CUAHN-VIO

Content-and-uncertainty-aware homography network for visual-inertial odometry

Xu, Yingfu; de Croon, Guido C.H.E.

DOI
10.1016/j.robot.2024.104866

Publication date
2025

Document Version
Final published version

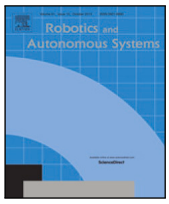
Published in
Robotics and Autonomous Systems

Robotics and Autonomous Systems 185 (2025) 104866

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



CUAHN-VIO: Content-and-uncertainty-aware homography network for visual-inertial odometry

Yingfu Xu *, Guido C.H.E. de Croon

Micro Air Vehicle Lab, Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, Delft, 2629HS, The Netherlands

ARTICLE INFO

Keywords:
Deep homography
Self-supervised learning
Uncertainty estimation
Visual-inertial odometry
Micro air vehicle

ABSTRACT

Learning-based visual ego-motion estimation is promising yet not ready for navigating agile mobile robots in the real world. In this article, 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 frontend is a content-and-uncertainty-aware homography network (CUAHN). Content awareness measures the robustness of the network toward 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 requires no 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 backend 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 (~23 ms), and stable processing latency (~26 ms), CUAHN-VIO successfully runs onboard an Nvidia Jetson TX2 embedded processor to navigate a fast autonomous MAV.

1. Introduction

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 [2–7] detect and track handcrafted feature points along image frames. Direct approaches [8–10] directly utilize the photometric intensities of pixels. Hybrid approaches [11,12] combine both. Although such approaches has been widely recognized, their vision frontends 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 [13–18], 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 [16–21]. 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 [14,22–29]. 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 [26], 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 [27,28] or videos with synchronized IMU data streams [14]. 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

* Corresponding author.
E-mail addresses: yingfu.xu.94@gmail.com (Y. Xu), g.c.h.e.decroon@tudelft.nl (G.C.H.E. de Croon).

https://doi.org/10.1016/j.robot.2024.104866
Received 3 August 2024; Received in revised form 23 October 2024; Accepted 16 November 2024
Available online 22 November 2024

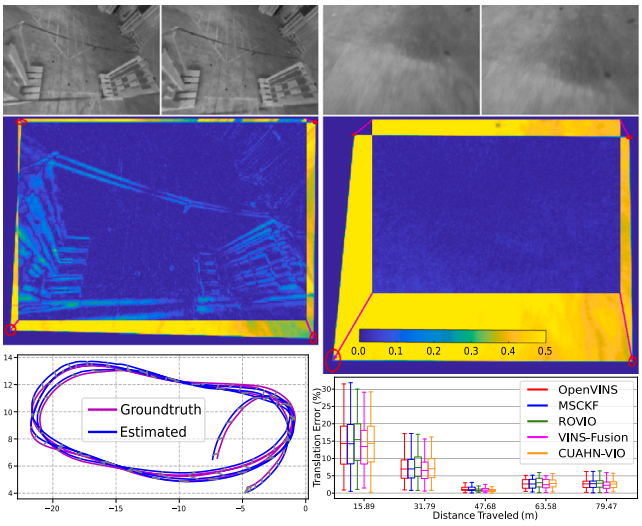


Fig. 1. Visualized outputs of CUAHN-VIO evaluated on Seq. 13 of the UZH-FPV dataset [1]. CUAHN is robust to sparse non-planar objects and motion blur. Example image pairs are respectively shown on the left and right of the first row. The colormaps (2nd row) show the photometric error between the current 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. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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 [13,15,19] 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 [30], a car dataset with 3-DoF motion. When it comes to a smaller number of training samples and more difficult motion patterns, networks [14, 17,18,20,21] show worse accuracy than traditional approaches on EuRoC [31], 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) backend lead to better accuracy than constant hand-tuned uncertainty [17,28].

The last challenge is high computation time. The causes are, for instance, the network being too deep [17], combining multiple networks that together are very large [29], or using an expensive intermediate representation such as a dense optical flow map [13,16]. Works [13,16, 17,29] reported network inference time of more than 40 ms measured on Nvidia GPUs designed for desktop computers.

In this article, we propose CUAHN-VIO that overcomes the three challenges to a large extent. Instead of *PoseNet*, the vision frontend is a network that predicts the planar homography transformation. It is a pixel-level task and thus generalizes 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. CUAHN-VIO requires the MAV to fly above mostly planar ground, which makes it less generic than other solutions that suit a forward-facing camera. But this requirement is not strict as shown by the experiments, which makes it suitable for most indoor environments.

In our previous work [15], 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 work aims to establish a complete VIO system with a network-based vision frontend to pursue accurate ego-motion estimation in agile maneuvers of MAVs. The network, CUAHN, is more capable than the translation networks in [15], featured by that CUAHN predicts homography transformation that encodes 6-DoF camera motion, and CUAHN estimates the prediction uncertainty.

