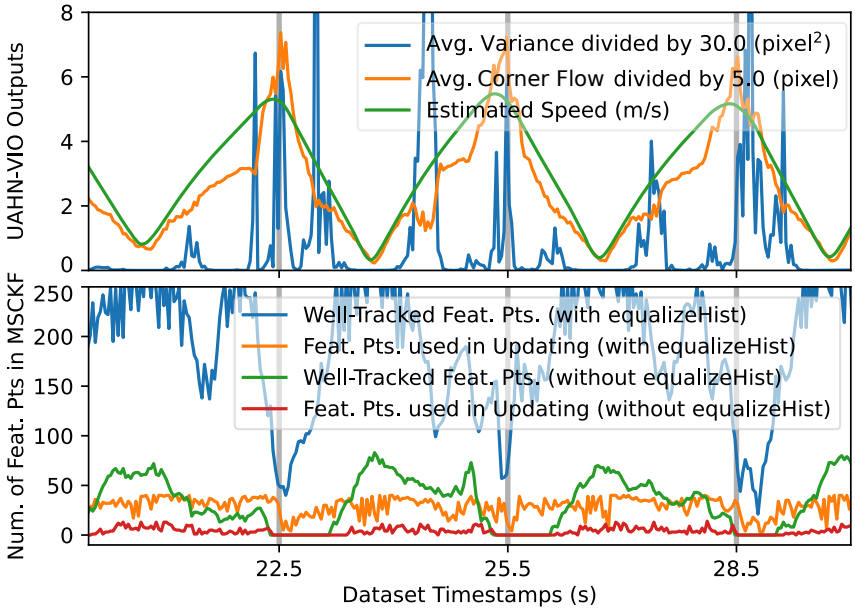


Figure 4.12: The outputs of UAHN-VIO (6-5 of Table 4.6) and MSCKF (2nd row of Table 4.7) processing the two-waypoint shuttle flight sequence. The average of the network-estimated variances of the eight elements of the corner flow and the average of the absolute values of the eight elements of the corner flow are downscaled for better illustration. The well-tracked points are the ones that fulfill the epipolar constraint calculated within a RANSAC scheme.

A well-known problem of handcrafted visual feature points is that detection and tracking become more difficult in the presence of growing motion blur. The bottom subplot of Fig. 4.12 shows the sharp declines in the number of points when optical flow was around its peaks. With histogram equalization, the number of points drops to less than 20% of before. Without histogram equalization, the number drops to and stays at zero until the speed is slow enough. The lack of visual updating causes the MSCKF to drift as shown in the left subplot of Fig. 4.13. Fig. 4.14 shows an image captured when the optical flow is close to a peak. It is too blurry for FAST feature point [62] that relies on local gradients. Histogram equalization increases the image gradients and produces several points without lowering the threshold for feature detection and tracking. But it also induces noise. Most point trajectories only have two frames and very few have three, which indicates that it is hard to keep tracking the already hard-to-detect points. In contrast, despite the severe reduction of local gradients, there are remaining gradients at bigger scales that can

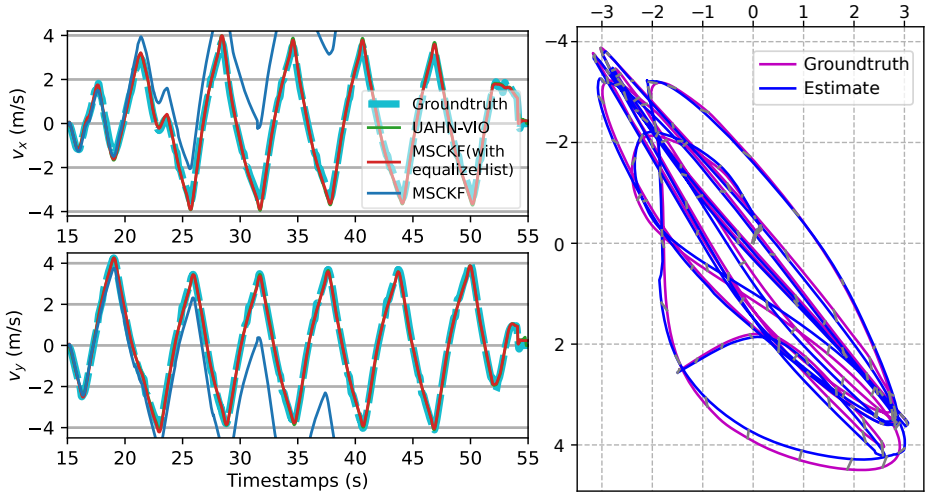


Figure 4.13: UAHN-VIO (6-5 in Table 4.6) and MSCKF (100 feature points) with and without histogram equalization are evaluated on the two-waypoint shuttle flight sequence, running on a TX2 processor. The left subplot shows the velocity expressed in the body frame. The right subplot is the trajectory evaluation of UAHN-VIO plotted by [56]. The groundtruth was recorded by an Optitrack motion capture system. The average and maximum speeds during the flight are respectively 2.87 m/s and 5.41 m/s. The distance to the ground (z-axis) is stabilized at one meter.

be captured by the network. As shown by the photometric error map (bottom right of Fig. 4.14), the network is able to retrieve a reasonable homography transformation that aligns the images well.

A clear phenomenon shown by the top subplot of Fig. 4.12 is that the network is more likely to have big uncertainty estimation at high speed. On most occasions when the network-estimated variance grows, the number of feature points dramatically declines or is already low, which is an indicator of emerging motion blur. Motion blur can be treated as noise in the input of the network and thus it outputs big predictive uncertainty. Fig. 4.10 shows an example that for similar image content with different amounts of blur, based on the reasonable ranges speculated from the photometric error map, the variance is overestimated when the blur is more. Overestimated variances make the relatively accurate mean predictions less trusted in the measurement updates of EKF and thus cause suboptimal results. More examples of uncertainty estimation for blurry images and the positive correlation between speed and estimated uncertainty are available in Subsection 4.9.10 in the Appendix of this chapter for interested readers.

To summarize, VIO approaches that utilize feature points would not necessarily drift because of the lack of points caused by motion blur. Histogram equalization as image pre-processing can significantly increase the number of useful points. Besides, the well-designed VIO back-ends compensate for the effect of fewer points to some extent. We did not use datasets with long periods of ongoing motion blur in this chapter, but expect that these would be more problematic for feature-based approaches. The network suffers from the overestimated uncertainty caused by motion blur. But we have not observed that the

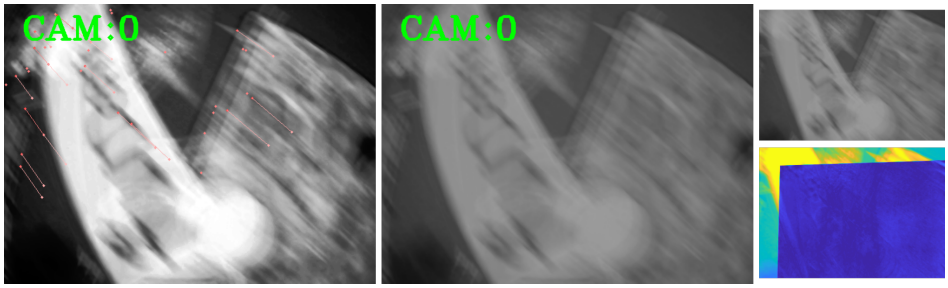


Figure 4.14: An example of the highly blurry images captured during the two-waypoint shuttle flight. Feature point tracking results of MSCKF with and without histogram equalization are shown on the left and middle respectively. These two images are from the visualization module of [7]. In the left image, there are a few point tracking trajectories visualized by small points and thin line segments in red color. They are better to be viewed after zooming in. There is no point tracking trajectory in the middle image. The top right is the undistorted and resized image that is the input of the network. The bottom right is the photometric error map corresponding to the prediction of network 6-5 in Table 4.6. Dark blue indicates small photometric error. The average of the network-estimated variances of the eight elements of the corner flow is 31.12 (pixel^2). Images in this figure are shown in the actual resolutions when being fed into the VIO approaches. The network uses the smaller resolution.

accuracy of mean prediction is noticeably affected. Also because of the higher accuracy of UAHN-VIO than other approaches in most tests in this chapter, we believe that the proposed network has advanced robustness toward motion blur.

4.7.6. POTENTIAL IMPROVEMENTS

About reaching the end of this chapter, we discuss the shortcomings of CUAHN-VIO and ideas that can possibly improve its performance. In this work, we mainly focus on the network that is the vision front-end. Compared with other VIO solutions, the EKF back-end of CUAHN-VIO is very simple and toy-like. The benefit is high time efficiency. However, it lacks a recovery mechanism. One failure case was observed in a real-world flight experiment when the MAV was landing and very close to the ground. The shadow of the body of the MAV was captured by the downward-facing camera and caused an outlier network output. The estimated height was wrongly updated as a minus value, and the VIO crashed. A proper recovery mechanism requires more research.

CUAHN-VIO only updates the current filter state. Keyframe achieved big success in many VIO solutions. It can also be applied to CUAHN-VIO following the similar scheme proposed in [11]. Besides, involving camera poses at multiple time steps into a sliding window may help achieve smoother estimated trajectories.

Note that the CUAHN-VIO introduced in this chapter can only be applied to environments where the planar surface is orthogonal to the gravity vector. But it is possible to extend the application scenario to a slope by improving the EKF back-end. An interested reader can refer to [52] and [11], whose proposed methods estimate the unit normal vector of the plane in the field of view and thus can be applied to a slope.

In the training of CUAHN, the four network blocks are trained to handle the whole ho-

mography transformation. When the *a priori* corner flow propagated by the EKF is utilized for pre-warping, the distribution of the network input can be different from the training set. It can be helpful to fine-tune the network when running the whole VIO on a video. We do not implement this idea mainly due to the concern of overfitting to the scene and motion pattern of the fine-tuning videos.

4.8. CONCLUSIONS

In this chapter, we propose CUAHN-VIO. Its vision front-end is a homography transformation network with uncertainty awareness and the back-end is a simple EKF. Evaluations show its comparable accuracy to SOTA traditional VIO approaches and its advantages in processing latency. The robustness toward motion blur, a trait of learning-based approaches, is observed again in this chapter. The synthetic big-scale training set is proven to enable a homography network to generalize well to the real world. Comparative studies show that, in our context, content-aware learning helps the accuracy to a small extent while uncertainty estimation from the network contributes significantly. Most importantly, different from pursuing better performance through deeper networks and complicated loss functions, this work points out that, without requiring ground truth, a small-size network with a practical training scheme for uncertainty estimation can also stand out.

4.9. APPENDIX

4.9.1. NETWORK ARCHITECTURE

A network block is made of several convolutional layers followed by one or two (only the last block has two) fully-connected layer(s). All the layers except for the output layer of each block are followed by a Leaky ReLU activation with a negative slope of 0.1. There is no normalization layer.

4.9.2. MODEL SIZE

The Basic Model has 5,228,280 parameters. The Master Model is an enlarged version of the Basic Model. It has 6,897,840 parameters. For the student model that outputs predictive uncertainty, a subnetwork of two fully-connected layers with 1,313,032 parameters is added to the 4th block. The total number of parameters of a student model is 6,541,312, among which 3,612,312 belong to the 4th block. These parameters are trained in uncertainty-aware learning. The convolutional decoder network for mask prediction has 1,090,681 parameters that are trained in content-aware learning.

4.9.3. IMPLEMENTATION AND TRAINING

We implement the networks by PyTorch [63] v1.7.1. The training is executed on a server with GTX1080ti GPU and CUDA v10.1. The AdamW [64] optimizer with $\beta = (0.9, 0.999)$ and weight decay $\lambda = 0.01$ is utilized during the 50 training epochs. The batch size is 16. The initial learning rate is 2e-4 and it is divided by 2 after 10, 20, 30, 35, 40, and 45 epochs. The weight parameters are initialized by Kaiming initialization [65] and bias parameters are initialized to zero. The network performs inference on the validation set after each training epoch. The set of parameters that achieve the smallest average validation error are saved

as the best model and evaluated on the testing set. Most networks in this chapter follow the above training procedure, except that Master Model has a smaller initial learning rate (5e-5).

To run a network in a C++ environment, we utilize PyTorch C++ API. We use TorchScript to trace the trained network model in the Python environment to generate a file that can be loaded by C++ code. Further, we use LibTorch in the C++ environment to load the traced model and run the network.

The visual processing time of VIO shown in Table 4.7 of the main body of this chapter is higher than pure network inference (Table 4.6 of the main body of this chapter). The reason is that visual processing includes network inference and other processes supporting it. The processes are image undistortion and resize, and data type conversion between C++ double, OpenCV Mat, and LibTorch tensor. Their total time consumption is around 4.0 to 4.5 milliseconds.

In the training of the content-aware mask based on Laplacian probability distribution, we tried to train the network to directly output $\log b_k$ (Eq. (4.7) of the main body of this chapter) in order to avoid zero values of b_k. However, the training became unstable. Good results were achieved when b_k is the sum of the square of the network output and a small value (2e-6).

4.9.4. COMPARISON OF BASIC HOMOGRAPHY NETWORKS

We compared different network architectures and downsample methods in the early stage of this work. The first architecture utilizes the feature pyramid extractor (FPE) network inspired by the PWC-Net [66]. The FPE is made of three convolutional layers. It processes both images independently and outputs pyramidal feature maps. When the pyramid level is one higher, the height and width of the feature map are halved and the number of channels is doubled. This architecture has 5,466,320 parameters. The second architecture gets the image pyramid by downsampling the images to half/one-fourth/one-eighth of their height and width by average pooling or bilinear interpolation. To achieve a similar model size and inference speed to the first architecture, there are more channels in some of the intermediate tensors in the 3rd block and one more convolutional layer in the 4th block. This architecture has 5,228,280 parameters. We compare these two architectures because they achieved similar performance in translational motion prediction and outperformed others in [16].

Table 4.8: Comparison of basic homography networks.

Network Input	Downsample	Avg. Error (pixel)		Time Cons. (ms)	
		Model 1	Model 2	Python	C++
Feature Maps	FPE	0.409	0.388	28.66	21.66
Pyramidal Images	Avg. Pooling	**0.275**	0.285	28.20	21.16
Pyramidal Images	Bilinear Interp.	0.280	0.281	29.06	21.65
Pyramidal Images	Bilinear Interp.	0.498*	0.501*	21.24*	18.27*

* Directly regress to homography matrix

We trained two models of each architecture to exclude the influence of individual circumstances. The time consumption of network inference is measured on the GPU of an Nvidia Jetson TX2 mobile processor in Max-P ARM power mode. The shown values in the Python column are the averages of one thousand times of inference on the validation set of the Basic Dataset when the batch size is one. The C++ time consumption values are measured when processing the downward-facing Seq. 2 of the UZH-FPV dataset. From Table 4.8 we can conclude that the architecture using pyramidal images is more accurate and faster than its peer utilizing FPE. Downsampling by average pooling is slightly faster than bilinear interpolation. Their accuracy is very close. Besides corner flow, directly regressing to the eight elements of $\boldsymbol{H}$ (the last diagonal element is 1.0) is also implemented for the first three blocks (bottom row of Table 4.8). It does not require DLT solving so it runs faster. But obviously, it is less accurate.

Model 1 of the network using pyramidal images and average pooling in Table 4.8 is selected as the Basic Model in the main body of this chapter.

4.9.5. DIRECT LINEAR TRANSFORMATION (DLT) SOLVER

DLT solver aims at solving the homography matrix $\boldsymbol{H}$ that has 8 degrees of freedom from at least four pairs of corresponding points. As Eq. (4.19) shows, $\boldsymbol{H}$ relates the pixel coordinates in two images of a point lying on a planar surface.

$$\lambda \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \boldsymbol{H} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}, \boldsymbol{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix} \tag{4.19}$$

From Eq. (4.19), we have $\lambda = h_7 u_1 + h_8 v_1 + 1$, then we can form up a linear equation set $\boldsymbol{A}_i \boldsymbol{h} = \boldsymbol{b}_i$, where

$$\begin{aligned} \boldsymbol{A}_i &= \begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -u_2 u_1 & -u_2 v_1 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -v_2 u_1 & -v_2 v_1 \end{bmatrix} \\ \boldsymbol{h} &= \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 \end{bmatrix}^T \\ \boldsymbol{b}_i &= \begin{bmatrix} u_2 & v_2 \end{bmatrix}^T \end{aligned} \tag{4.20}$$

Since we only have the predicted offsets of four corner pixels, i ranges from 1 to 4. Then by forming up the 8×8 matrix $\boldsymbol{A}$ and 8-d vector $\boldsymbol{b}$ from $\boldsymbol{A}_i$ and $\boldsymbol{b}_i$, and calculating the inverse matrix of $\boldsymbol{A}$, $\boldsymbol{h}$ can be solved as $\boldsymbol{h} = \boldsymbol{A}^{-1}\boldsymbol{b}$.

4.9.6. WHY LEARNING REQUIRES A TEACHER NETWORK?

In Subsection 4.4.3 of the main body of this chapter, we introduce how to perform content-aware learning by the predictive uncertainty of photometric matching. A reader may think that the error of homography transformation prediction can be reflected by the photometric matching uncertainty, which is true. Here we explain the reason why we do not use the photometric matching uncertainty map to calculate the uncertainty of homography transformation prediction.

The predictive uncertainty map b_k explains the uncertainty of photometric matching that is affected by both image content and the accuracy of homography transforma-

tion. For a well-trained content-aware network, b_k mainly reflects the content information in complicated scenes as shown in Fig. 4.6 of the main body of this chapter. Because the photometric error induced by the prediction error is much smaller compared to the non-homography content. Since it is unknown whether a pixel belongs to the single non-reflective plane, it is not tractable to decouple the prediction error and the image content. Even if b_k is determined by the prediction error alone, *e.g.*, the pixels on the plane are accurately semantically segmented or the input images have no unfavorable content, it is not clear to us how to calculate the prediction uncertainty from b_k.

Therefore, we follow the common practice of using the negative log-likelihood (NLL) loss to learn the predictive uncertainty. The teacher network is supposed to have high accuracy to provide the learning targets of mean values.

4

4.9.7. COMPARISON OF DIFFERENT OUTPUT DIMENSIONS

In addition to predicting eight different values for each $\log\sigma_n^2$ (8-d variance), inspired by the concise approach utilized in [40], we also implement predicting a generalized single value that represents the overall uncertainty (1-d variance) and duplicating it eight times to fill in all dimensions. It assumes that noise in the input affects all dimensions of $\boldsymbol{f}_4$ equally, neglecting the difference among them. Two types of variance representation under different supervisions are compared in Table 4.9. The 4th block of the student model is randomly initialized. The best model in one training attempt is the one achieving the smallest average imitation error on the validation set. In terms of the prediction accuracy of the mean values, 1-d variance is slightly better. While 8-d variance has obvious advantages in predictive uncertainty as evidenced by lower AUSE and higher inside rate.

Table 4.9: Comparison of predictive uncertainty.

Super-vision	Dim. of Var.	Avg. Error (pixel)	Avg. Imitation Error (pixel)	Avg. Var.* (pixel)	AUSE	Inside Rate (3σ)
GT	1	0.454	0.490	28.16	455.9	96.46
GT	8	0.454	0.489	16.66	370.0	97.63
master	1	0.419	0.329	11.80	445.7	80.83
master	8	0.428	0.336	7.69	357.3	86.96
self	1	0.406	0.210	2.59	756.7	76.93
self	8	0.408	0.208	1.83	565.9	79.08

* The networks predict rare unreasonably big variances. The averages are calculated after removing the 0.1% biggest values.

4.9.8. DIFFICULT TESTING SAMPLES

Fig. 4.15 shows several difficult testing samples from which we explore what leads to high prediction error and high predictive uncertainty. Comparing the Basic Model and the Master Model, the Master Model achieves accurate predictions except for the 1st and 5th images. Significant blur (1st and 2nd columns) and lack of texture (4th and 5th columns) cause big prediction errors of the Basic Model that has less model capacity. The master

model achieves high accuracy on the images of the 2nd and 4th columns, while the errors of the Master-Teach student model are big. It indicates that the student fails to imitate the teacher well. From the appearance of the images, we speculate that the difficulty in imitation is due to the unfavorable image contents from which extracting desired information is hard for a network. That is to say, the big predictive variances are caused by unfavorable image contents. It corroborates the definition of predictive uncertainty, which is that it captures the uncertainty in network output caused by the content-determined observation noise. Based on our observations, the noise is mainly caused by textureless image content and blur.

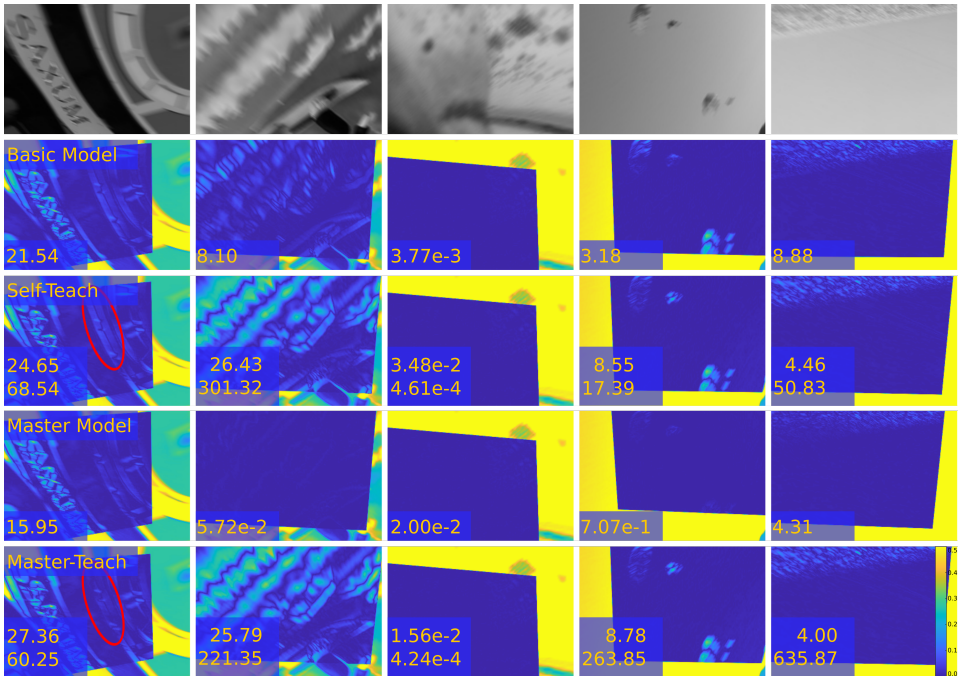


Figure 4.15: Five testing samples from the Basic Dataset that contain unfavorable content: significant blur (1st to 4th) and lack of texture (4th and 5th). 1st row shows one of the input images I_p. The 2nd to 5th rows show the photometric error map of $(I_p, \widetilde{I}_c)$, where dark blue means a low error. The data shown at the bottom left corner of a photometric error map includes the error of the 1st element of $\boldsymbol{f}_{\text{total}}$ (*i.e.* the u component of the upper-left corner) and its predictive variance below it if the network is a student. The 2nd and 4th rows correspond to the teacher networks. The 3rd and 5th rows respectively correspond to the student networks in Self-Teach and Master-Teach. The red ellipses in the 1st column highlight one of their nuances.

4.9.9. CORRELATION BETWEEN PREDICTIVE AND EMPIRICAL UNCERTAINTY

The predictive variance $\sigma^2_{\text{pred.}}$ and empirical variance $\sigma^2_{\text{emp.}}$ are logged and plotted in Fig. 4.16. The idea of estimating empirical uncertainty is to model the distribution of parameters by independent network samples. The randomly initialized networks are more in-

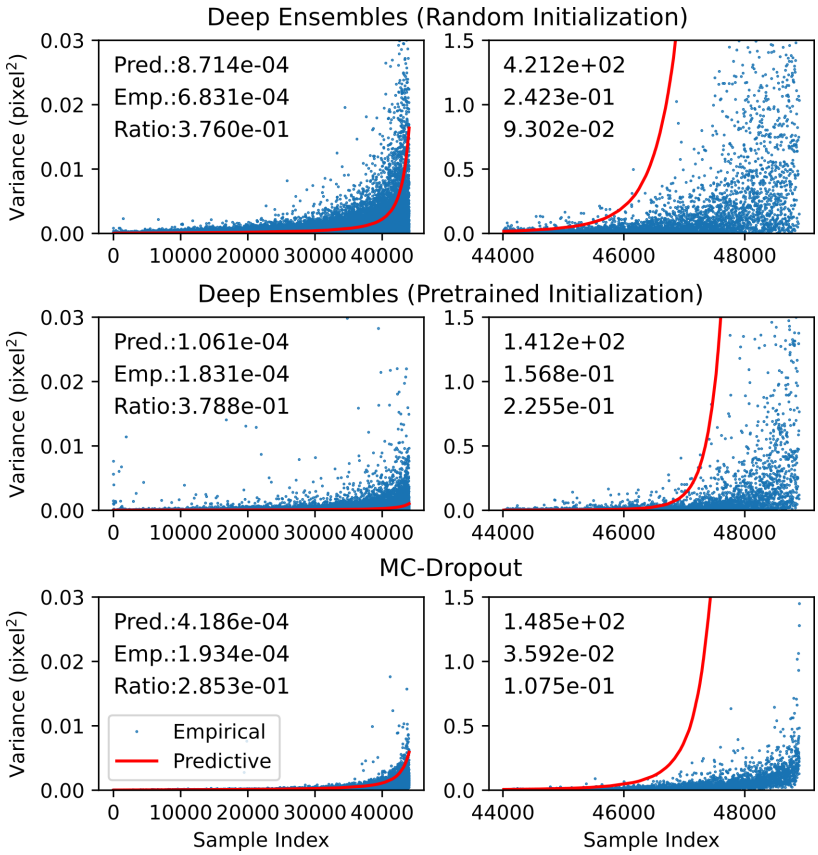


Figure 4.16: Predictive variance $\sigma^2_{\text{pred.}}$ and empirical variance $\sigma^2_{\text{emp.}}$ of three network models in an ensemble. The $\sigma^2_{\text{pred.}}$ - $\sigma^2_{\text{emp.}}$ pairs of each model are sorted according to $\sigma^2_{\text{pred.}}$ and separated into two groups. 90% of the data that has smaller $\sigma^2_{\text{pred.}}$ is shown in the subplots on the left. The rest 10% that has big $\sigma^2_{\text{pred.}}$ is shown in the subplots on the right. Note that the scales of y-axes of the right subplots are much bigger than the left ones. The numbers around the top left corner of each plot are statistical values of the data in the group, which are the averages of $\sigma^2_{\text{pred.}}$, the averages of $\sigma^2_{\text{emp.}}$, and the average ratio of $\sigma^2_{\text{emp.}}$ in the total variance $\sigma^2_{\text{pred.}} + \sigma^2_{\text{emp.}}$. 10% of the total data is plotted to avoid overly dense points.

dependent from each other than the networks initialized by the same parameter set from the Basic Model. For the MC-Dropout networks, due to the initialization of convolutional layers being the same, the independence is compromised. Thus the randomly initialized ensemble has the best knowledge of the parameter distributions and captures the biggest $\sigma^2_{\text{emp.}}$. Comparing $\sigma^2_{\text{emp.}}$ and $\sigma^2_{\text{pred.}}$, in general $\sigma^2_{\text{emp.}}$ has smaller values. For all three models, $\sigma^2_{\text{emp.}}$ accounts for roughly one-third of the total variance for 90% of testing data as shown in the left three subplots.

In the right subplots, with $\sigma^2_{\text{pred.}}$ rapid grows, $\sigma^2_{\text{emp.}}$ also has the trend to increase. The correlation can also be observed in the left subplots. The reason behind the correlation is discussed as follows. A small $\sigma^2_{\text{pred.}}$ means that the network models in the ensemble "think" that their predictions are close to the teacher network with the same input. Thus the networks have similar predictions that lead to small $\sigma^2_{\text{emp.}}$. Similarly, a big $\sigma^2_{\text{pred.}}$ indicates the predictions are far different from the teacher's. Different models may have very different predictions due to the randomness in training. Thus a bigger $\sigma^2_{\text{emp.}}$ is produced. In the perspective of data distribution of the dataset, the testing data leading to big $\sigma^2_{\text{pred.}}$ is the minority. Such images are likely to have unfavorable content that is rare in the training set. Network's unfamiliarity with such contents increases $\sigma^2_{\text{emp.}}$.

Table 4.10: Uncertainty estimation on UZH-FPV.

Average Value	4 (6.55)	2 (6.97)	9 (11.23)	12 (4.33)	13 (7.92)	14 (9.54)
Pred. Var.	92.4	106.8	332.1	41.3	177.5	552.5
Emp. Var.	0.043	0.061	0.108	0.030	0.071	0.124
Ratio Emp. Var.	0.203	0.171	0.120	0.217	0.177	0.139
Pred. Var.	102.4	161.9	438.5	47.6	165.9	415.0
Emp. Var.	0.130	0.191	0.327	0.060	0.120	0.245
Ratio Emp. Var.	0.316	0.275	0.187	0.329	0.232	0.178

In order to gain more insight, we log the $\sigma^2_{\text{pred.}}$ and $\sigma^2_{\text{emp.}}$ on UZH-FPV and show the average values in Table 4.10. The light yellow group shows the results of VIO 6-2 in Table 4.6 of the main body of this chapter. The light green group shows the results of a variant of 4-4 with four blocks and without *a priori*. The three sequences on the left were recorded when there are crowded low objects on the ground. The other three sequences were filmed in the same flight arena but the ground is relatively cleaner from objects. We respectively sort the sequences of the two environments in ascending order of speed. Because the network input image can vary with *a priori* homography transformation, we run all four network blocks and the inputs are the original images. Thus different networks have identical inputs. The data in Table 4.10 tells that both predictive and empirical variances increase with speed. The growth in $\sigma^2_{\text{pred.}}$ is more significant and thus the ratio of $\sigma^2_{\text{emp.}}$ decreases with speed. It is related to the motion blur considered as observation noise by the network, as discussed before. The positive correlation between $\sigma^2_{\text{pred.}}$ and $\sigma^2_{\text{emp.}}$ is observable. For both $\sigma^2_{\text{pred.}}$ and $\sigma^2_{\text{emp.}}$, deep ensembles produce bigger values, implying more comprehensive uncertainty estimation that contributes to better AUSE and Inside Rate, as shown in

Fig. 4.9 in the main body of this chapter.

Snapshot Ensemble [67] learns multiple models faster than deep ensembles. Since it shares the disadvantage of high inference time cost with deep ensembles (limited by our current implementation), it is not separately discussed. The approach proposed in [68] that enhances model diversity in the ensemble by altering hyperparameters is not discussed for the same reason.

4.9.10. NETWORK UNCERTAINTY AND VELOCITY

As supplementary to Fig. 4.12 in the main body of this chapter, Fig. 4.17 illustrates the statistically positive correlation between the magnitude of optical flow and network uncertainty estimation. We do not know the exact magnitude of optical flow but it is determined by the speed divided by the distance to the ground and the rotation rate. Since the rotation in the dataset is mostly slow, the height-scaled speed can serve as an approximate equivalent of the magnitude of optical flow. For both sequences, estimated variance is more likely to have a big value when the speed is high. In the slow sequence, most estimated variances have very small values thanks to the clear and sharp images. While in the fast sequence where blurry images are common, the positive correlation between the estimated variance and height-scaled speed is obvious. Fig. 4.18 shows cases where the network has big variance estimations.

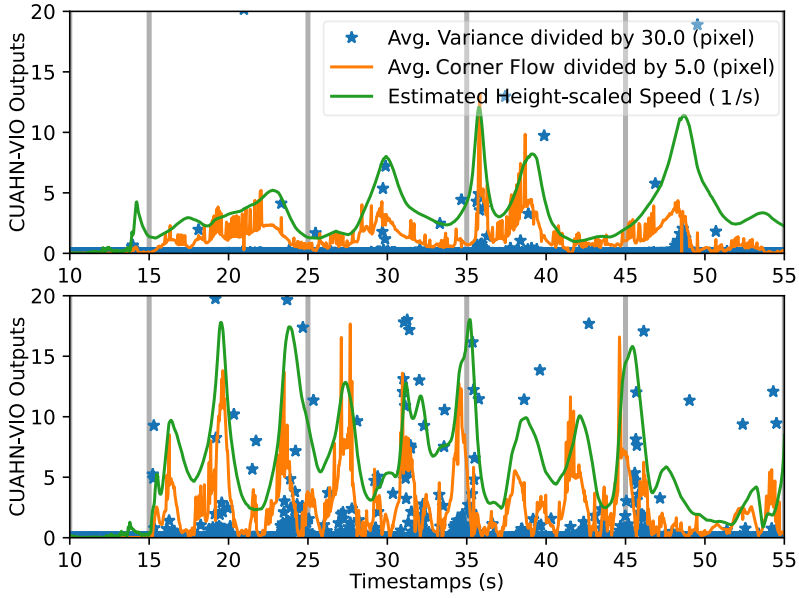


Figure 4.17: The outputs of CUAHN-VIO (6-4 of Table 4.6 in the main body of this chapter) evaluated on a slow sequence (Seq. 12) and a fast sequence (Seq. 14) of UZH-FPV dataset. Both sequnces were collected in the same environment.

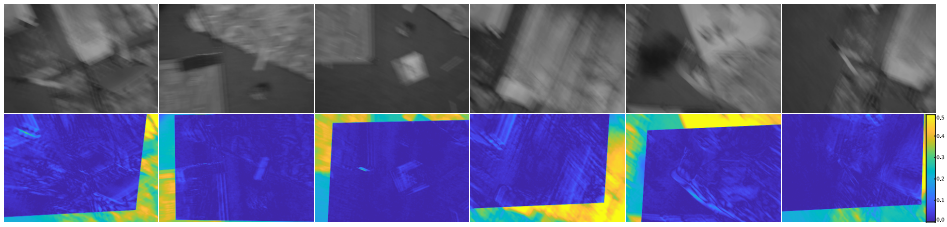


Figure 4.18: Example images from the high-speed two-waypoint shuttle flight sequence. The top row shows the original images and the bottom row shows the photometric error maps of the image pairs warped according to the network predictions of homography transformation. Significant motion blur causes big predictive uncertainty. The average values of the variance estimations of the eight elements of the corner flow are respectively 394.99, 106.41, 59.80, 183.90, 190.22, 221.72 (pixel^2) for the six examples.

4.9.11. UAHN-VIO FOR FEED-BACK CONTROL

Two of the common practices for VIO to obtain better performance in autonomous flights are 1) publishing the estimated states in higher frequency, *e.g.*, IMU measurement frequency, and 2) propagating the states to the current time using the IMU measurements that come after the latest image to compensate for the image processing latency. Since we aim to evaluate the VIO instead of pursuing the overall performance of autonomous flight, the above methods are not deployed in this chapter. Nevertheless, thanks to the short and almost constant processing time of UAHN-VIO, the MAV performed stable controlled flights. The RMSE of the ATE of the estimated trajectory by UAHN-VIO is 0.1521 in the two-waypoint shuttle flight (Fig. 4.13 of the main body of this chapter). The trajectories of autonomous flights of tracking a circle trajectory and fixed-point hovering are shown in Fig. 4.19. The RMSEs of the ATEs are respectively 0.2775 and 0.0155.

4.9.12. EKF STATE PROPAGATION

In our implementation, we use the simple zeroth-order integration Eq. (4.21). When propagating the states from t to $t+1$, we use the average of the IMU measurements sampled at the two time points Eq. (4.22).

$$
\begin{aligned}
\boldsymbol{p}_{t+1} &= \boldsymbol{p}_t + \Delta t(-[\overline{\boldsymbol{\omega}}]_\times \boldsymbol{p}_t + \boldsymbol{v}_t), \\
\boldsymbol{v}_{t+1} &= \boldsymbol{v}_t + \Delta t(-[\overline{\boldsymbol{\omega}}]_\times \boldsymbol{v}_t + \overline{\boldsymbol{a}} + \boldsymbol{R}^{-1}(\boldsymbol{q}_t)\boldsymbol{g}), \\
\boldsymbol{q}_{t+1} &= \boldsymbol{q}_t \otimes \boldsymbol{q}\{\overline{\boldsymbol{\omega}}\Delta t\} = \boldsymbol{q}_t \otimes \begin{bmatrix} \cos(\|\overline{\boldsymbol{\omega}}\|\Delta t/2) \\ \frac{\overline{\boldsymbol{\omega}}}{\|\overline{\boldsymbol{\omega}}\|}\sin(\|\overline{\boldsymbol{\omega}}\|\Delta t/2) \end{bmatrix}, \\
\boldsymbol{b}_{a,t+1} &= \boldsymbol{b}_{a,t}, \quad \boldsymbol{b}_{g,t+1} = \boldsymbol{b}_{g,t}, \\
\boldsymbol{f}_{j,t+1} &= \boldsymbol{f}_{j,t} - \Delta t(\boldsymbol{I} - (\boldsymbol{c}_j + \boldsymbol{f}_{j,t})\boldsymbol{e}_z^T)\boldsymbol{H}(\boldsymbol{c}_j + \boldsymbol{f}_{j,t})
\end{aligned}
\tag{4.21}
$$

$$
\overline{\boldsymbol{a}} = 0.5 \cdot (\hat{\boldsymbol{a}}_t + \hat{\boldsymbol{a}}_{t+1}), \quad \overline{\boldsymbol{\omega}} = 0.5 \cdot (\hat{\boldsymbol{\omega}}_t + \hat{\boldsymbol{\omega}}_{t+1}) \tag{4.22}
$$

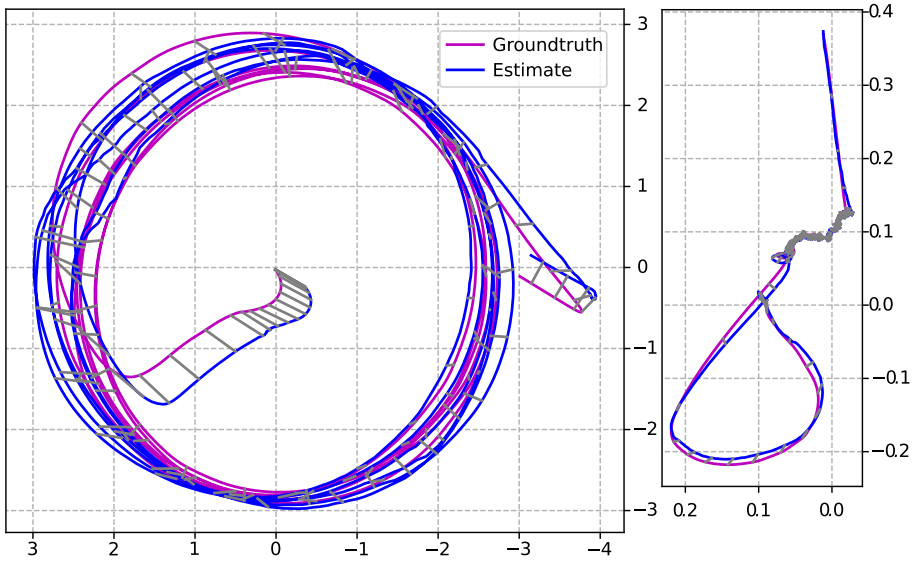


Figure 4.19: Trajectories of circle trajectory tracking and fixed-point hovering.

4.9.13. *a Priori* HOMOGRAPHY

As shown in Fig. 4.10 of the main body of this chapter, the variance estimations for both image pairs are smaller when utilizing *a priori* homography. In the 2nd row, a noticeable phenomenon is that the photometric error map of $(\widetilde{I}_{t,2}, I_{t-1})$ is bigger than $(\widetilde{I}_{t,\text{prior}}, I_{t-1})$. The reason can be that, in training, the networks are trained to infer the whole homography transformation from the original image pairs. The 2nd block often handles bigger disparities in training than the ones between $\widetilde{I}_{t,\text{prior}}$ and I_{t-1} and thus tends to output bigger values.