The main contributions of this work can be summarized as:

- We propose a practical scheme of training a planar homography prediction network in a self-supervised fashion. 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 backend. 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.

2. Related works

2.1. Learning-based visual ego-motion estimation

For *PoseNets* learning to predict 3-DoF translational motion in metric scale from monocular video [16,17,19,27,28], 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 [13] 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 [13], 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.

DROID-SLAM [32] adopts an optical flow network that iterative refines its predictions. The predicted flow maps are used as constraints for a BA backend that optimizes camera poses. 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. It is a computationally demanding system that requires a powerful NVIDIA GeForceRTX-3090 GPU for tracking and local BA to process the downsampled monocular video of EuRoC dataset at > 10 fps.

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 [18,20]. 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. Practicing this idea, an end-to-end supervised learning scheme for a loosely-coupled VIO is proposed in [21]. Its backend is a differentiable EKF whose states are propagated by integrating IMU measurements.

For self-supervised learning, SfMLearner [22] firstly proposed to simultaneously train two networks that respectively predict $T_{t\to s}$ and $\mathcal{D}_t$. $T_{t\to s}$ is the relative pose between source image $\mathcal{I}_s$ and target image $\mathcal{I}_t$. $\mathcal{D}_t$ is the pixel-wise depth map of $\mathcal{I}_t$. An image $\widetilde{\mathcal{I}}_s$ can be synthesized by warping $\mathcal{I}_s$ according to the 2-d projections of the 3-d point cloud established from $\mathcal{D}_t$ on the image plane of $\mathcal{I}_s$ 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 $\widetilde{\mathcal{I}}_s$ and $\mathcal{I}_t$. We call it reprojection-based loss for simplicity. This scheme was further developed by also predicting $\mathcal{D}_s$ and punishing the 3-d geometric inconsistency between $\mathcal{D}_t$ and $\mathcal{D}_s$ [25,26].

Also using reprojection-based loss, SelfVIO [29] 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 [18,20]. IMU measurements bring in motion information with metric scale, however, as pointed out in [14], 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 [21], the *PoseNet* prediction in [14] 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.

2.2. Network uncertainty estimation in computer vision tasks

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 how 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 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 article 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 [14,17,21,40,41], monocular depth [33,40,42], optical flow [43], semantic segmentation [33], and image classification [36]. Recently, the authors of [44] utilize the predictive uncertainty of optical flow to navigate a flying robot by detecting obstacles, gaps, *etc.* 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 [45].

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 [28] leverages the predictive uncertainty. The uncertainty map of photometric matching acts as the weights of the photometric energy in the BA backend. The relative pose network of D3VO has no uncertainty estimation, so the weights in the optimization of pose energy are set as constant. In [17], 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. [21] 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.

The proposed approach of [46] estimates the empirical uncertainty of pose predictions by MC-Dropout and the predictive uncertainty by additional layers attached to the *PoseNet*. The uncertainty estimation is utilized in fusing the pose predictions with the pose estimation from a traditional geometric VO by a Kalman filter. It is not shown how to use the proposed network itself for robot navigation. Regarding time efficiency, the network inference achieves more than 90 Hz running on a big GPU for desktop computers. The authors claim the possibility of achieving real-time performance on embedded processors but no experimental data is shown. We are cautiously skeptical, because of the heavy optical flow encoding network and the fact that multiple image pairs (5 in [46]) are required to be processed for a single pose prediction, due to the long short-term memory (LSTM) layers that model temporal dependencies across consecutive image pairs. [46] performs supervised learning, and the training and testing are conducted on the same dataset. All of these facts make its capacity to deploy onboard robots questionable.

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 [14] learns predictive uncertainty of relative camera pose from the error of *a posteriori* ego-motion, same as the supervised VIO [21]. 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.

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 [47–49] and self-supervised [50,51] 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 [15,48,49]. 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 [52] 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 [49], 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. [51] 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.

3. System overview

Fig. 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 frontend, a DNN (Fig. 5), predicts the planar homography transformation and the uncertainty. They are utilized in updating the EKF backend as shown in Fig. 3.

Learning-based VIO approaches [14,18,20,21,29] 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 [14].

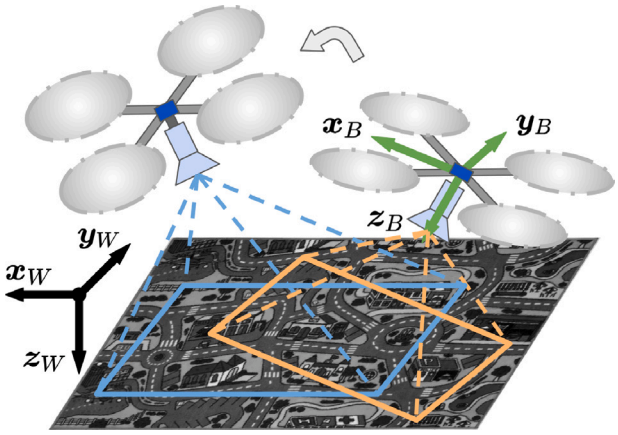


Fig. 2. An overview illustration of the application scenario, sensor setup, and coordinate definition. *W* stands for world frame and *B* stands for body frame, *i.e.*, IMU frame.