4.9.14. ITERATIVE EKF

Inspired by [12, 52], we tried the iterative EKF scheme. After the first EKF update, the corner flow is updated. The image pair is then warped according to the updated corner flow and input to the network again to estimate the remaining visual disparities. And then EKF is updated once more according to the outputs of the second network inference. However, this scheme fails to improve accuracy.

4.9.15. PARAMETER TUNING OF SOTA VIO APPROACHES

For OpenVINS, we use the open-sourced code of it and modified all the parameters mentioned in the report† from the original values in the launch file. For MSCKF, we use the same code as OpenVINS by setting the number of SLAM points to zero.

At the beginning of each sequence of UZH-FPV, the MAV was swung by the human operator and then put on the ground. For sequences No. 2, 9, and 12, when we started ROVIO at the very beginning of the sequences, the estimated trajectories showed huge

†https://rpg.ifi.uzh.ch/uzh-fpv/ICRA2020/reports/open-vins.pdf

drifts. By changing the starting time of ROVIO to after the MAV is swung and before taking off, we obtained much better results on these sequences. We tried the IMU noise parameters given by UZH-FPV and the parameters in the ROVIO's configuration file for EuRoC. After several attempts, we found that the combination of IMU noise densities for EuRoC and the IMU random walk parameters given by UZH-FPV yielded good results.

We modified the source code of VINS-Fusion to get odometry output at frame rate. We observed occasional big drifting in the estimated trajectories of VINS-Fusion. So we ran VINS-Fusion several times on each sequence until we got two or more good estimated trajectories, and used the best one in comparison. We used the IMU noise parameters given by UZH-FPV for VINS-Fusion. We observed that multiplying the given white noise standard deviation of the gyroscope and the accelerometer by 0.005 yields better results.

For LARVIO, we use the parameters mentioned in the report[‡]. The IMU noise parameters are the same as the ones in EuRoC.yaml of the open-sourced code. The VIO started at the beginning of Seq. 9. But for other sequences, the VIO started after the hand-swinging part.

ORB-SLAM3 initializes a map when the drone slowed down to make turns and the optical flow was relatively small. But it lost tracking very soon when the drone speeded up. We tried to lower the thresholds for the feature extraction but it made little difference.

4.9.16. SUPPLEMENTARY MATERIALS

The code developed for CUAHN-VIO will be open-sourced at https://github.com/tudelft/CUAHN-VIO upon the publication of this chapter. A video of VIO runtime performance is available at https://youtu.be/_NgDkgON-nE.

REFERENCES

[1] Y. Xu and G. C. de Croon, *Cuahn-vio: Content-and-uncertainty-aware homography network for visual-inertial odometry,* arXiv preprint arXiv:2208.13935 (2022).

[2] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, *Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset,* in *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019) pp. 6713–6719.

[3] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, *Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam,* IEEE Transactions on Robotics **37**, 1874 (2021).

[4] M. Li and A. I. Mourikis, *High-precision, consistent ekf-based visual-inertial odometry,* The International Journal of Robotics Research **32**, 690 (2013).

[5] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, *Keyframe-based visual–inertial odometry using nonlinear optimization,* The International Journal of Robotics Research **34**, 314 (2015).

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

[‡]https://fpv.ifi.uzh.ch/wp-content/uploads/sourcenova/uni-comp/2019-2020-uzh-fpv-benchmark/submissions/20/details.pdf

[7] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, *Openvins: A research platform for visual-inertial estimation,* in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2020) pp. 4666–4672.

[8] X. Qiu, H. Zhang, and W. Fu, *Lightweight hybrid visual-inertial odometry with closed-form zero velocity update,* Chinese Journal of Aeronautics (2020).

[9] 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.

[10] J. Engel, V. Koltun, and D. Cremers, *Direct sparse odometry,* IEEE transactions on pattern analysis and machine intelligence **40**, 611 (2017).

[11] S. Zhong and P. Chirarattananon, *An efficient iterated ekf-based direct visual-inertial odometry for mavs using a single plane primitive,* IEEE Robotics and Automation Letters **6**, 486 (2020).

[12] 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 **36**, 1053 (2017).

[13] 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**, 249 (2016).

[14] W. Wang, Y. Hu, and S. Scherer, *Tartanvo: A generalizable learning-based vo,* in *Conference on Robot Learning* (PMLR, 2021) pp. 1761–1772.

[15] B. Wagstaff, E. Wise, and J. Kelly, *A self-supervised, differentiable kalman filter for uncertainty-aware visual-inertial odometry,* arXiv preprint arXiv:2203.07207 (2022).

[16] Y. Xu and G. C. de Croon, *Cnn-based ego-motion estimation for fast mav maneuvers,* in *2021 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2021) pp. 7606–7612.

[17] 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**, 18 (2015).

[18] S. Wang, R. Clark, H. Wen, and N. Trigoni, *End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks,* The International Journal of Robotics Research **37**, 513 (2018).

[19] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, *Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem,* in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31 (2017).

[20] C. M. Parameshwara, G. Hari, C. Fermüller, N. J. Sanket, and Y. Aloimonos, *Diffposenet: Direct differentiable camera pose estimation,* arXiv preprint arXiv:2203.11174 (2022).

[21] L. Han, Y. Lin, G. Du, and S. Lian, *Deepvio: Self-supervised deep learning of monocular visual inertial odometry using 3d geometric constraints,* in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2019) pp. 6906–6913.

[22] C. Li and S. L. Waslander, *Towards end-to-end learning of visual inertial odometry with an ekf,* in *2020 17th Conference on Computer and Robot Vision (CRV)* (IEEE, 2020) pp. 190–197.

[23] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, *Unsupervised learning of depth and ego-motion from video,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017) pp. 1851–1858.

[24] Z. Yin and J. Shi, *Geonet: Unsupervised learning of dense depth, optical flow and camera pose,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018) pp. 1983–1992.

[25] Y. Chen, C. Schmid, and C. Sminchisescu, *Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera,* in *Proceedings of the IEEE International Conference on Computer Vision* (2019) pp. 7063–7072.

[26] R. Mahjourian, M. Wicke, and A. Angelova, *Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018) pp. 5667–5675.

[27] J. Bian, Z. Li, N. Wang, H. Zhan, C. Shen, M.-M. Cheng, and I. Reid, *Unsupervised scale-consistent depth and ego-motion learning from monocular video,* Advances in neural information processing systems **32** (2019).

[28] R. Li, S. Wang, Z. Long, and D. Gu, *Undeepvo: Monocular visual odometry through unsupervised deep learning,* in *2018 IEEE international conference on robotics and automation (ICRA)* (IEEE, 2018) pp. 7286–7291.

[29] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers, *D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020) pp. 1281–1292.

[30] Y. Almalioglu, M. Turan, M. R. U. Saputra, P. P. de Gusmão, A. Markham, and N. Trigoni, *Selfvio: Self-supervised deep monocular visual–inertial odometry and depth estimation,* Neural Networks **150**, 119 (2022).

[31] A. Geiger, P. Lenz, and R. Urtasun, *Are we ready for autonomous driving? the kitti vision benchmark suite,* in *2012 IEEE conference on computer vision and pattern recognition* (IEEE, 2012) pp. 3354–3361.

[32] 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 **35**, 1157 (2016).

[33] A. Kendall and Y. Gal, *What uncertainties do we need in bayesian deep learning for computer vision?* in *Proceedings of the 31st International Conference on Neural Information Processing Systems* (2017) pp. 5580–5590.

[34] D. A. Nix and A. S. Weigend, *Estimating the mean and variance of the target probability distribution,* in *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, Vol. 1 (IEEE, 1994) pp. 55–60.

[35] R. M. Neal, *BAYESIAN LEARNING FOR NEURAL NETWORKS*, Ph.D. thesis, University of Toronto (1995).

[36] B. Lakshminarayanan, A. Pritzel, and C. Blundell, *Simple and scalable predictive uncertainty estimation using deep ensembles,* Advances in Neural Information Processing Systems **30** (2017).

[37] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson, *A simple baseline for bayesian uncertainty in deep learning,* Advances in Neural Information Processing Systems **32**, 13153 (2019).

[38] Y. Gal and Z. Ghahramani, *Dropout as a bayesian approximation: Representing model uncertainty in deep learning,* in *international conference on machine learning* (PMLR, 2016) pp. 1050–1059.

[39] E. Kaufmann, M. Gehrig, P. Foehn, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, *Beauty and the beast: Optimal methods meet learning for drone racing,* in *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019) pp. 690–696.

[40] M. Klodt and A. Vedaldi, *Supervising the new with the old: learning sfm from sfm,* in *Proceedings of the European Conference on Computer Vision (ECCV)* (2018) pp. 698–713.

[41] V. Peretroukhin, B. Wagstaff, and J. Kelly, *Deep probabilistic regression of elements of so (3) using quaternion averaging and uncertainty injection.* in *CVPR Workshops* (2019) pp. 83–86.

[42] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia, *On the uncertainty of self-supervised monocular depth estimation,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020) pp. 3227–3237.

[43] E. Ilg, O. Cicek, S. Galesso, A. Klein, O. Makansi, F. Hutter, and T. Brox, *Uncertainty estimates and multi-hypotheses networks for optical flow,* in *Proceedings of the European Conference on Computer Vision (ECCV)* (2018) pp. 652–667.

[44] A. Kendall and R. Cipolla, *Modelling uncertainty in deep learning for camera relocalization,* in *2016 IEEE international conference on Robotics and Automation (ICRA)* (IEEE, 2016) pp. 4762–4769.

[45] D. DeTone, T. Malisiewicz, and A. Rabinovich, *Deep image homography estimation,* arXiv preprint arXiv:1606.03798 (2016).

[46] F. Erlik Nowruzi, R. Laganiere, and N. Japkowicz, *Homography estimation from image pairs with hierarchical convolutional networks,* in *Proceedings of the IEEE International Conference on Computer Vision Workshops* (2017) pp. 913–920.

[47] H. Le, F. Liu, S. Zhang, and A. Agarwala, *Deep homography estimation for dynamic scenes,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020) pp. 7652–7661.

[48] T. Nguyen, S. W. Chen, S. S. Shivakumar, C. J. Taylor, and V. Kumar, *Unsupervised deep homography: A fast and robust homography estimation model,* IEEE Robotics and Automation Letters **3**, 2346 (2018).

[49] J. Zhang, C. Wang, S. Liu, L. Jia, N. Ye, J. Wang, J. Zhou, and J. Sun, *Content-aware unsupervised deep homography estimation,* in *European Conference on Computer Vision* (Springer, 2020) pp. 653–669.

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

[51] S. Baker, A. Datta, and T. Kanade, *Parameterizing homographies,* in *Technical Report CMU-RI-TR-06-11* (2006).

[52] S. Zhong and P. Chirarattananon, *Direct visual-inertial ego-motion estimation via iterated extended kalman filter,* IEEE Robotics and Automation Letters **5**, 1476 (2020).

[53] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, *Digging into self-supervised monocular depth estimation,* in *Proceedings of the IEEE/CVF international conference on computer vision* (2019) pp. 3828–3838.

[54] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation,* in *International Conference on Medical image computing and computer-assisted intervention* (Springer, 2015) pp. 234–241.

[55] J. Sola, *Quaternion kinematics for the error-state kalman filter,* arXiv preprint arXiv:1711.02508 (2017).

[56] Z. Zhang and D. Scaramuzza, *A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry,* in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2018) pp. 7244–7251.

[57] Z. Teed and J. Deng, *Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras,* Advances in neural information processing systems **34**, 16558 (2021).

[58] S. Heymann, K. Müller, A. Smolic, B. Froehlich, and T. Wiegand, *Sift implementation and optimization for general-purpose gpu,* (2007).

[59] A. B. Laguna and K. Mikolajczyk, *Key. net: Keypoint detection by handcrafted and learned cnn filters revisited,* IEEE Transactions on Pattern Analysis and Machine Intelligence (2022).

[60] D. DeTone, T. Malisiewicz, and A. Rabinovich, *Superpoint: Self-supervised interest point detection and description,* in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (2018) pp. 224–236.

[61] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, *Superglue: Learning feature matching with graph neural networks,* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020) pp. 4938–4947.

[62] E. Rosten and T. Drummond, *Machine learning for high-speed corner detection,* in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9* (Springer, 2006) pp. 430–443.

[63] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, *Pytorch: An imperative style, high-performance deep learning library,* in *Advances in neural information processing systems* (2019) pp. 8026–8037.

[64] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization,* arXiv preprint arXiv:1711.05101 (2017).

[65] K. He, X. Zhang, S. Ren, and J. Sun, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,* in *Proceedings of the IEEE international conference on computer vision* (2015) pp. 1026–1034.

[66] 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.

[67] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, *Snapshot ensembles: Train 1, get m for free,* arXiv preprint arXiv:1704.00109 (2017).

[68] Z. Y. Ding, J. Y. Loo, V. M. Baskaran, S. G. Nurzaman, and C. P. Tan, *Predictive uncertainty estimation using deep learning for soft robot multimodal sensing,* IEEE Robotics and Automation Letters **6**, 951 (2021).

5

LIGHTWEIGHT VISUAL-INERTIAL ODOMETRY AND MONOCULAR DEPTH LEARNED FROM SELF-SUPERVISED STRUCTURE-FROM-MOTION

Learning-based approaches for the visual navigation of micro air vehicles (MAVs) are playing increasingly important roles. Among them, ego-motion estimation and obstacle detection are two essential capacities for safe autonomous flights. In this chapter, we propose efficient solutions based on a pose network and a lightweight monocular depth network, respectively. A supervision signal for training the two networks is the simultaneous self-supervised learning of pose and depth, i.e., self-supervised structure-from-motion (SfM), which has big-size networks and an iterative mechanism to pursue higher accuracy in both pose and depth predictions. For ego-motion estimation, the pose network predicts the relative motion of the camera between consecutive image frames and estimates the prediction uncertainty. The distributions of visual measurements are then fused with inertial measurements by an extended Kalman filter (EKF), making up efficient visual-inertial odometry (VIO). For training the lightweight monocular depth network, we utilize the supervision signals of both the photometric error based on view synthesis and the knowledge distillation from a deeper depth network. Evaluations show that the proposed VIO and the monocular depth network have passable accuracy, given their lightweight designs that fit the requirements of real-time onboard processing.

Parts of this chapter have been accepted by the International Micro Air Vehicle Conference and Competition (2023), and the rest parts have been accepted by the 22nd World Congress of the International Federation of Automatic Control (2023).

5.1. INTRODUCTION

Ego-motion estimation of micro air vehicles (MAVs) is essential for autonomous flight and has been a major research topic in the robotics domain. Monocular cameras are often used for ego-motion estimation of lightweight MAVs in environments without stable GPS signals, because of their small size, lightweight, low energy consumption, and rich environmental information captured. After years of research, ego-motion estimation approaches that use vision in conjunction with inertial measurement unit (IMU), *i.e.*, visual-inertial odometry (VIO), are widely applied to MAVs. Many mainstream VIO approaches [1–3] use visual feature points as the vision processing front-end. Assuming the scene is stationary, information about the camera motion can be obtained by tracking the two-dimensional (2-d) pixel locations of the same feature point in multiple temporally consecutive images. Therefore, extracting a sufficient number of feature points and then tracking or matching them accurately across frames is the key to accurate ego-motion estimation. However, because of the dependence on appearance consistency and image gradient, visual feature points are susceptible to varying illumination and image blur.

Besides feature points, researchers have been exploring deep learning for visual ego-motion estimation. Learning-based approaches train one or multiple artificial neural network(s) (ANN) to predict the relative pose [4–6] between temporally consecutive images or visual correspondences that encode ego-motion, *i.e.*, optical flow [7, 8]. A pose network can be trained by the ground-truth relative pose in supervised learning. But the acquisition of accurate camera position and orientation in the real world often requires external sensors such as motion capture systems (MCSs) [9]. So the scenes of captured images for training are restricted to the range of the stationary motion capture sensors. Thus it limits the size of the training dataset and deteriorates the generalization capacity of the pose network. In contrast, self-supervised learning is an attractive alternative for freeing the need for ground truth. The first self-supervised learning scheme of a pose network was proposed in [4], where a monocular depth network is trained simultaneously. It is sometimes referred to as self-supervised structure-from-motion (SfM) in the literature. According to the predicted dense depth and relative camera pose, a virtual view can be synthesized by reprojecting the world points into the image frame. The photometric error between the synthesized view and the actually captured image is the main supervision signal. This scheme has been further developed by many follow-up works and is adopted in this chapter. A more detailed description is provided in Section 5.2.

It has been observed that ANNs show better robustness towards unfavorable conditions such as illumination change, motion blur, and dynamic scenes [5, 6, 8]. There are learning-based approaches [7, 8, 10] achieving better accuracy on the EuRoC MAV datasets [11] than traditional approaches [12, 13]. But general challenges exist in learning-based approaches. Firstly, many works train and test on the same dataset without considering the network's generalization capability. Secondly, achieving high accuracy requires a significant amount of computational power, making such approaches unsuitable for computationally constrained MAVs. Thirdly, many works cannot be directly deployed on MAVs due to various limitations. For example, translational motion lacks scale in [8]. The scale is ambiguous in the case of self-supervised learning with monocular videos [14–16]. The scale is also unknown in the case of self-supervised VIO that processes IMU measurements with an ANN [17]. Because the loss function has no constraint on the scale, the

metric-scale accelerometer measurements can be scaled arbitrarily by the ANN. Another problem in using an IMU network is that, when the supervision signal only regularizes the relative pose [17, 18], the gravity information encoded in the accelerometer measurements is not exploited. So the gravity direction remains unknown, though it is observable for a monocular visual-inertial system [19].

The first topic of this chapter aims to address the above-mentioned challenges by introducing a learning-based VIO. It can be directly deployed on an MAV due to its notable characteristics, such as efficiency in real-time processing on a mobile processor, translational motion estimation with metric scale, estimating the direction of gravity, and generalizing to new environments. The vision front-end is a pose network that infers the relative rotation and translation of the camera from an image pair. It is capable of estimating the prediction uncertainty. In addition, the network's training can be conducted with or without ground-truth labels.

In addition to ego-motion estimation, obstacle avoidance is essential for autonomous MAVs. For an MAV equipped with a monocular camera, there are multiple ways to avoid obstacles. A visual simultaneous localization and mapping (V-SLAM) system can incrementally build a map of the environment [20, 21]. A trajectory planner can output collision-free trajectories based on the map for the MAV to track [22]. This kind of scheme fully utilizes available visual information about the environment and is more likely to produce optimal flight trajectories. But a significant amount of computational power is required for mapping and planning. Instead of explicitly mapping the environment, there are learning-based approaches predicting the desired motion command based on the input image. The end-to-end navigation network proposed in [23] was trained by human driving actions and can directly generate an obstacle-avoiding navigation policy. The disadvantage of end-to-end networks is that it is hard to understand the strategy they follow, which makes it harder to predict the strategy's capability of generalizing to unknown environments.

A simpler way is to consider only the obstacles in the current field of view. Optical flow can reflect the environmental structure, assuming stationary surroundings. The authors of [24] focused on substantially reducing neural network size while retaining sufficient performance for dense optic flow estimation. The network was applied to obstacle avoidance, by simply comparing the left and right halves of the flow map. The disadvantage of using optic flow for obstacle avoidance is that sufficient motion is essential and that obstacles in the direction of motion are hard to detect. Another way is resorting to a dense depth map that reflects the distances of the environmental structures relative to the camera. The pixels with small depths point out the nearby obstacles. Pixel depth can be obtained using a stereo camera and a stereo matching algorithm, or an RGB-D (depth) camera. Thanks to recent developments in monocular depth estimation with deep learning, an accurate dense depth map can be inferred from a single image.

There are not only works pursuing higher accuracy with large networks [14, 25] but also works focusing on networks that are small in size and have fast inference speed, aiming to be deployed on *memory-and-computation-constrained* mobile devices. Many of them adopt self-supervised learning [26–28] jointly with a pose network. To better train a small-size network, learning from a larger network via knowledge distillation is also a popular choice [29–32]. The second topic of this chapter is investigating how to use the

self-supervised learning framework of camera pose and monocular depth in training a lightweight depth network that achieves the highest possible accuracy. The network is trained on a small training set collected in a cluttered environment and is expected to navigate an MAV safely. Note that, unlike the pose network, the depth network studied does not pursue cross-dataset generalization capacity.

In summary, focusing on general 3-d environments, we study both the camera pose network and the monocular depth network in this chapter. Both networks are required to be lightweight to reduce the demand for computational power. Self-supervised learning schemes for pose and depth [14–16] are adopted to train teacher networks with high prediction accuracy. On the basis of the teacher networks, lightweight pose and depth networks are trained as student networks. Our main contributions are:

1. We explore the way to an efficient learning-based monocular VIO based on a pose network and an extended Kalman filter (EKF). Specifically, the pose network predicts the relative pose between the newest image pair and estimates its prediction uncertainty. The effects of three elements of the pose network design are evaluated and analyzed. They are the supervision signal (with *vs.* without ground truth), the scale of translational motion prediction (with metric scale *vs.* direction only), and with *vs.* without fine-tuning on the target dataset.

2. The training of a lightweight depth network combines two supervision signals, which are respectively self-supervised learning using pose-dependent and reprojection-based view synthesis, and knowledge distillation from a bigger-size network. Better accuracy is achieved than using either one of them solely.

The structure of this chapter is as follows. In Section 5.2, the self-supervised learning scheme for pose and depth teacher networks is introduced. In Section 5.3, the training of the uncertainty-aware pose network is first introduced, followed by the description of the EKF-based VIO. The evaluations of the VIO include cross-dataset generalization, and the ablation study comparing learning from the teacher networks with supervised learning by ground-truth labels. After ego-motion estimation, the topic moved to monocular depth. In Section 5.4, we compare and combine the two learning schemes for the lightweight depth network on three datasets, each collected in a single environment. Lastly, the conclusions are summarized in Section 5.5.

5.2. TEACHER NETWORKS

5.2.1. IMPROVED SELF-SUPERVISED SFM

The captured scenes usually overlap for two temporally consecutive images in a video. Camera motion can be deduced by the correspondences of pixels filming the overlapping scene. In self-supervised SfM, the pose network predicts the relative pose (rotation and translation) $T_{t\to s}$ between the source image I_s and the target image I_t, using the image pair of I_s and I_t as input. The depth network predicts D_t, the pixel-wise depth map of I_t, using the single image I_t as input. A point cloud can be established from D_t. The appearance of each point is the corresponding pixel intensity. Given the relative pose $T_{t\to s}$, the expression of the 3-d coordinates of the point cloud in the camera frame of I_s can be

obtained. Reprojecting the point cloud to the image plane of I_s and warping I_s by differentiable bilinear interpolation [33] produce a synthesized image $\widetilde{I}_s$. When both D_t and $T_{t\to s}$ are accurate, assuming the scene is static, brightness is constant, and the pixels are visible in both images, the pixel intensities of $\widetilde{I}_s$ reflect the observation by I_s. Further, assuming the scene appearance is constant, *i.e.*, constant illumination and camera sensitivity and a Lambertian scene whose brightness does not vary with the observer's angle of view, $\widetilde{I}_s$ and I_t should have the same pixel intensities. So, under the assumptions mentioned above, maximizing the similarity between $\widetilde{I}_s$ and I_t leads to accurate D_t and $T_{t\to s}$. The loss function adopted by [14] is shown in Eq. (5.1).

$$L_{\text{photo}} = \frac{1}{|V|}\sum_{k\in V}\frac{\alpha}{2}(1-\text{SSIM}(I_t,\widetilde{I}_s)_k) + (1-\alpha)\cdot|I_{t,k}-\widetilde{I}_{s,k}| \tag{5.1}$$

V denotes the set of all pixels. The loss function combines the $L1$ loss of pixel-wise photometric error and the structured similarity index measure (SSIM) loss, with $\alpha = 0.85$. Eq. (5.1) is referred to as the reprojection-based loss in some works and in this chapter. Since there is no constraint on the scale, training with Eq. (5.1) leads to D_t and the translational part of $T_{t\to s}$ having the same ambiguous scale.

For a general 3-d scene in the real world, the assumptions required by learning correct pose and depth using the loss function Eq. (5.1) are not always met. A correct loss can be established on a pixel of I_t only when the world point captured by the pixel is static, not occluded in I_s, and its appearance keeps constant in I_s and I_t. In addition, the camera needs to have enough translational motion to constrain D_t. Otherwise, arbitrary depth predictions lead to the same loss. The strategy proposed in Monodepth2 [14] filters out pixels that violate some of the assumptions. It uses more than one I_s to construct multiple photometric error maps with I_t. For a pixel, only the smallest photometric error in the error maps is minimized in network training. It can avoid minimizing the incorrect photometric error derived from a world point occluded in one of the source images. For instance, if a point p is occluded in $I_{s,i}$ but visible in I_t and $I_{s,i+1}$, and D_t and $T_{t\to s}$ are close to being accurate, then the corresponding photometric difference $(\widetilde{I}_{s,i}, I_t)_p$ is very likely to be big and bigger than $(\widetilde{I}_{s,i+1}, I_t)_p$. So in the training, we should exclude $(\widetilde{I}_{s,i}, I_t)_p$ and only minimize $(\widetilde{I}_{s,i+1}, I_t)_p$. Another strategy is to exclude pixels at whose locations the intensities have little change in consecutive images. The constant intensity can be caused by objects moving at the same velocity as the camera, a stationary camera, or a textureless image region. The first situation violates the static-scene assumption. In the second situation, arbitrary depth predictions lead to the same loss. In the third situation, a relative pose prediction that is close to zero and an arbitrary depth prediction will produce a small photometric error of a pixel that is far away from image textures. To summarize, the photometric errors of such pixels cannot validly constrain the network predictions. Given the higher robustness brought by the auto-masking strategies, we select Monodepth2 as the base of training the teacher networks and develop our code based on its open-sourced code.

Eq. (5.1) utilizes scene appearance consistency. Bian *et al.* [15] proposed another loss term that constrains the depth predictions. It is based on the geometry consistency of the 3-d positions of the world points. The depth maps D_t and D_s of both images I_t and I_s are predicted by the depth network. The loss can be constructed in three steps. First,

from D_t, pixel locations, and camera intrinsics, the 3-d positions $[x_t, y_t, D_t]^T$ of the world points p_t captured by the pixels of I_t can be obtained. $[x_t, y_t, D_t]^T$ is expressed in the camera frame of I_t. Second, by $T_{t\to s}$, we can calculate the position of p_t expressed in the camera frame of I_s, as $[{}_sx_t, {}_sy_t, {}_sD_t]^T = SE3(T_{t\to s}) \cdot [x_t, y_t, D_t]^T$. The depth map ${}_sD_t$ of p_t relative to the camera of I_s is thus obtained. A pixel at the location (u, v) of ${}_sD_t$ encodes the depth of the p_t captured by the pixel (u, v) of I_t. ${}_sD_t$ is a function of D_t and $T_{t\to s}$. Third, according to $[{}_sx_t, {}_sy_t, {}_sD_t]^T$, p_t is reprojected into the image plane of I_s and warp D_s to synthesize a depth map $\widetilde{D}_s$. A pixel at the location (u, v) of $\widetilde{D}_s$ encodes the depth of the p_t captured by the pixel (u, v) of I_t. $\widetilde{D}_s$ is calculated by warping D_s using $[{}_sx_t, {}_sy_t, {}_sD_t]^T$, so it is a function of D_s, D_t, and $T_{t\to s}$. In the two depth maps ${}_sD_t$ and $\widetilde{D}_s$, the two pixels at the same location encode two depth predictions of the same world point. The loss derives from the difference between ${}_sD_t$ and $\widetilde{D}_s$. It explicitly encourages the scale of D_t, D_s, and $T_{t\to s}$ to be the same within a training batch. For a sequence of images, this means that D_i, D_{i-1}, and $T_{i\to i-1}$ have the same scale, D_{i+1}, D_i, and $T_{i+1\to i}$ have the same scale. With training going on, scale consistency can propagate to the entire sequence. For the relative pose predictions, the integration will lead to globally scale-consistent trajectories. For depth, scale consistency allows directly comparing depth across images, which gives more information in obstacle detection. This chapter adopts this geometry consistency loss L_{geo}, as shown in Eq. (5.2). The expression of L_{geo} is taken from [15].

In Chapters 3 and 4, multiple network blocks run consecutively to refine the network prediction incrementally. Similarly, in [34] and [16], the pose network inference is conducted for multiple times by iterative view synthesis that relies on the depth map prediction. In the first run of the iteration, the inputs of the pose network are the original I_s and I_t. With the first pose prediction $T_{t\to s}$ and D_t, the warped source image $\widetilde{I}_{s,1}$ can be synthesized. With network predictions becoming increasingly accurate as the training goes on, I_t is expected to have smaller visual disparities with $\widetilde{I}_{s,1}$ than with I_s. In the second run, the pose network infers the relative pose $T_{t\to s,1}$ from the inputs $\widetilde{I}_{s,1}$ and I_t. $T_{t\to s,1}$ is then composed to $T_{t\to s}$. The composed transformation is used to synthesize the $\widetilde{I}_{s,2}$ that is an input of the third iteration, and so on. The reason why this iterative mechanism is adopted by us is the higher accuracy in pose prediction. Instead of predicting the total transformation by a single inference, the pose network incrementally refines the relative pose prediction by inferring from input images that are more and more similar, making the problem more tractable. Because the depth map D_t is involved in synthesizing the input image of the pose network, the final pose prediction is a product of both the pose network and the depth network.

In summary, the self-supervised learning of the teacher networks enhances Eq. (5.1) by the auto-masking strategies from Monodepth2, the geometry consistency loss, and the iterative pose predictions, as shown in Eq. (5.2).

$$
\begin{aligned}
L &= \lambda_{\text{geo}} \cdot L_{\text{geo}} + \sum_{i=1}^{N} \lambda_i \cdot L_{\text{photo},i} \\
L_{\text{photo},i} &= \frac{1}{|V|} \sum_{k\in V} \mathscr{M}_{\text{auto}} \left(\frac{\alpha}{2}(1 - \text{SSIM}(I_t, \widetilde{I}_{s,i})_k) + (1-\alpha)\cdot |I_{t,k} - \widetilde{I}_{s,i,k}|\right) \\
L_{\text{geo}} &= \frac{1}{|V|} \sum_{k\in V} \frac{|{}_sD_{t,k} - \widetilde{D}_{s,k}|}{{}_sD_{t,k} + \widetilde{D}_{s,k}}
\end{aligned}
\tag{5.2}
$$

N is the total number of iterations. The geometry consistency loss L_{geo} is computed only once by the ${}_sD_t$ and $\widetilde{D}_s$ that are computed using the final pose prediction after all iterations. Hyperparameters λ_{geo} and λ_i are scalar weights of loss terms. $\mathscr{M}_{\text{auto}}(\cdot)$ stands for applying the auto-masking strategies of Monodepth2 to the pixels. The smoothness loss for depth prediction is the edge-aware smoothness. It is implemented by the open-sourced code of Monodepth2. We omit its expression in Eq. (5.2). An interested reader can refer to [14].

Instead of predicting depth maps at four scales as Monodepth2 does, we predict a single depth map at the full resolution. It results in higher accuracy in both depth and pose predictions. To better learn how to handle fast motion, our implementation uses a bigger temporal step in loading image snippets. There is 25% possibility to load I_{i-2}, I_i, and I_{i+2} and 75% possibility to load the neighboring images I_{i-1}, I_i, and I_{i+1}. Because the relative translational motion between I_i and I_{i+2} is usually bigger than I_i and I_{i+1}, it is better for learning depth. When learning depth is the first priority, as in Section 5.4, the possibility of using the bigger temporal step is increased to 75%.

5.2.2. DATASETS AND NETWORK TRAINING

TartanVO [8] and Droid-SLAM [7] are two of a few works that achieve good performance of cross-dataset generalization. Both of them use TartanAir [35] dataset for training. It is a large-scale dataset that has 369 image sequences collected in 18 photo-realistic simulation environments with the presence of dynamic objects, varying illumination, diverse motion patterns, and various weather conditions. We take most of the images (296,899) for training and a small number (1,522) for in-domain testing. Testing images are randomly sampled from all the sequences.

A pose network infers relative pose from raw images, so camera intrinsics affect the prediction. Such a pose network requires the input image to always have the specific image resolution and camera intrinsics. Therefore, in cross-dataset testing and real-world deployment, the images need to be preprocessed to be the same as training images in terms of resolution and camera intrinsics. In this chapter, input images to a pose network are required to be grey-scale and have a resolution of 192 × 352 and intrinsics $f_x = 176$, $f_y = 176$, $c_x = 176$, $c_y = 96$. This setting fits both the training set (TartanAir) and the ego-motion estimation testing set (the EuRoC MAV dataset [11]). Images of TartanAir lose pixels near the top and bottom edges after being transformed to this setting. But if we transform the images of EuRoC to have the same resolution and camera intrinsics as TartanAir, the characteristics of the lens and optic sensor of the camera used in collecting the EuRoC dataset lead the images to have black curving edges. This phenomenon reveals the lack of cross-camera generalization of pose networks. A solution based on preprocessing the input optical flow map is proposed in [35], while the pose network in this paper uses raw images as input. So the cross-camera generalization is not solved.

The networks in this section are trained on the entire TartanAir dataset for 100 epochs. We utilized the 1 cycle learning rate policy [36] with the maximum learning rate of 2.5e-4 in self-supervised learning. The architectures of the pose and depth networks are based on ResNet-18, the same as Monodepth2. The number of iterations $N = 3$. The weights of the loss for each iteration are $\lambda_0 = 0.25$, $\lambda_1 = 0.5$, and $\lambda_2 = 1.0$. The weight for geometry consistency $\lambda_{\text{geo}} = 0.2$. The batch size is 32.

The purpose of training the teacher networks using the entire TartanAir dataset is to have as-accurate-as-possible pose predictions. They are used as the learning targets of an uncertainty-aware pose network, as is introduced in Section 5.3. We compare the pose prediction accuracy of the iterative teacher networks trained by the enhanced loss function (Eq. (5.2)) with the original Monodepth2 in Table 5.1. The accuracy metric is the average of the norms of error vectors, $e = \frac{1}{N}\sum||\boldsymbol{v}_{\text{GT}} - \boldsymbol{v}_{\text{Pred.}}||_2$. The rotation is expressed by axis-angle and implemented in the open-sourced code of Monodepth2. The predicted translation and the ground-truth translation are normalized to 3-d unit vectors before calculating the error because the scale of the translation prediction is ambiguous. When the translation predictions are used as the learning targets of the uncertainty-aware pose network, they are also normalized.

Table 5.1: Average pose prediction errors on TartanAir testing set and EuRoC testing set. "FT" indicates that the networks are fine-tuned on the EuRoC training set based on the parameters trained on the TartanAir training set. **Bold** represents the best.

Network	TartanAir Rot.	TartanAir Trans.	EuRoC Rot.	EuRoC Trans.
ours	**1.99e-03**	**8.77e-02**	3.09e-03	2.39e-01
ours-FT	-	-	**1.73e-03**	**1.62e-01**
Monodepth2	2.21e-03	1.23e-01	4.08e-03	2.89e-01
Monodepth2-FT	-	-	3.18e-03	2.48e-01

As shown in Table 5.1, we perform an in-domain test on the TartanAir testing set and an out-of-domain test on a part of the EuRoC MAV dataset [11]. There are 11 sequences in the EuRoC dataset. The EuRoC training set is made of all sequences except for *Vicon Room 1 03*, *Vicon Room 2 03*, and *Machine Hall 05*. These three sequences are used for testing. From the first and third rows of Table 5.1, we can see that our networks have smaller average errors in pose predictions on both tests.

We finetune the networks trained on the TartanAir dataset on the training set of EuRoC and test them on the three testing sequences, as shown in the second and fourth rows of Table 5.1. Although TartanAir is a large-size dataset with a wide distribution, we still observe that fine-tuning brings obvious improvement in pose prediction accuracy. In Fig. 5.2, we show the trajectories estimated by integrating the pose network predictions on the *Machine Hall 05* sequence of the EuRoC dataset as an example of the improvement from fine-tuning. The estimated trajectories of sequence *Machine Hall 05* by the four networks in Table 5.1 are shown in Fig. 5.1.

5.3. EFFICIENT VIO BASED ON POSE NETWORK AND EKF

The previous section introduces the methodology to train the self-supervised teacher network. As already explained in the introduction, the teacher networks are not suitable for the ego-motion estimation of an MAV. Specifically, the pose prediction of the teacher networks relies on one forward pass of the depth network and multiple forward passes of the pose network, which require considerable computational power. ,In addition, the translational predictions have an ambiguous scale and the gravity direction is not estimated.

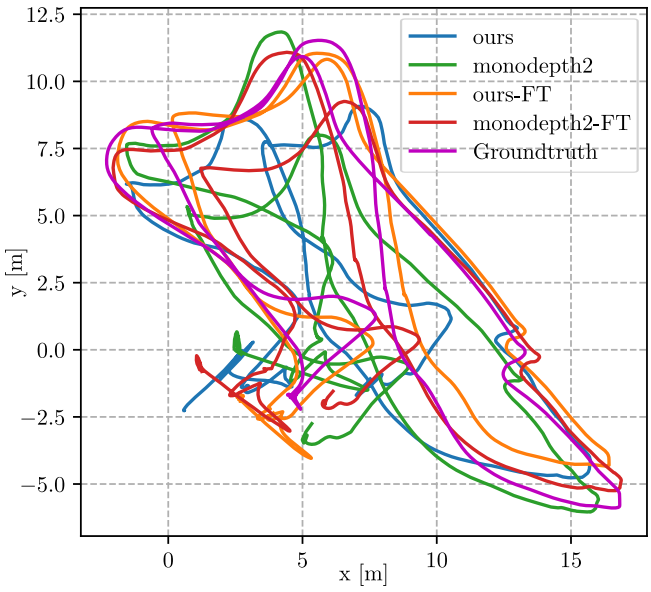
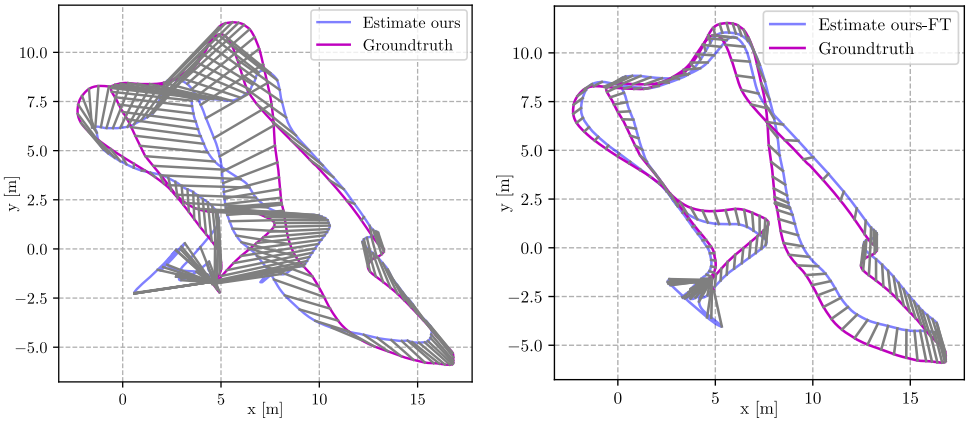


Figure 5.1: The trajectories are obtained by integrating relative pose predictions on the *Machine Hall 05* sequence of the EuRoC dataset. They are aligned with the ground-truth trajectory by the 7-DoF (degree-of-freedom) *Sim3* alignment (3-d rotation, 3-d translation, 1-d scale) using an open-sourced toolkit [37].



(a) Before Fine-tuning.

(b) After Fine-tuning.

Figure 5.2: Predicted trajectories by the networks trained by the loss function Eq. (5.2). Comparison between with and without fine-tuning on the EuRoC training set. These two trajectories are also shown in Fig. 5.1.