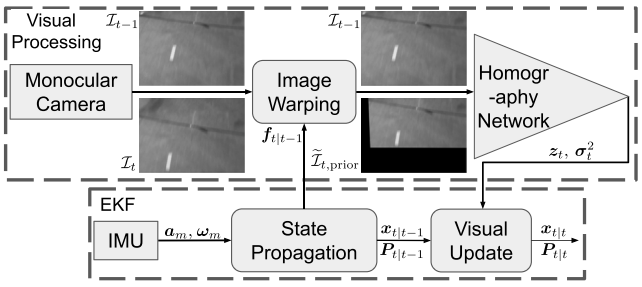


Fig. 3. An overview data flow diagram of CUAHN-VIO.

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 backend 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 (Section 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 frontend 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 (Section 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 (Section 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 (Section 5.2). The student network estimates empirical uncertainty by deep ensembles or MC-Dropout (Section 5.3). Uncertainty estimation turns out to be important in improving the VIO accuracy.

The backend of CUAHN-VIO is a simple extended Kalman filter (EKF), as shown in Fig. 3 and introduced in detail in Section 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

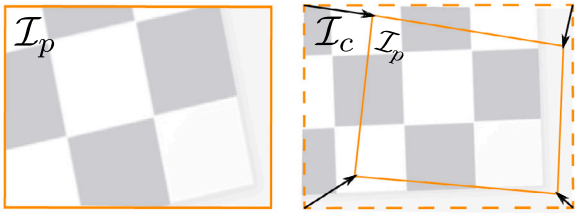


Fig. 4. An example of 8-d corner flow $\boldsymbol{f}$.
Source: Images are adapted from [53].

$\boldsymbol{f}_{t|t-1}$ is utilized for pre-warping the current image $\mathcal{I}_t$. The new image $\widetilde{\mathcal{I}}_{t,\text{prior}}$ synthesized by warping is more similar to the previous image $\mathcal{I}_{t-1}$ 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. Planar homography network

4.1. Datasets

The training dataset is the same as [15]. 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 article.

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. 6. This dataset is called the MYNT Dataset.

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

4.2. Self-supervised cascaded network blocks

When the network acts as a VIO vision frontend, its input images are temporally consecutive, so we refer to them as the previous image $\mathcal{I}_p$ and the current image $\mathcal{I}_c$. The homography transformation is parameterized as four 2-d optical flow vectors in the image plane of $\mathcal{I}_c$. They point from the image corners to the pixels corresponding to the corners of $\mathcal{I}_p$, as illustrated in Fig. 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 [47–50] and it shows better performance than 3 × 3 homography matrix when using traditional methods [54].

As shown in Fig. 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 $\mathcal{I}_p$ and $\mathcal{I}_c$ and regresses to $\boldsymbol{f}_1$. The direct linear transformation (DLT[1]) solver calculates the homography matrix $\boldsymbol{H}_1$ from $\boldsymbol{f}_1$. The correspondence between the float pixel coordinate (u_c, v_c) in $\mathcal{I}_c$ and the integer pixel coordinate (u_p, v_p) in $\mathcal{I}_p$ is

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

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 $\mathcal{I}_c$ using differentiable bilinear interpolation [52]. The synthesized image is referred to as $\widetilde{\mathcal{I}}_{c,1}$, the warped $\mathcal{I}_c$ according to $\boldsymbol{f}_1$. $\widetilde{\mathcal{I}}_{c,1}$ is then downsampled and concatenated with the downsampled $\mathcal{I}_p$ 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 $\widetilde{\mathcal{I}}_{c,1}$ to the pixels in $\widetilde{\mathcal{I}}_{c,1}$ that have the same intensities as the corners of $\mathcal{I}_p$. $\boldsymbol{H}_2$ 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 $\mathcal{I}_c$ to synthesize $\widetilde{\mathcal{I}}_{c,2}$, 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 $\mathcal{I}_c$ and $\mathcal{I}_p$. It is obtained by

$$\lambda(\boldsymbol{f}_{\text{total},j} + \boldsymbol{c}_{j,\mathcal{I}_c}) = \boldsymbol{H}_{\text{integ.},3}(\boldsymbol{f}_{4,j} + \boldsymbol{c}_{j,\widetilde{\mathcal{I}}_{c,3}}), \tag{2}$$