In this section, we introduce a computationally efficient VIO that can estimate metric-scale translational motion and the gravity direction. Its vision front-end is a pose network. It performs a single forward pass for a pose prediction. The back-end is a simple EKF. We train several front-end pose networks using two supervision signals and compare the resulting VIO accuracy. They are, respectively, the ground-truth poses and the teacher networks' pose predictions. When using the teacher networks, the front-end pose network is a student pose network, and the whole learning pipeline does not require ground-truth labels of the pose. Different from the teacher networks, the front-end pose network not only predicts the relative pose but also estimates the uncertainty of its prediction. So it is called an uncertainty-aware pose network.

5.3.1. UNCERTAINTY-AWARE POSE NETWORK

The architecture of the pose network is based on the pose network used by Monodepth2. It has an encoder based on ResNet-18. We modify the architecture of the decoder part to enable it for uncertainty estimation. The output tensor of the last convolutional layer is the input to two fully connected (FC) subnetworks. One FC network predicts the mean value, and the other predicts the *aleatoric* uncertainty that reflects the noise inherent in the network input. It is referred to as predictive uncertainty in Chapter 4 and this chapter. The two FC networks have the same architecture of two FC layers. There is a 5% dropout for the input of each FC layer, to estimate the *epistemic* uncertainty using MC-Dropout [38]. *Epistemic* uncertainty captures the ignorance about the ideal network that maps noiseless input to the desired output. It is called empirical uncertainty in Chapter 4 and this chapter. The total uncertainty is the sum of predictive uncertainty and empirical uncertainty. The uncertainty estimation of the front-end uncertainty-aware pose network follows the same methodologies as in Chapter 4. An interested reader can read Chapter 4 for more details. We introduce the most important steps in the following.

Treating the observed pose as a sample from a Gaussian distribution, the loss function for learning predictive uncertainty is the negative log-likelihood (NLL) loss, as shown in Eq. (5.3). This loss function is used for the same purpose in [39].

$$L_{\text{NLL}} = \sum_{n=1}^{6} \frac{1}{2\sigma_{n,\text{pred.}}^2} \| T_n - \mu_n \|^2 + \frac{1}{2} \log(\sigma_{n,\text{pred.}}^2) \tag{5.3}$$

μ_n is the prediction of the mean value of the relative pose vector. $\sigma_{n,\text{pred.}}^2$ is the variance prediction corresponding to the predictive uncertainty. n indexes over the six dimensions of relative pose (3-d rotation and 3-d translation). T_n denotes the learning target of the mean prediction μ_n. T_n can be the ground-truth relative pose or the pose prediction of the trained teacher networks. The pose network can be trained to predict translational motion with the metric scale. It can also be trained to predict normalized translation expressed by a unit vector that indicates the motion direction without scale. We implement both of them and compare them in Subsection 5.3.3. When the pose network is trained to predict translational direction, its prediction of the mean translation is normalized. The translational part of T_n is normalized accordingly.

MC-Dropout [38] requires multiple forward passes to sample from the distributions of network parameters. m indexes over the forward passes. We set the number of MC-Dropout sampling M=8. After each forward pass, a mean prediction $\mu_{n,m}$ is obtained, as

well as a variance prediction of predictive uncertainty $\sigma^2_{\text{pred.},n,m}$. As Eq. (5.4) shows, the variance of empirical uncertainty $\sigma^2_{\text{emp.},n}$ is calculated empirically from the multiple mean predictions $\mu_{n,m}$. The total variance σ^2_n is the sum of the empirical variance $\sigma^2_{\text{emp.},n}$ and the average of predictive variances $\sigma^2_{\text{Avg., pred.},n}$.

$$\begin{aligned} \sigma_n^2 &= \sigma^2_{\text{Avg., pred.},n} + \sigma^2_{\text{emp.},n} \\ \sigma^2_{\text{Avg., pred.},n} = \frac{1}{M}\sum_{m=1}^{M} \sigma^2_{\text{pred.},n,m}, &\quad \sigma^2_{\text{emp.},n} = \frac{1}{M}\sum_{m=1}^{M} (\mu_{n,m} - \mu_n)^2 \\ \mu_n &= \frac{1}{M}\sum_{m=1}^{M} \mu_{n,m} \end{aligned} \tag{5.4}$$

5.3.2. EKF-BASED BACK-END

The VIO back-end is an EKF. It is a simplified variant of the EKF-based back-end of the robocentric VIO [40]. The robocentric VIO keeps a window of the previous camera poses in the state vector. Our simplified variant only estimates the relative pose between the current IMU frame and the local frame of reference. The EKF state vector is defined as

$$\boldsymbol{x} := \left[{}^{R_k}\boldsymbol{p}_G, \quad {}^{R_k}_{G}\boldsymbol{q}, \quad \boldsymbol{g}_{R_k}, \quad {}^{I_t}\boldsymbol{p}_{R_k}, \quad {}^{I_t}_{R_k}\boldsymbol{q}, \quad \boldsymbol{v}_{I_t}, \quad \boldsymbol{b}_a, \quad \boldsymbol{b}_g \right]. \tag{5.5}$$

I_t is the current IMU frame at time t. When a new image has been captured, the new reference frame R is set to be the same as the I_t at the time. R_k is the current reference frame. It is the kth reference frame since the VIO is initialized. G stands for the global frame. It is the first reference frame R_0, *i.e.*, the IMU frame when the first image is captured after the initialization of the VIO. ${}^{R_k}\boldsymbol{p}_G$ is a translation vector pointing from the origin of G to the origin of R_k, expressed in R_k. It is about the global position of R_k. ${}^{I_t}\boldsymbol{p}_{R_k}$ is a translation vector pointing from the origin of R_k to the origin of I_t, expressed in I_t. It is about the local position of I_t relative to R_k. ${}^{R_k}_{G}\boldsymbol{q}$ is the Hamilton quaternion reflecting the relative rotation between G and R_k. ${}^{I_t}_{R_k}\boldsymbol{q}$ reflects the relative rotation between R_k and I_t. $\boldsymbol{g}_{R_k}$ indicates the gravity vector expressed in R_k. $\boldsymbol{v}_{I_t}$ is the translational velocity of the IMU expressed in I_t. $\boldsymbol{b}_a$ and $\boldsymbol{b}_g$ are respectively the additive bias on accelerometer and gyroscope.

As shown in Eq. (5.6), IMU measurements are modeled as the sum of the desired actual value ($\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{\omega}}$), additive bias, and white Gaussian noise ($\boldsymbol{w}_a$ and $\boldsymbol{w}_g$).

$$\boldsymbol{a}_m = \hat{\boldsymbol{a}} + \boldsymbol{b}_a + \boldsymbol{w}_a, \quad \boldsymbol{\omega}_m = \hat{\boldsymbol{\omega}} + \boldsymbol{b}_g + \boldsymbol{w}_g \tag{5.6}$$

$$\begin{aligned} {}^{I_t}\dot{\boldsymbol{p}}_{R_k} &= -[\hat{\boldsymbol{\omega}}]_\times \cdot {}^{I_t}\boldsymbol{p}_{R_k} + \boldsymbol{v}_{I_t} + \boldsymbol{w}_{\boldsymbol{p}}, \\ \dot{\boldsymbol{v}}_{I_t} &= -[\hat{\boldsymbol{\omega}}]_\times \cdot \boldsymbol{v}_{I_t} + \hat{\boldsymbol{a}} + \boldsymbol{R}({}^{I_t}_{R_k}\boldsymbol{q})^T \cdot \boldsymbol{g}_{R_k}, \\ {}^{I_t}_{R_k}\dot{\boldsymbol{q}} &= \frac{1}{2}\,{}^{I_t}_{R_k}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \hat{\boldsymbol{\omega}} \end{bmatrix}, \\ \dot{\boldsymbol{b}}_a &= \boldsymbol{w}_{\boldsymbol{b}_a}, \quad \dot{\boldsymbol{b}}_g = \boldsymbol{w}_{\boldsymbol{b}_g}. \end{aligned} \tag{5.7}$$

Eq. (5.7) shows the IMU-driven state dynamics ($\dot{\boldsymbol{x}}$). $[\hat{\boldsymbol{\omega}}]_\times$ represents the skew-symmetric matrix associated with $\hat{\boldsymbol{\omega}}$. $\boldsymbol{w}_{\boldsymbol{p}}$ is the process noise in position integration. $\boldsymbol{R}({}^{I_t}_{R_k}\boldsymbol{q})$ is a

transformation function from ${}^{I_t}_{R_k}\boldsymbol{q}$ to *SO3* rotation matrix that maps a vector expressed in I_t to its expression in R_k. $\otimes$ denotes quaternion product. We utilize the techniques introduced in [41] for quaternion-related calculation.

The filter states in Eq. (5.7) are propagated by the IMU measurements until a new image I_{k+1} is captured. The pair of I_{k+1} and the previous image I_k are the inputs of the uncertainty-aware pose network that is introduced in Subsection 5.3.1. The network outputs $\boldsymbol{\mu_t}$, $\boldsymbol{\sigma_t^2}$, $\boldsymbol{\mu_\theta}$, and $\boldsymbol{\sigma_\theta^2}$. They are the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$ of translation $\boldsymbol{t}$ and rotation $\boldsymbol{\theta}$, respectively.

$$\boldsymbol{z_{\mu_t}} = R_I^C \cdot (\boldsymbol{t}_{IC} + {}^{I_t}\boldsymbol{p}_{R_k} - \boldsymbol{R}({}^{I_t}_{R_k}\boldsymbol{q})^T \cdot \boldsymbol{t}_{IC}) + \boldsymbol{w_t}, \quad \boldsymbol{z_{\mu_\theta}} = R_I^C \cdot \boldsymbol{\theta}({}^{I_t}_{R_k}\boldsymbol{q}) + \boldsymbol{w_\theta}. \tag{5.8}$$

The measurement equations are shown in Eq. (5.8). R_I^C is the rotation matrix from the IMU frame to the camera frame. $\boldsymbol{t}_{IC}$ is the translation vector points from the IMU to the camera, expressed in the IMU frame. $\boldsymbol{\theta}(\cdot)$ converts a quaternion to an axis-angle expression. Elements of $\boldsymbol{\sigma_t^2}$ and $\boldsymbol{\sigma_\theta^2}$ are used as the diagonal elements of the measurement noise matrix. After the measurement update, the *a posterior* relative pose estimation ${}^{I_t}\hat{\boldsymbol{p}}_{R_k}$ and ${}^{I_t}_{R_k}\hat{\boldsymbol{q}}$ is composed to the global pose, as shown in Eq. (5.9).

$$ {}^{R_{k+1}}_{G}\boldsymbol{q} = {}^{I_t}_{R_k}\hat{\boldsymbol{q}} \otimes {}^{R_k}_{G}\boldsymbol{q}, \quad {}^{R_{k+1}}\boldsymbol{p}_G = \boldsymbol{R}({}^{I_t}_{R_k}\hat{\boldsymbol{q}})^T \cdot {}^{R_k}\boldsymbol{p}_G + {}^{I_t}\hat{\boldsymbol{p}}_{R_k} \tag{5.9}$$

The current IMU frame I_t becomes the new reference frame R_{k+1}. The expression of the gravity vector $\boldsymbol{g}_{R_{k+1}}$ in R_{k+1} is calculated as $\boldsymbol{g}_{R_{k+1}} = \boldsymbol{R}({}^{I_t}_{R_k}\hat{\boldsymbol{q}})^T \cdot \hat{\boldsymbol{g}}_{R_k}$. ${}^{I_t}\boldsymbol{p}_{R_{k+1}}$ and ${}^{I_t}_{R_{k+1}}\boldsymbol{q}$ are set to a zero vector and a unit quaternion whose vector part is a zero vector, respectively. Their corresponding elements in the covariance matrix are set to zeros too. Readers who are interested in the filter design can refer to [40] for more details.

For the EKF, when $\boldsymbol{\mu_t}$ is the normalized translation, the propagated camera translation is normalized accordingly. The measurement equation of translational motion is then

$$\boldsymbol{z_{\mu_t}} = R_I^C \cdot \frac{\boldsymbol{t}_{IC} + {}^{I_t}\boldsymbol{p}_{R_k} - \boldsymbol{R}({}^{I_t}_{R_k}\boldsymbol{q})^T \cdot \boldsymbol{t}_{IC}}{\| \boldsymbol{t}_{IC} + {}^{I_t}\boldsymbol{p}_{R_k} - \boldsymbol{R}({}^{I_t}_{R_k}\boldsymbol{q})^T \cdot \boldsymbol{t}_{IC} \|} + \boldsymbol{w_t} \tag{5.10}$$

5.3.3. EVALUATION

In the previous part of this section, we have successively introduced the vision front-end which is the uncertainty-aware pose network and the EKF-based robocentric back-end. The VIO system consisting of them is referred to as PoseNet-VIO in this chapter. In this subsection, PoseNet-VIO is evaluated on the EuRoC MAV dataset [11]. The PoseNet-VIO variants in Table 5.2 have the same parameters in the EKF-based back-end. Their vision front-ends, the pose networks, are different. They are trained by the NLL loss (Eq. (5.3)) with different learning targets. The learning targets of the pose networks of ① and ② are the pose predictions of the self-supervised teacher networks. The remaining four networks learn from the ground truth. The translation prediction of ① to ④ in Table 5.2 are normalized to a unit vector that indicates the direction of translational motion. In contrast, ⑤ and ⑥ predict translational motion with the metric scale that is learned from the ground-truth labels of the relative poses. Since we use monocular videos to train the

Table 5.2: Accuracy of the proposed method (top group, rows 1 to 6) compared with the baseline method (bottom group, rows 7 and 8). The name of the EuRoC dataset flight sequences (*Vicon Room 101 to 103, 201 to 203*, and *Machine Hall 01 to 05*) are abbreviated to fit the page width. "V" stands for *Vicon Room*, and "MH" stands for *Machine Hall*. Data below the sequence names shows the root-mean-square errors (RMSE) of the absolute translation errors (ATEs) of the estimated trajectories. **Bold** represents the overall best and underline represents the best of the proposed method.

ID	FT[1]	GT[2]	MS[3]	V11	V12	V13	V21	V22	V23	MH1	MH2	MH3	MH4	MH5
①				4.24	2.34	3.04	4.05	4.97	5.88	6.42	3.92	4.79	14.1	4.04
②	✓			-	-	2.23	-	-	4.33	-	-	-	-	4.25
③		✓		3.48	2.51	2.74	4.07	5.49	7.73	5.60	3.93	6.18	17.7	19.8
④	✓	✓		-	-	2.42	-	-	7.99	-	-	-	-	8.16
⑤		✓	✓	1.23	2.86	3.74	1.86	4.37	4.27	7.24	5.06	**4.26**	3.24	3.68
⑥	✓	✓	✓	-	-	1.55	-	-	4.36	-	-	-	-	3.04
⑦	MSCKF (51 pts)			0.16	0.13	0.12	245	0.13	**0.16**	38.0	5.20	130	2.35	**1.00**
⑧	MSCKF (199 pts)			**0.09**	**0.09**	**0.11**	**0.12**	**0.10**	0.20	**0.34**	**0.24**	58.5	**0.65**	1.54

[1] Fine-tuning (FT) the pose network on the training sequences of the EuRoC dataset.
[2] Using ground-truth (GT) labels as the learning targets in the NLL loss. If not using GT, the learning targets are the predictions of the self-supervised teacher networks.
[3] The pose network learns to predict translational motion with the metric scale (MS). Otherwise, the prediction is the normalized translation, a unit vector.

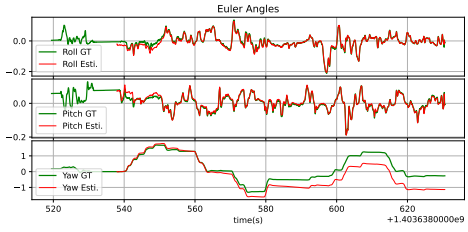
teacher networks, their prediction of translational motion has an ambiguous scale. Therefore, the pose networks trained by the teacher networks can only have normalized translational motion predictions.

The root-mean-square error (RMSE) of absolute translation errors (ATE) is the metric of VIO accuracy. The calculation is conducted by [37]. The alignment of the estimated trajectory and the ground-truth trajectory has 4-DoF (yaw and 3-d translation). PoseNet-VIO is compared with a state-of-the-art (SOTA) VIO MSCKF [1] implemented by [42]. The C++ code of PoseNet-VIO is implemented based on the open-sourced code of [42]. Here we only compare with one SOTA VIO since the gap in accuracy is big, as shown in Table 5.2. The comparison of MSCKF with other VIO solutions can be found in [43].

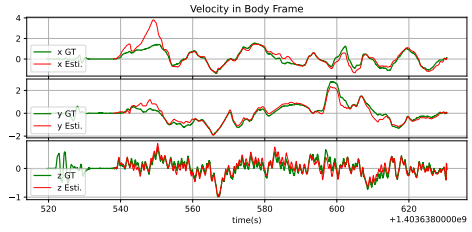
At the beginning of the five EuRoC sequences collected in the *Machine Hall*, the MAV was moved by a human operator and then stayed stationary for a while before taking off. For a PoseNet-VIO whose front-end pose network has normalized translation predictions, the lack of translational motion leads to drift. So, when testing PoseNet-VIO variants ① to ④, the starting time points of the *Machine Hall* sequences are about one second before the takeoff. For MSCKF, if there was a significant drift in the beginning, we did the same.

A design target of PoseNet-VIO is high efficiency. Thus it has non-iterative network inference and a basic EKF. The time consumption is constant. But since MSCKF uses feature points, the number of points affects the time consumption. The time consumptions of PoseNet-VIO and MSCKF are measured during the tests on *Machine Hall 05* sequence of EuRoC. For PoseNet-VIO, the average total time cost is 7.811 ms, of which 6.036 ms is for network inference. For the default setting of MSCKF (⑧ in Table 5.2), the average total time cost is 17.415 ms, with 7.583 ms for processing feature points. The average number

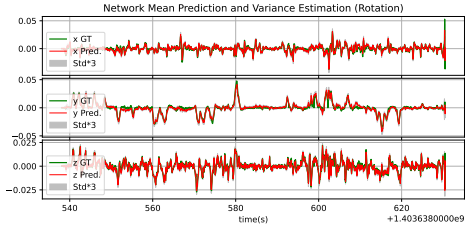
of points is around 199. We modified the number of feature points to track to a smaller number than the default to make the average of the total time consumption of processing a frame close to the time consumption of PoseNet-VIO. When the average number of maintained points in an image frame is around 51 (⑦ in Table 5.2), the average total time cost is 7.933 ms, with 3.821 ms for feature point tracking and detection.



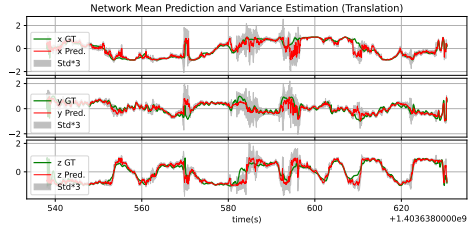
(a) VIO Euler Angles Estimation (rad).



(b) VIO Velocity Estimation Expressed in MAV Body Frame (m/s).



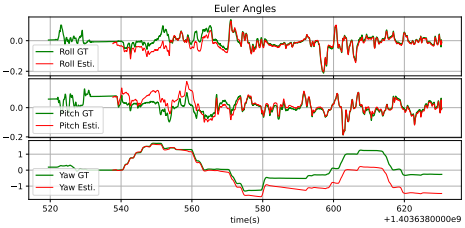
(c) Pose Network Rotation (Axis–Angle) Predictions (rad).



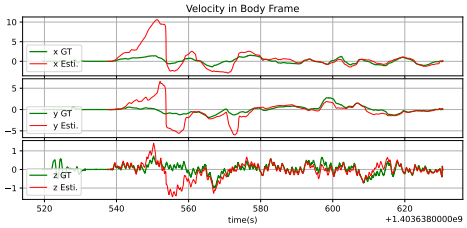
(d) Pose Network Normalized Translation Predictions.

Figure 5.3: Results of PoseNet-VIO variant ①, tested on *Machine Hall 05*. The top two subplots show the EKF *a posterior* estimations (VIO outputs) of the MAV attitude and translational velocity expressed in the MAV body frame. The attitude is relative to the global frame whose z-axis is opposite to the gravity direction. The bottom two subplots show the outputs (mean prediction and uncertainty estimation of the relative pose) of the front-end pose network.

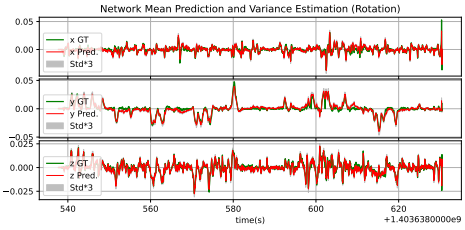
In the sequences *Vicon Room 2 01*, *Machine Hall 01*, and *Machine Hall 03*, MSCKF drifts after very slow motion, which leads to big trajectory errors. But in general, as shown in Table 5.2, PoseNet-VIO is far less accurate than MSCKF. Comparing the variants of PoseNet-VIO, a noticeable phenomenon is that networks predicting translational motion with metric scale lead to better VIO accuracy than networks predicting normalized translation. Comparing the ATEs of ③ and ⑤, we can see that ⑤ has advantages in seven out of the eleven sequences. This is further evidenced by Fig. 5.4 (b) and Fig. 5.5 (b), where the estimated velocity of ⑤ is more accurate. Without utilizing the knowledge of objects' sizes in the scene, the metric scale is unobservable for a monocular video. The metric-scale translation prediction is based on the knowledge of the training set. It can be problematic when generalizing to a different dataset. But Fig. 5.5 (d) shows that the errors of translational motion predictions are acceptable. The generalization is fine. We attribute it to the large number and wide distribution of the training samples. Subplot (b) for VIO velocity estimations of Fig. 5.4 shows that using normalized translation (③) leads to big



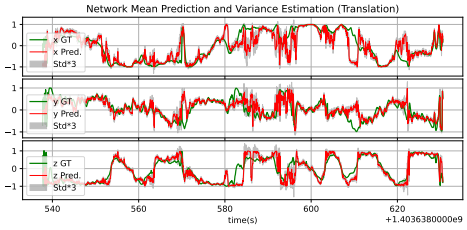
(a) VIO Euler Angles Estimation (rad).



(b) VIO Velocity Estimation Expressed in MAV Body Frame (m/s).

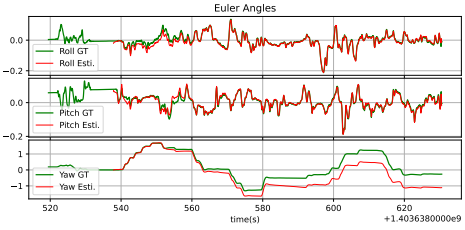


(c) Pose Network Rotation (Axis–Angle) Predictions (rad).

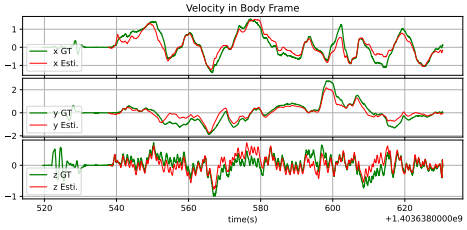


(d) Pose Network Normalized Translation Predictions.

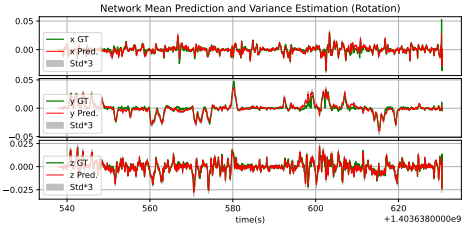
Figure 5.4: Results of PoseNet-VIO variant ③, tested on *Machine Hall 05*. See the caption of Fig. 5.3 for more information on the states shown in the subplots.



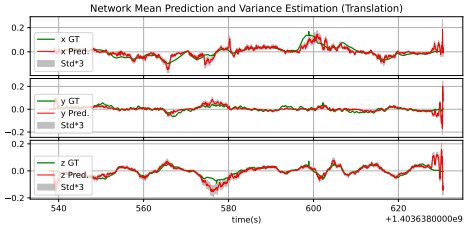
(a) VIO Euler Angles Estimation (rad).



(b) VIO Velocity Estimation Expressed in MAV Body Frame (m/s)..

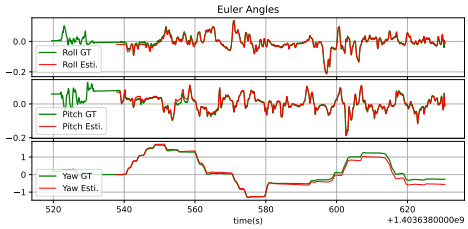


(c) Pose Network Rotation (Axis–Angle) Predictions (rad).

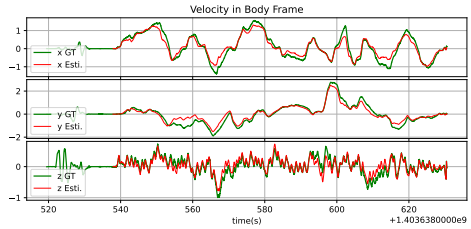


(d) Pose Network Translation Predictions (m).

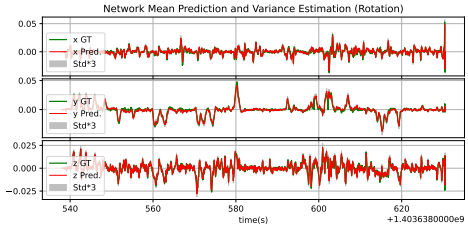
Figure 5.5: Results of PoseNet-VIO variant ⑤, tested on *Machine Hall 05*. See the caption of Fig. 5.3 for more information on the states shown in the subplots.



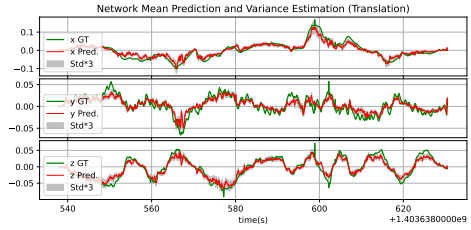
(a) VIO Euler Angles Estimation (rad).



(b) VIO Velocity Estimation Expressed in MAV Body Frame (m/s).



(c) Pose Network Rotation (Axis–Angle) Predictions (rad).



(d) Pose Network Translation Predictions (m).

Figure 5.6: Results of PoseNet-VIO variant ⑥, tested on *Machine Hall 05*. See the caption of Fig. 5.3 for more information on the states shown in the subplots.

drifts in the beginning before the estimations converge after around 35 seconds. For ①, it takes around 10 seconds for the velocity estimation to converge, as shown in subplot (b) of Fig. 5.3. While using metric-scale translation does not have this problem, as shown in the two subplots (b) of Fig. 5.5 and Fig. 5.6. In summary, given the current design, using normalized translation leads to worse VIO accuracy in cross-dataset evaluation.

Another phenomenon worth noticing is the effect of fine-tuning. Fine-tuning brings better accuracy for all the variants, indicating that although we have a big-scale and widely distributed training set, the generalization capacity can still be improved. When the translation predictions have the metric scale, we can see better accuracy in all subplots of Fig. 5.6 than Fig. 5.5. Especially for the pose network's predictions of translational motion shown in subplot (d), the predictions are more accurate after fine-tuning. The possible reason is that the network further learns the metric scale of the objects captured in the EuRoC training set.

The PoseNet-VIO variant ① interests us most because the training does not use any ground-truth labels or in-domain data. Given the not-excellent but acceptable accuracy in Euler angle estimation and velocity estimation after the convergence, as shown in Fig. 5.3, we think this PoseNet-VIO variant can act as an attitude and velocity estimator and be used for short-term navigation. From Table 5.2, Fig. 5.3, and Fig. 5.4, we notice that using the ground-truth poses in the training of the front-end pose network (③) does not improve the VIO accuracy over using the predictions of the self-supervised teacher networks (①). In eight out of eleven sequences, PoseNet-VIO variant ① has better accuracy. The accuracy of network predictions of rotation has no clear difference, as shown in the two

subplots (c) of Fig. 5.3 and Fig. 5.4. But for the translation prediction shown in the two subplots (d), ① performs better than ③, especially in terms of uncertainty estimation. The three-time standard deviations of ① better reflect the errors of the mean predictions. This phenomenon indicates that self-supervised teacher networks are powerful replacements for ground-truth labels.

As shown in the subplot (c) of Fig. 5.3, the network predictions of relative rotation are aligned with the ground truth, as also shown in Fig. 5.4, Fig. 5.5, and Fig. 5.6. The translation predictions are relatively worse and noisier. Around 585, 595, and 610 seconds, translation predictions are obviously inaccurate, as shown in subplot (d) of Fig. 5.3. From subplot (b), we can see that the velocity is slow during those time slots. Slow velocity is a possible trigger of inaccurate translation prediction. More research is required to enable the network that predicts translation direction to better handle slow motion. We leave it to potential future work. About potential ways of improving the PoseNet-VIO, one way is a VIO back-end that utilizes network predictions of multiple time steps instead of only the newest network prediction (current design). Another way is to learn the metric scale from the self-supervised teacher networks. It requires the teacher networks to be trained on stereo videos.

5.4. LIGHTWEIGHT MONOCULAR DEPTH NETWORK

After studying ego-motion estimation, in the rest of this chapter, we explore approaches to train a lightweight monocular depth network and improve its prediction accuracy using the self-supervised teacher networks. The architecture of the lightweight depth network (LDN) is shown in Fig. 5.7. It is a simplified variant of the depth network used in [14], which is based on ResNet-18. Most depth networks in the literature have the downsampling encoder and upsampling decoder architecture, predicting a depth map with the same resolution as the input image. On the contrary, our network performs three times downsampling to reach 1/8th resolution without any upsampling layer. It has 391,793 trainable parameters in total. According to our knowledge of the literature, only the MiniNet [26] has fewer parameters than the proposed network. MiniNet has an iterative recurrent module, while our network performs a one-time forward pass. Note that we do not compare the proposed depth network with other lightweight depth networks in the rest of this chapter because our focus is on ground-truth-free training schemes instead of network architecture design. The training schemes studied can be applied to a depth network with any architecture.

The main reason for such a network architecture is that the network is developed to run onboard a nano quadcopter. The network architecture has to be hardware-compatible with the Crazyflie nano quadcopter. It is equipped with a Bitcraze AI-deck [44] that carries a tiny camera and a GAP8 processor where the depth network can be deployed. A PyTorch-trained network model is required to be converted by the software [45, 46] to be flashed into the onboard processor. The proposed network architecture is empirically optimized under the given hardware/software constraints. The resolution of the images captured by the onboard camera is 324×244. They are cropped to 320×224 before input to the depth network. The 1/8th resolution depth map has 40×28 pixels. The resolution may seem low, but it provides adequate information to detect nearby obstacles. For training, the low resolution brings no significant problems to either self-supervised learning or knowledge

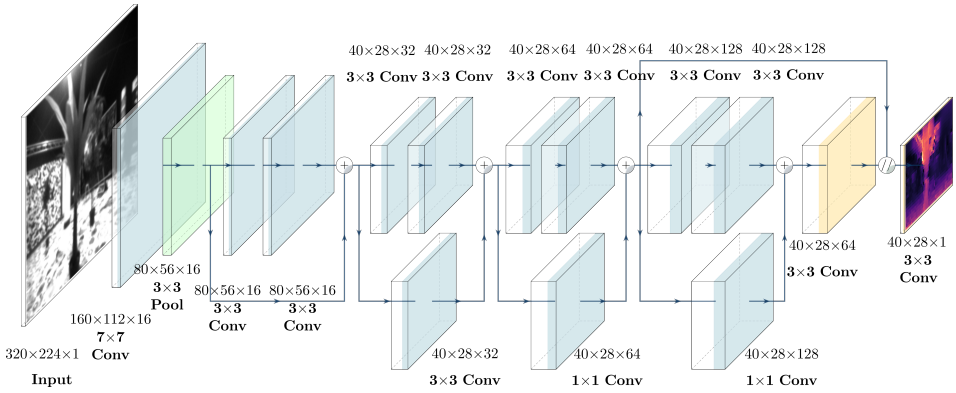


Figure 5.7: Architecture of the lightweight CNN for monocular depth prediction. "+" denotes element-wise summation and "//" stands for tensor concatenation along the channel dimension. Blue color denotes that the convolutional layer is followed by a batch normalization layer and a ReLU activation function. Color yellow denotes that the convolutional layer is followed by a ReLU6 activation function. Color green denotes a max pooling layer with stride=2.

distillation.

5.4.1. TRAINING SCHEMES

To explore the best way to train LDN, we tried different loss functions, hyperparameters, and teacher networks, as shown in Table 5.3. We compare five training schemes by training on three datasets, corresponding to the three groups divided by horizontal lines in Table 5.3. The Teacher networks shown in the first row of each group are trained from scratch on the individual dataset using the hyperparameters introduced in Section 5.2, except that the batch size is 8. They are the pre-trained teacher (PTT) networks in the training of LDN-2, LDN-3, and LDN-4. The teacher networks perform only inference in the training of an LDN with their parameters fixed. LDN-5 also learns from its teacher networks. But the teacher networks of LDN-5 are not pre-trained. They are trained from scratch simultaneously with LDN-5. The training setup of LDN-5's teacher networks is the same as the Teacher networks in the first row of Table 5.3. During the training, LDN-5 and its teacher networks have the same inputs at each step. The predictions of the teacher networks are used to train the LDN-5, which means, at the beginning of training, LDN-5 learns from the random predictions of the randomly initialized teacher networks. As the training goes on, the teacher networks' predictions have higher and higher accuracy and LDN-5 utilizes these better predictions in its training.

There are two supervision signals for training an LDN. The first is self-supervised learning using the enhanced reprojection-based loss (Eq. (5.2)). The photometric matching difference between the synthesized image by image warping and the actually captured image is the main supervision signal. Image warping requires both depth map prediction and relative pose prediction. In the training of an LDN, the depth map is predicted by the LDN. For the pose prediction, we implement and compare two sources. The first one is

to train a pose network from scratch simultaneously with the LDN and use the prediction of the pose network to construct the loss. The training of LDN-1 uses this scheme. In Section 5.2, the teacher networks use the reprojection-based loss (Eq. (5.2)) with a weight $\lambda_{\text{geo}} = 0.2$ for the geometry consistency loss and three iterations of pose network inference ($N = 3$). For training an LDN-1, we found that a smaller weight for geometry consistency loss $\lambda_{\text{geo}} = 0.02$ leads to better accuracy. As for the number of pose predictions, the pose network infers only once in each training step of LDN-1. The reason follows. Both the depth network and the pose network take full-resolution grey-scale images as inputs. In constructing the photometric matching loss, the grey-scale images are downsampled to 1/8th resolution to have the same resolution as the predicted depth maps D_{LDN}. The resolution of the synthesized images is also 1/8th of the original images. But the pose network infers from the images with full resolution. The low-resolution synthesized images cannot act as the pose network input, and thus the iterative pose prediction cannot be used in the training of LDN-1, which means $N = 1$ in Eq. (5.2). The second option for getting the relative pose is using the pose predictions of the teacher networks. LDN-3, LDN-4, and LDN-5 use this scheme. For LDN-3 and LDN-4, the teacher networks are pre-trained, so the pose prediction accuracy is high from the beginning. For LDN-5, the teacher pose prediction accuracy keeps increasing while training.

The second supervision signal is the knowledge distillation (KD) loss from the deeper teacher depth network. The teacher depth network predicts depth maps D_T with the same resolution as the input image. A low-resolution depth map D_{LDN} predicted by an LDN is upsampled via bilinear interpolation to have the same resolution as D_T that serves as the proxy label. We use the $L1$ loss. The weight of KD loss is a constant value of 1.0. As shown in Table 5.3, LDN-2, LDN-3, LDN-4, and LDN-5 use this KD loss. For LDN-2, the KD loss is the only supervision signal. For LDN-3, LDN-4, and LDN-5, both the KD loss and the reprojection-based self-supervised loss are used in combination, as shown in Eq. (5.11).

$$L_{\text{LDN,comb.}} = \lambda_{\text{geo}} \cdot L_{\text{geo}} + L_{\text{photo.}} + L_{\text{KD}}, \quad L_{\text{KD}} = |D_T - up(D_{\text{LDN}})| \tag{5.11}$$

We take three subsets from the TartanAir [35] dataset to evaluate the training schemes. Raw images and ground-truth depth maps are resized to the resolution of 320×224. The subset of the top group is made of *office* and *office2* environments. It has 18,505 and 3,135 images for training and testing, respectively. The networks are trained for 120 epochs. The second subset is collected in the *neighborhood* environment. There are 27,466 training items and 4,647 validation items. The number of training epochs is 80. The subset of the group at the bottom of the table contains the *seasonsforest* and *seasonsforest winter* environments. We randomly split the images into 14,251 training samples and 2,408 testing samples. The trainings last for 155 epochs. The batch size is 8 for all three datasets.

Ground-truth depth maps are downsampled to the same resolution as LDN predictions for accuracy evaluation. Many images have pixels filming the sky, which have infinite depths. Because the sky is mostly texture-less, the self-supervised learning scheme cannot learn the correct depth. Therefore we exclude from the evaluation all pixels whose ground-truth depth is more than 100 meters. We follow the same evaluation procedure as [14]. Predicted depth maps have ambiguous scales. A scalar factor is calculated by dividing the median of the depths in the ground-truth map by the median of the depths in the predicted map. Then the depths in the predicted map are linearly scaled by this

factor to align with the scale of the ground truth. The seven metrics of depth prediction accuracy are shown in the first row of Table 5.3. The first four count the error of all pixels. $\delta < 1.25^n$ indicate the ratio of the predictions whose differences to the ground-truth values lay within the thresholds, *i.e.*, close enough to the ground truth. A down arrow indicates that a lower value means higher accuracy, and an up arrow denotes that a higher value means higher accuracy.

From the data in Table 5.3, we notice that using KD as the only supervision signal (LDN-2) may lead to outlier depth predictions that are very inaccurate, as indicated by the big *Sq Rel* values in the first and the third groups. Using both KD and SSL but without the geometry consistency loss (LDN-3) leads to outliers as well. Using SSL loss alone to train the LDN and a pose network from scratch (LDN-1) produces few apparent outliers. Comparing LDN-1's accuracy with LDN-4, the gap is evident, especially in the metrics $\delta < 1.25^n$. Combining all loss terms, LDN-4 has the overall best performance. In all metrics, the accuracy of LDN-5 is only slightly worse than LDN-4. An advantage of LDN-5 over LDN-4 is that LDN-5 requires only one training process. There is no need to wait for the training of the teacher networks to be finished and then train the LDN. It shortens the time consumption for deploying in a new target environment. Another phenomenon we observed from Table 5.3 is that for all the metrics except for $\delta < 1.25$, LDN-4 is not much worse than the Teacher networks. It shows that under the experimental conditions, the performance of an LDN with only 391,793 parameters does not fall much behind the much bigger teacher depth network with ~37 times more parameters (14,833,945). This increases the confidence in LDN and nano quadcopters for obstacle detection in a known environment. Examples of the predictions of LDN-4 on the testing datasets are shown in Fig. 5.8.

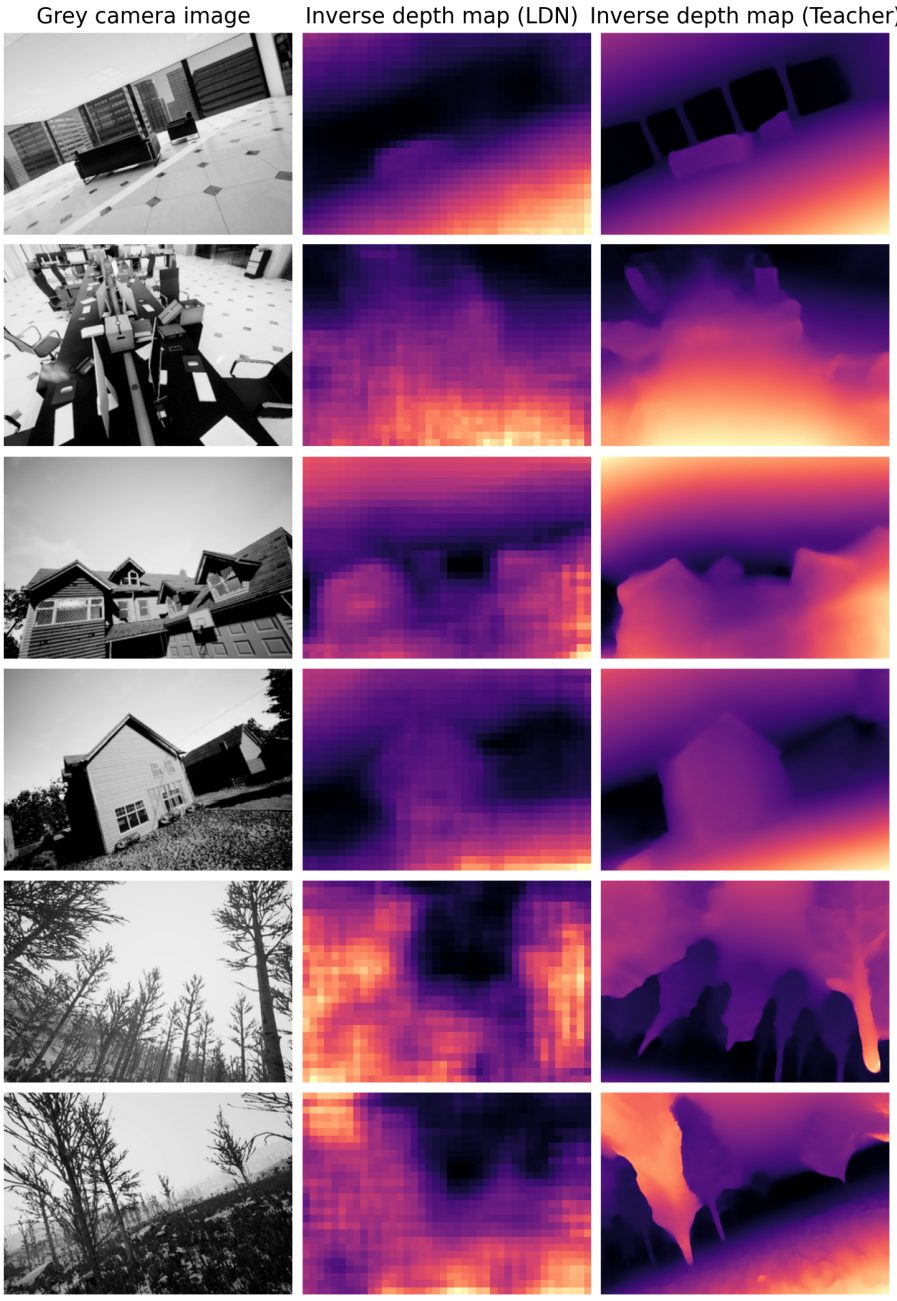


Figure 5.8: The depth map predictions of LDN-4 (middle column) and Teacher depth network (right column). The first two rows are examples of the *office* environment. The third and fourth rows show the *neighborhood* environment. The two rows at the bottom correspond to the *seasonsforest* environment.

Table 5.3: Ablation study on quantitative depth prediction accuracy. Three groups of comparative experiments are run on three different datasets. A **bold number** indicates the best accuracy metric achieved by an LDN.

Model	SSL[1]	λ_{geo}	KD[2]	PTT[3]	Abs Rel↓	Sq Rel↓	RMSE↓	RMSE log↓	$\delta < 1.25$↑	$\delta < 1.25^2$↑	$\delta < 1.25^3$↑
Teacher	✓	0.2			0.0925	0.8052	4.7785	0.2378	91.30 %	95.75 %	97.20 %
LDN-1	✓	0.02			0.2735	1.3616	5.9309	0.4399	57.38 %	82.78 %	91.74 %
LDN-2			✓	✓	0.2542	260.60	22.072	0.3115	**83.76** %	**93.67** %	**96.26** %
LDN-3	✓	0.0	✓	✓	0.2537	397.25	21.415	0.3124	83.30 %	93.64 %	96.24 %
LDN-4	✓	0.2	✓	✓	**0.1574**	**1.0726**	**5.4606**	**0.3104**	82.80 %	93.30 %	96.10 %
LDN-5	✓	0.2	✓		0.1617	1.1024	5.4925	0.3165	81.77 %	92.99 %	95.96 %
Teacher	✓	0.2			0.2373	2.1778	7.1647	0.3184	71.15 %	86.23 %	92.57 %
LDN-1	✓	0.02			0.3888	4.2736	10.552	0.5167	45.72 %	70.60 %	82.73 %
LDN-2			✓	✓	0.3202	3.7768	**8.2885**	0.3936	57.87 %	80.12 %	89.70 %
LDN-3	✓	0.0	✓	✓	**0.3186**	7.7472	8.4581	**0.3925**	**57.91** %	**80.20** %	**89.80** %
LDN-4	✓	0.2	✓	✓	0.3204	**2.8655**	8.5541	0.4044	56.88 %	79.00 %	88.87 %
LDN-5	✓	0.2	✓		0.3232	2.9012	8.5883	0.4064	56.58 %	78.76 %	88.76 %
Teacher	✓	0.2			0.4596	5.1482	10.596	0.5282	58.06 %	76.01 %	84.18 %
LDN-1	✓	0.02			**0.5374**	7.7569	14.000	0.6515	47.32 %	65.97 %	76.04 %
LDN-2			✓	✓	0.5945	48.237	13.878	0.5753	52.91 %	72.29 %	81.61 %
LDN-3	✓	0.0	✓	✓	0.5659	33.697	16.351	**0.5683**	**53.54** %	**72.65** %	**81.89** %
LDN-4	✓	0.2	✓	✓	0.5804	**6.9544**	**12.673**	0.6003	50.84 %	69.80 %	79.28 %
LDN-5	✓	0.2	✓		0.5865	7.0496	12.700	0.6026	50.71 %	69.64 %	79.09 %

[1] This column indicates whether the network training uses self-supervised learning (SSL). When the teacher networks are available, the relative pose prediction that constructs the reprojection-based loss for the LDN is the prediction of the teacher networks.

[2] Model Distillation (KD). The LDN acts as a student network that imitates the outputs of the teacher depth network using $L1$ loss. This column also indicates whether there is a teacher network involved in training.

[3] Pre-trained Teacher (PTT). It is the Teacher model in the first row of each group. The parameters of PTT stay fixed during the training of LDN. When the teacher model is not pre-trained, as in the case of LDN-5, it is trained by SSL from scratch along with the LDN.

5.4.2. REAL-WORLD TESTING

(a) *CyberZoo* environment. The obstacles include screens, chairs, artificial plants, pillars, humans, *etc.*

(b) A typical *Corridor* in an office building where there are trashcans and chairs.

Figure 5.9: Real-world testing environments.

We collect two datasets in two real-world environments. The grey-scale images captured by the tiny camera onboard the quadcopter were transmitted via wireless communication to a laptop computer. During data collection, we held the quadcopter in hand and performed random motion. One environment is a controllable experimental environment *CyberZoo* (Fig. 5.9 (a)). In this environment, the layout of objects is easy to change in order to test with different conditions, *e.g.*, density and type of obstacles. 9,151 images were captured in total. Another environment is a typical *corridor* in an office building (Fig. 5.9 (b)). Here we captured 2,717 images. There is no ground-truth label available and all the images are used for training. Two LDNs are trained from scratch for 480 epochs on these two datasets, respectively, by the training scheme of LDN-5 in Table 5.3. These two trained LDNs are for flight experiments, aiming to detect the obstacles in the environments where the training datasets are collected. Two testing examples of the LDN trained on the *corridor* dataset are shown in the top two rows of Fig. 5.10. Example predictions of the LDN

trained on the *CyberZoo* dataset are shown in the bottom two rows of Fig. 5.10. The camera distortion is ignored in both training and testing to lower the computation for image preprocessing. During flight experiments, the LDN directly infers from the cropped raw images.

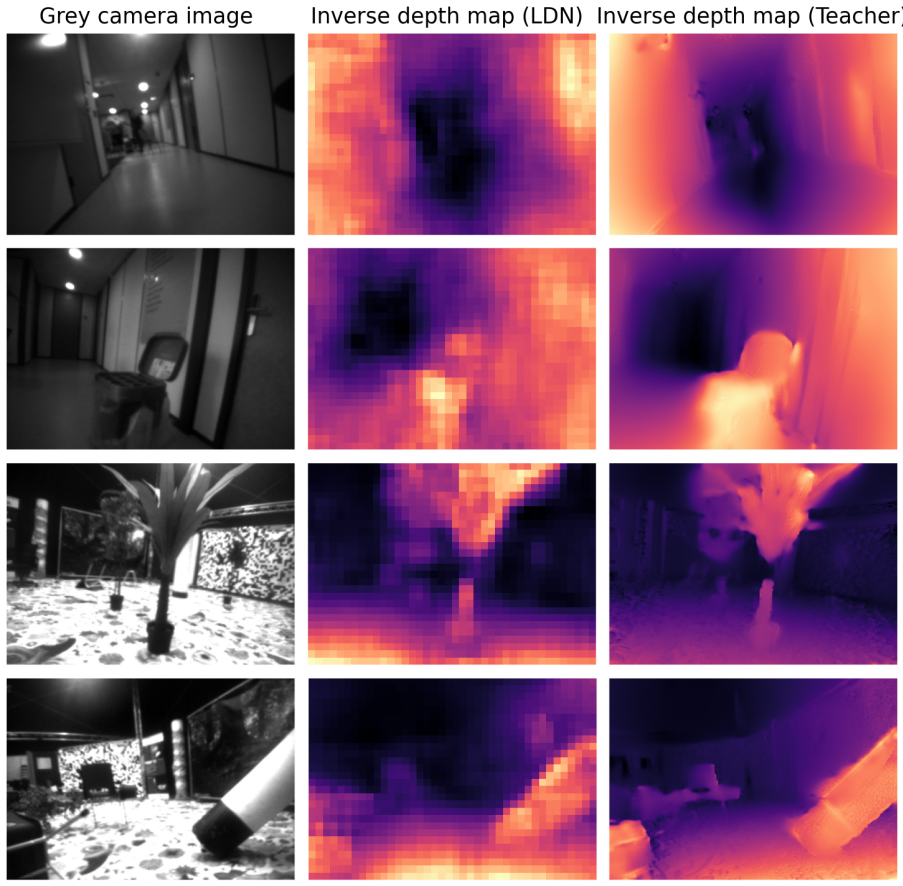


Figure 5.10: The depth map predictions of LDN-5 (middle column) and Teacher depth network (right column). The top two rows show example images of the *corridor* environment. The bottom two rows correspond to the *cyberzoo* environment.

The trained LDNs are then used for obstacle detection in flight experiments of a nano quadcopter. The captured grey-scale images by the onboard camera were transposed via wireless communication to a laptop computer, where the network inferred from the images and a flight controller generated high-level control commands according to the depth map predictions. In this chapter, the depth network runs offboard. Onboard deployment is left for potential future work. The network inference frequency is around 7 Hz, mainly restricted by the time consumption of image transfer. Example images and

predicted depth maps logged during the flight experiments are shown in Fig. 5.10.

During a flight experiment in the *CyberZoo*, we placed objects that were not captured by the training dataset in the flight arena. The quadcopter avoided all of them successfully in multiple runs. In Fig. 5.11, we show examples of detecting unknown obstacles. In the first row, a running human with motion blur was successfully detected, although humans stood still in the training dataset. The second row shows an example of a partially detected unseen obstacle. We speculate that the network's generalization to the unknown objects can be attributed to the familiar background in the environment. To further explore the generalization capacity, we use an LDN trained on the *CyberZoo* dataset to navigate the quadcopter through the *Corridor* environment. As shown in the third row in Fig. 5.11, the network detects a part of the wall. A possible reason is that the wall and floor structures also appear in the *CyberZoo* environment. The quadcopter flew safely through the corridor until it crashed on an undetected trashcan, as shown in the fourth row of Fig. 5.11.

5.5. CONCLUSIONS

For ego-motion estimation, we proposed the PoseNet-VIO based on an uncertainty-aware pose network and an EKF. The accuracy of PoseNet-VIO is worse than mainstream VIO solutions, but it may act as an efficient attitude and velocity estimator for short-term navigation. Based on the cross-dataset evaluation and the comparison between different types of supervision, we have the following findings. a) The simulation-to-reality generalization capacity of the pose network is generally satisfactory, also for the metric-scale translation prediction. b) Training an uncertainty-aware pose network using the predictions of self-supervised teacher networks leads to rivaling VIO accuracy to using ground-truth pose. c) Metric-scale translational motion predictions produce better VIO accuracy than normalized translation predictions. These findings can be valuable to future research on learning-based visual ego-motion estimation.

For depth prediction, we verify that the combination of knowledge distillation and reprojection-based self-supervised learning leads to the decent accuracy of a lightweight monocular depth network. We focus on the case that the network is trained on a dataset collected in a known target environment. Experiments show that the proposed network has some generalization capability. As for works [25, 29] that pursue outstanding generalization by exploiting huge training datasets, our network is not comparable. When the flight environment is unknown and network generalization has high priority, the workflow proposed in [29] can be followed to train our depth network. The workflow is to generate proxy labels for images in the target environment by a trained big-size depth network. The proxy labels are expected to be accurate enough because the big-size depth network is trained on a large number of images of a big distribution, and thus it is generalization-capable. An LDN can be trained using the proxy labels.

REFERENCES

[1] M. Li and A. I. Mourikis, *High-precision, consistent ekf-based visual-inertial odometry,* The International Journal of Robotics Research **32**, 690 (2013).

[2] T. Qin, P. Li, and S. Shen, *Vins-mono: A robust and versatile monocular visual-inertial*

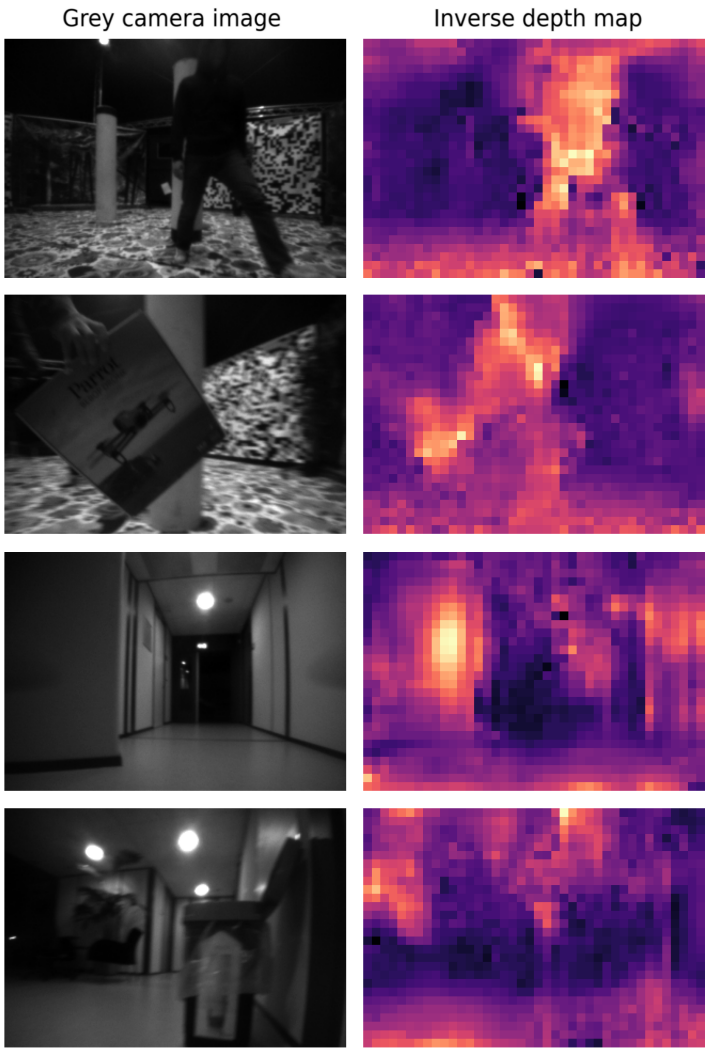


Figure 5.11: From top to bottom: dynamic obstacle (running human), unknown obstacle (carton), unknown environment (corridor) and a failure case of detecting an unknown obstacle (trashcan).

state estimator, IEEE Transactions on Robotics **34**, 1004 (2018).

[3] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, *Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam,* IEEE Transactions on Robotics **37**, 1874 (2021).

[4] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, *Unsupervised learning of depth and ego-motion from video,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017) pp. 1851–1858.

[5] S. Wang, R. Clark, H. Wen, and N. Trigoni, *End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks,* The International Journal of Robotics Research **37**, 513 (2018).

[6] B. Wagstaff, E. Wise, and J. Kelly, *A self-supervised, differentiable kalman filter for uncertainty-aware visual-inertial odometry,* arXiv preprint arXiv:2203.07207 (2022).

[7] Z. Teed and J. Deng, *Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras,* Advances in neural information processing systems **34**, 16558 (2021).

[8] W. Wang, Y. Hu, and S. Scherer, *Tartanvo: A generalizable learning-based vo,* in *Conference on Robot Learning* (PMLR, 2021) pp. 1761–1772.

[9] Bitcraze, *Motion capture positioning,* `https://www.bitcraze.io/documentation/system/positioning/mocap-positioning/` (2023), accessed on 3rd April 2023.

[10] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers, *D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020) pp. 1281–1292.

[11] 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 **35**, 1157 (2016).

[12] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, *Orb-slam: a versatile and accurate monocular slam system,* IEEE transactions on robotics **31**, 1147 (2015).

[13] 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**, 249 (2016).

[14] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, *Digging into self-supervised monocular depth estimation,* in *Proceedings of the IEEE/CVF international conference on computer vision* (2019) pp. 3828–3838.

[15] J. Bian, Z. Li, N. Wang, H. Zhan, C. Shen, M.-M. Cheng, and I. Reid, *Unsupervised scale-consistent depth and ego-motion learning from monocular video,* Advances in neural information processing systems **32** (2019).

[16] B. Wagstaff, V. Peretroukhin, and J. Kelly, *On the coupling of depth and egomotion networks for self-supervised structure from motion,* IEEE Robotics and Automation Letters **7**, 6766 (2022).

[17] Y. Almalioglu, M. Turan, M. R. U. Saputra, P. P. de Gusmão, A. Markham, and N. Trigoni, *Selfvio: Self-supervised deep monocular visual–inertial odometry and depth estimation,* Neural Networks **150**, 119 (2022).

[18] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, *Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem,* in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31 (2017).

[19] G. Huang, *Visual-inertial navigation: A concise review,* in *2019 international conference on robotics and automation (ICRA)* (IEEE, 2019) pp. 9572–9582.

[20] R. Mur-Artal and J. D. Tardós, *Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,* IEEE transactions on robotics **33**, 1255 (2017).

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

[22] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen, *Autonomous aerial navigation using monocular visual-inertial fusion,* Journal of Field Robotics **35**, 23 (2018).

[23] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, *Dronet: Learning to fly by driving,* IEEE Robotics and Automation Letters **3**, 1088 (2018).

[24] R. J. Bouwmeester, F. Paredes-Vallés, and G. C. de Croon, *Nanoflownet: Real-time dense optical flow on a nano quadcopter,* arXiv preprint arXiv:2209.06918 (2022).

[25] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, *Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer,* IEEE transactions on pattern analysis and machine intelligence (2020).

[26] J. Liu, Q. Li, R. Cao, W. Tang, and G. Qiu, *Mininet: An extremely lightweight convolutional neural network for real-time unsupervised monocular depth estimation,* ISPRS Journal of Photogrammetry and Remote Sensing **166**, 255 (2020).

[27] M. Xiong, Z. Zhang, T. Zhang, and H. Xiong, *Ld-net: A lightweight network for real-time self-supervised monocular depth estimation,* IEEE Signal Processing Letters **29**, 882 (2022).

[28] M. Poggi, F. Tosi, F. Aleotti, and S. Mattoccia, *Real-time self-supervised monocular depth estimation without gpu,* IEEE Transactions on Intelligent Transportation Systems (2022).

[29] F. Aleotti, G. Zaccaroni, L. Bartolomei, M. Poggi, F. Tosi, and S. Mattoccia, *Real-time single image depth perception in the wild with handheld devices,* Sensors **21**, 15 (2020).

[30] J. Hu, C. Fan, H. Jiang, X. Guo, Y. Gao, X. Lu, and T. L. Lam, *Boosting light-weight depth estimation via knowledge distillation,* arXiv preprint arXiv:2105.06143 (2021).

[31] M. K. Yucel, V. Dimaridou, A. Drosou, and A. Saa-Garriga, *Real-time monocular depth estimation with sparse supervision on mobile,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021) pp. 2428–2437.

[32] Y. Wang, X. Li, M. Shi, K. Xian, and Z. Cao, *Knowledge distillation for fast and accurate monocular depth estimation on mobile devices,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021) pp. 2457–2465.

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

[34] M. Hosseinzadeh, R. Fahimi, Y. Wang, *et al., Unsupervised learning of camera pose with compositional re-estimation,* in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2020) pp. 11–20.

[35] W. Wang, D. Zhu, X. Wang, Y. Hu, Y. Qiu, C. Wang, Y. Hu, A. Kapoor, and S. Scherer, *Tartanair: A dataset to push the limits of visual slam,* in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2020) pp. 4909–4916.

[36] L. N. Smith and N. Topin, *Super-convergence: Very fast training of neural networks using large learning rates,* in *Artificial intelligence and machine learning for multi-domain operations applications*, Vol. 11006 (SPIE, 2019) pp. 369–386.

[37] Z. Zhang and D. Scaramuzza, *A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry,* in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2018) pp. 7244–7251.

[38] Y. Gal and Z. Ghahramani, *Dropout as a bayesian approximation: Representing model uncertainty in deep learning,* in *international conference on machine learning* (PMLR, 2016) pp. 1050–1059.

[39] B. Lakshminarayanan, A. Pritzel, and C. Blundell, *Simple and scalable predictive uncertainty estimation using deep ensembles,* Advances in Neural Information Processing Systems **30** (2017).

[40] Z. Huai and G. Huang, *Robocentric visual-inertial odometry,* The International Journal of Robotics Research **41**, 667 (2022).

[41] J. Sola, *Quaternion kinematics for the error-state kalman filter,* arXiv preprint arXiv:1711.02508 (2017).

[42] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, *Openvins: A research platform for visual-inertial estimation,* in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2020) pp. 4666–4672.

[43] 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)* (IEEE, 2018) pp. 2502–2509.

[44] Bitcraze, *Ai deck 1.1*, https://www.bitcraze.io/products/ai-deck/ (2023), accessed on 3rd April 2023.

[45] F. Conti, *Technical report: Nemo dnn quantization for deployment model,* arXiv preprint arXiv:2004.05930 (2020).

[46] A. Burrello, A. Garofalo, N. Bruschi, G. Tagliavini, D. Rossi, and F. Conti, *Dory: Automatic end-to-end deployment of real-world dnns on low-cost iot mcus,* IEEE Transactions on Computers , 1 (2021).

6

CONCLUSION

6.1. ANSWERS TO RESEARCH QUESTIONS

Chapters 2 to 5 show the research outcome with respect to the research questions raised in Chapter 1. The following presents the summarized answers to each question.

Research Question 1

How to design an ego-motion estimator that uses as little computing power as possible while maintaining acceptable accuracy?

Integration of inertial measurements for ego-motion estimation suffers from drifting over time due to the existence of sensor bias and noise. Vision measurement can correct for accumulated error at the cost of computationally heavy visual processing. Chapter 2 tackles this problem by involving an additional information source, a simple aerodynamic model, in ego-motion estimation. It achieves estimations with bounded errors in two out of three dimensions of attitude and velocity. The rest dimension is corrected at the frame rate by the vision information. The calculation of the epipolar constraints of the visual feature point correspondences takes the rotation estimation as known information and outputs relative pose. Utilizing the rotation estimation increases robustness and reduces the computation demand. Experiments show that the proposed estimator satisfies the balance of accuracy and efficiency, and the accuracy did not drop significantly after the image stream was cut.

Research Question 2

Can an ANN maintain the accuracy of inferring translation velocity from blurry images captured by a downward-facing camera while achieving real-time performance on an onboard processor and generalization to unknown environments?

Chapter 3 shows that the network architecture of cascaded network blocks connected

by image warping is able to infer 3-d translational motion in real time and has better prediction accuracy than other architectures. The training dataset is made up of a large number of real-world scenes captured by a simulated camera in fast motion and with realistic motion blur. This dataset contributes to the network's generalization and robustness towards motion blur. Experiments show that the accuracy of network predictions is better than that of feature points when there is significant motion blur. Another finding is that self-supervised learning based on photometric error leads to better performance than using ground-truth labels. The proposed network demonstrates its practicality by running onboard an MAV in autonomous flights. The appearance of the environment is not included in the training dataset. Network-based ego-motion estimation is used for feedback flight control.

Research Question 3

How to train a planar homography network without using ground-truth labels, estimate the prediction uncertainty of the network, and build an efficient and accurate VIO upon it?

Based on the answer to Research Question 2, it is evident that cascaded network blocks can be trained in a self-supervised fashion and exhibit robustness towards motion blur. In Chapter 4, the self-supervised loss function for the 8-d planar homography transformation network is based on image synthesis by differentiable image warping, similar to the loss function in Chapter 3. To enable the network to estimate its prediction uncertainty, more output neurons are required for the uncertainty related to input noise. Dropout is implemented for the last two layers to estimate the uncertainty reflecting ignorance of the ideal network model. First, a big-size self-supervised network is trained to produce predictions close to the ground truth. Its predictions then act as the learning targets of the smaller-size uncertainty-aware network. With high-quality uncertainty estimation, the homography network output can be fused with inertial measurements by an EKF. The homography transformation used in the EKF updating is about only two temporally consecutive frames instead of considering more frames in a sliding window, as in many VIO solutions. As a result, the back-end processing is highly efficient. The total time consumption of the proposed VIO processing one frame is 25~29 milliseconds on a mobile processor. The evaluations show that the proposed VIO generalizes to real-world datasets, and its accuracy is on-par with state-of-the-art VIO methods.

Research Question 4

How to obtain accurate network predictions of pose and depth without ground truth and use them to train a lightweight uncertainty-aware pose network and a lightweight depth network for a forward-facing camera?

The answer to Research Question 3 reveals that in order to train an uncertainty-aware network without using ground truth, a big-size self-supervised network with high prediction accuracy is required. Different from planar scenes where camera motion is encoded in the homography transformation, for general 3-d structures, depth information is neces-

sary for view synthesis besides the camera pose. In the context of simultaneously learning pose and depth, in Chapter 5, iterative pose network inference based on depth-dependent view synthesis is adopted in the joint training of a pose network and a monocular depth network. The iteratively refined pose predictions allow for the training of a lightweight uncertainty-aware pose network using negative log likelihood loss. As for the lightweight depth network, a combination of pose-dependent photometric matching loss and knowledge distillation loss from the big-size depth network leads to better accuracy. Note that given the current small network sizes, the lightweight networks for depth and pose have limited accuracy. Experiments show that they merely meet the basic expectations when the tasks are simple. More research is required to reach better performance.

After answering the above research questions, let us look back to the general research goal:

Research Goal

To develop real-time onboard-processing visual ego-motion estimation solutions for autonomous MAVs deployable in unknown environments. The visual processing must maintain robustness during agile maneuvers.

This dissertation investigates approaches to robust ego-motion estimation in agile maneuvers for an autonomous MAV with limited onboard processing resources. In terms of efficiency, all four solutions are suitable for real-time processing on an onboard processor. Three out of four solutions (Chapters 2 to 4) utilized additional information about the platform and environment in addition to raw sensor measurement to mitigate the harm to accuracy from the shortage of computational power. For learning-based solutions (Chapters 3 to 5), we examined the generalization capacities by cross-dataset testing. In terms of robustness, motion blur is set as the main challenge in two solutions (Chapters 3 and 4) and is proven to be well tackled. In summary, the solutions examined by experiments support the conclusion that the research goal is basically achieved.

6.2. DISCUSSION

Before reaching the end, let us return to the starting point of this research journey and confront the ultimate question again. Is visual ego-motion estimation for MAV a solved problem? In other words, how far are we from the ideal solution for ego-motion estimation of a lightweight and agile MAV?

Indeed, the approaches proposed in this dissertation show promising performance in the testing scenarios for which they are designed. However, gaps exist between the current stage and the ideal solution. In the following, they are elaborated on in four aspects. Potential solutions are also discussed.

6.2.1. COMPUTATIONAL DEMAND

Artificial neural networks (ANNs) with more layers tend to produce better accuracy, yet increase the computational demand. Although GPU accelerates network inference, GPU itself is a challenge for a lightweight MAV. Deep network [1] or iterative network inferences [2] can run in real time only on standard-size GPUs designed for desktop computers. The

lightweight networks in this dissertation are affordable for a mobile GPU (Nvidia Jetson TX2) that can be carried by a middle-size quadrotor MAV. But when it comes to smaller MAVs, for example, the 72-gram racing MAV from [3], the networks already quickly become computationally too heavy.

In Chapters 3 and 4, the networks predict the planar homography transformation that provides enough information for view synthesis of a planar surface. So the prediction can be iteratively refined by multiple network inferences and achieves better accuracy. In Chapter 5, similar iterative refinement is also implemented for pose estimation using a forward-facing camera. But it is not adopted for onboard deployment given the fact that the dense depth map is required for view synthesis. Depth network inferencing means more computational demand onboard.

A potential way to run a deep network onboard a lightweight MAV is to further research network acceleration technologies at both the software and hardware level. For instance, training a network to have higher activation sparsity [4] can lead to a significant decrease in the computation of an event-driven processor [5].

6.2.2. SCALE AMBIGUITY

In the case of monocular vision, the scale is ambiguous. Pose networks in most literature are trained to memorize the metric scale of the training set and place a pitfall for generalization. When the network predicts the scale-less 2-D translation direction, stationary or pure rotation motion cannot be well handled.

The accelerometer is supposed to recover the metric scale in Chapter 5. But the accuracy is less good than expected. A better-designed VIO back-end than a basic EKF may improve accuracy. Another approach is to use optical flow as the constraint in the optimization of camera motion [2]. Both approaches have the cost of much higher computational demand. Utilizing the kinematic or dynamics model of the platform may be a way to introduce additional information about the metric scale at a relatively low computational cost.

6.2.3. NETWORK UNCERTAINTY ESTIMATION

In Chapter 4, it is shown that the network uncertainty estimation approaches reflect the accuracy of network prediction well in general. Although small in number, outliers of uncertainty estimation exist. There is no well-accepted theoretical guarantee of the reliability of the current uncertainty estimation approaches yet. More insights into the general problem of the uncertainty of artificial neural networks can be helpful not only for visual ego-motion estimation but also for almost all ANN applications.

6.2.4. ROBUSTNESS TOWARDS MOTION BLUR

In Chapter 3 and 4, the networks are trained with blurry images and show the robustness towards motion blur in testing. The network can output reasonable predictions when only a few handcrafted feature points can be reliably detected or tracked. However, this is far from enough to conclude that ANNs are immune to motion blur. More research is required to reveal the key element that affects blur robustness. Additionally, because motion blur is a function of camera motion and scene depth, it is also interesting to study how to recover camera motion from motion blur in a single image [6].

REFERENCES

[1] W. Wang, Y. Hu, and S. Scherer, *Tartanvo: A generalizable learning-based vo,* in *Conference on Robot Learning* (PMLR, 2021) pp. 1761–1772.

[2] Z. Teed and J. Deng, *Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras,* Advances in neural information processing systems **34**, 16558 (2021).

[3] S. Li, E. van der Horst, P. Duernay, C. De Wagter, and G. C. de Croon, *Visual model-predictive localization for computationally efficient autonomous racing of a 72-g drone,* Journal of Field Robotics **37**, 667 (2020).

[4] M. Kurtz, J. Kopinsky, R. Gelashvili, A. Matveev, J. Carr, M. Goin, W. Leiserson, S. Moore, N. Shavit, and D. Alistarh, *Inducing and exploiting activation sparsity for fast inference on deep neural networks,* in *International Conference on Machine Learning* (PMLR, 2020) pp. 5533–5543.

[5] A. Yousefzadeh, G.-J. Van Schaik, M. Tahghighi, P. Detterer, S. Traferro, M. Hijdra, J. Stuijt, F. Corradi, M. Sifalakis, and M. Konijnenburg, *Seneca: Scalable energy-efficient neuromorphic computer architecture,* in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (IEEE, 2022) pp. 371–374.

[6] S. Dai and Y. Wu, *Motion from blur,* in *2008 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2008) pp. 1–8.

ACKNOWLEDGEMENTS

On the morning of 18th September 2018, I landed at Schipol airport in an Airbus 380 aircraft operated by China Southern airline. At the end of 2022, I read the news that the aircraft had been retired by the airline. Time flies, especially during the Covid period. I realized that it had been four and a half years I had been living and working in the Netherlands as a PhD candidate. I said to myself it might be time that I should try to finish my PhD, though I was not satisfied with my research outcome and struggling to trade off my efforts in two unfinished projects. At the beginning of this Acknowledgment, I would like to thank myself for accepting that sometimes "good enough" is good enough.

Secondly, I sincerely appreciate my promotors, Guido de Croon and Christophe De Wagter. As my promotor and daily supervisor, Guido accompanied me throughout my PhD career with his excellent research skill and visionary guidance. Almost every time we had a project meeting, I was impressed that Guido not only understood the big picture but also located details and gave feasible suggestions for my next step. I appreciate his efforts in advancing my research progress and the detailed feedbacks on my every scientific writings. Guido is open-minded towards research topics. I started researching deep learning under his advice and happily found interest in it. I appreciate Guido's efforts in arranging my external research stay with IMEC the Netherlands. This experience opened my door to neuromorphic, a boosting research field. What is also worth mentioning is that he has a great sense of humor that relaxed me during research discussions. Christophe has extremely rich experience in designing flying robots. I thank him for helping with the drone platforms I used in my research. I also appreciate his insightful comments on this dissertation and the propositions. My promotors set examples as superior lifelong researchers. I have always been impressed by their endless passion for learning and open-mind thinking.

Thirdly, I would like to thank my colleagues I directly worked with. Shuo Li, as a senior colleague, guided me through the details of autonomous drone racing. Sihao Sun supported me mentally and technically during the competition in Macau. I had my first long-term teamwork with Nilay Sheth and Federico Paredes-Vallés. I gained robust oral English communication skills and my first international friendship from them. Later, I worked with Julien Dupeyroux, Jesse Hagenaars, Stein Stroobants, and Federico on the neuromorphic project. It is a good memory of having work dinners together. Special thanks to Julien, who influenced me positively in both work and life. And I thank his invitation to his wonderful Christmas dinner in the middle of Covid time. I thank Cheng Liu for being a great teammate in the drone obstacle avoidance project, in tennis and snow skiing as well. I also thank Ziqing Ma for her help in the above-mentioned activities. During my research

stay at IMEC the Netherlands, I worked with Manolis Sifalakis, Amirreza Yousefzadeh, and Guangzhi Tang. I appreciate their support and guidance when I was diving into a new knowledge domain.

It is my honor to work in the MAVLab. Here I worked with a group of researchers and engineers who have rich knowledge and a great passion for flying robots. I can always gain new knowledge from our meetings and discussions. MAVLab gives me a sense of belonging. I would like to thank the MAVLab members that have not been mentioned before: Kimberly McGuire, Kirk Scheper, Mario Coppola, Diana Olejnik, Tom van Dijk, Shushuai Li, Matěj Karásek, Sven Pfeiffer, Stavrow Bahnam, Robin Ferede, Matthew Yedutenko, Sunyou Hwang, Ewoud Smeur, Alessandro Mancinelli, Hang Yu, Yilun Wu, Salua Hamaza, Sunyi Wang, Liming Zheng, Anton Bredenbeck, Jane Ramirez, Chaoxiang Ye, Erik van der Horst, Freek van Tienen, and Dennis van Wijngaarden. My special thanks to Erik for his help on my first Ubuntu installation and drone repair, and to Shushuai who helped me with embedded hardware and dissertation writing.

In addition, I want to thank all the PhD students at the C&S department, for our leisure time such as coffee breaks, lunch time, BBQs, PhD drinks, PhD events, etc. We had fun and learned from each other in brainstorming a variety of research topics. My thanks go to: Dirk van Baelen, Jaime Junell, Malik Doole, Noor Nabi, Daniel Friesen, Jelmer Reitsma, Bo Sun, Jerom Maas, Isabel Metz, Sarah Barendswaard, Paolo Scaramuzzino, Dyah Jatiningrum, Annemarie Landman, Junzi Sun, Ye Zhang, Yingzhi Huang, Ying Yu, Marta Ribeiro, Wei Fu, Xuerui Wang, Rowenna Wijlens, Gijs de Rooij, Yifei Li, Yiyuan Zou, Tiago Monteiro Nunes, Jan Groot, Jiayu Chen, and Wenying Lyu. And thank you, Max Mulder and Bertine Markus, for leading our section and running the administration. I also wish to thank my friends outside the C&S department. They are: Jingwei Dong, Chengpeng Jiang, Xiujie Shan, Yi Zhang, Zhou Nie, Xiaohuan Lyu, Yujie Tang, Jing Chang, Langqing Sun, Ze Chang, Rui Feng, Haicheng Liu, Kai Wu, Jingyi Liu, and Wenxiu Wang. I sincerely thank my girlfriend, Cai Huang, who accompanied and supported me through the hardest time of my PhD career.

Before the end, I express my deepest gratitude to my parents, Wenying Li and Jun Xu, for their selfless love and support.

CURRICULUM VITÆ

Yingfu XU 徐英夫

09-01-1994	Born in Tieling, China.

EDUCATION

2009–2012	Tieling High School
2012–2016	B.Sc. in Flight Vehicle Design and Engineering Harbin Institute of Technology
2016–2018	M.Sc. in Aeronautical and Astronautical Science and Technology Harbin Institute of Technology
Since 2018	Ph.D. candidate in Aerial Robotics Delft University of Technology

AWARDS

2015	Merit Student of Heilongjiang Province
2018	Distinguished Master Student

LIST OF PUBLICATIONS

7. Guangzhi Tang, Kanishkan Vadivel, **Yingfu Xu**, Refik Bilgic, Kevin Shidqi, Paul Detterer, Stefano Traferro, Mario Konijnenburg, Manolis Sifalakis, Gert-Jan van Schaik, Amirreza Yousefzadeh, *SENECA: building a fully digital neuromorphic processor, design trade-offs and challenges*, Frontiers in Neuroscience 17 (2023).

6. Federico Paredes-Vallés, Jesse Hagenaars, Julien Dupeyroux, Stein Stroobants, **Yingfu Xu**, Guido C.H.E. de Croon, *Fully neuromorphic vision and control for autonomous drone flight*, Under review.

5. Cheng Liu*, **Yingfu Xu***, Erik-Jan van Kampen, Guido C.H.E. de Croon, *Nano Quadcopter Obstacle Avoidance with a Lightweight Monocular Depth Network*, Accepted by the 22nd World Congress of the International Federation of Automatic Control (2023). * Equal contribution.

4. **Yingfu Xu**, Guido C.H.E. de Croon, *A Lightweight Learning-based Visual-Inertial Odometry*, Accepted by the International Micro Air Vehicle Conference and Competition (IMAV), 2023.

3. **Yingfu Xu**, Guido C.H.E. de Croon, *CUAHN-VIO: Content-and-Uncertainty-Aware Homography Network for Visual-Inertial Odometry*, Under review.

2. **Yingfu Xu**, Guido C.H.E. de Croon, *CNN-based Ego-Motion Estimation for Fast MAV Maneuvers*, In 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 7606-7612. IEEE, 2021.

1. **Yingfu Xu**, Guido C.H.E. de Croon, *Efficient Model-Aided Visual-Inertial Ego-Motion Estimation for Multirotor MAVs*, Accepted by the International Micro Air Vehicle Conference and Competition (IMAV), 2023.