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

The similarity between $\widetilde{\mathcal{I}}_{c,i}$ and $\mathcal{I}_p$ indicates how accurate is $\boldsymbol{H}_{\text{integ.},i}$. The self-supervised loss function is established as Eqs. (3) and (4).

$$\mathcal{L}_i = \frac{1}{|V|}\sum_{k\in V}\mathcal{L}(\boldsymbol{I}_{i,k}) \tag{3}$$

$$\mathcal{L}(\boldsymbol{I}_{i,k}) = \frac{\alpha}{2}(1 - \text{SSIM}(\boldsymbol{I}_{i,k})) + (1-\alpha)\cdot|\mathcal{I}_{p,k} - \widetilde{\mathcal{I}}_{c,i,k}| \tag{4}$$

[1] An asterisk indicates that further elaboration is available in *Supplementary Document*. Applying to the whole article.

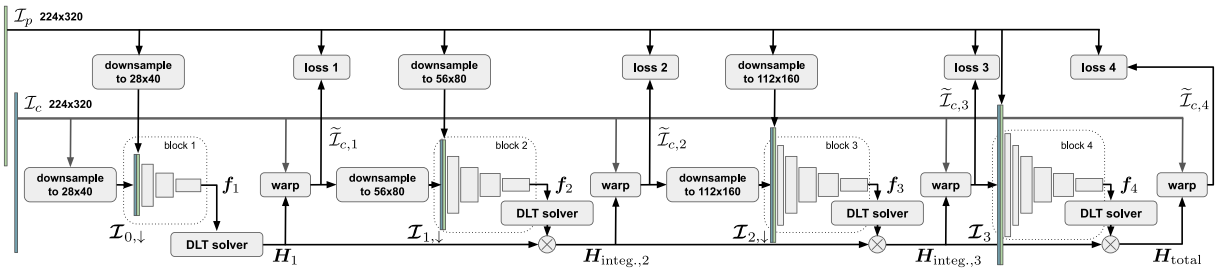


Fig. 5. The architecture of cascaded network blocks for planar homography transformation prediction. The downward arrow in the foot marker of $\mathcal{I}_{i,\downarrow}$ indicates that it has been downsampled. Data flows correspond to training. In inference, loss terms are not calculated and there is no DLT solver for $\boldsymbol{f}_4$. The network output $\boldsymbol{f}_{\text{total}}$ is obtained by Eq. (2).

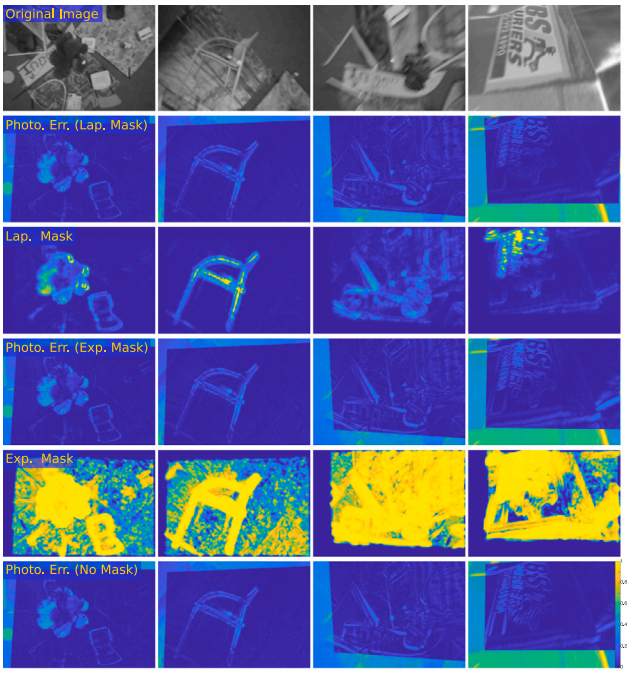


Fig. 6. From the top row to the bottom row: original image, photometric error maps of the network trained by Eq. (7), uncertainty maps (b_k) in Eq. (7), photometric error maps of the network trained by Eq. (5), explainability maps (E_k) in Eq. (5) ($\lambda_{\text{reg.}}$ =2e-3), and photometric error maps of the baseline network. The photometric error map is made of $|\mathcal{I}_{p,k} - \widetilde{\mathcal{I}}_{c,k}|$, where $\widetilde{\mathcal{I}}_c$ is the warped $\mathcal{I}_c$ 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 a 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. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

V denotes the set of all valid pixels excluding the ones sampled outside the border of $\mathcal{I}_c$. It is a mask calculated from $\boldsymbol{H}_{\text{integ.},i}$. $\boldsymbol{\mathcal{I}}_i$ denotes the aggregation of $\mathcal{I}_p$ and $\widetilde{\mathcal{I}}_{c,i}$. Same as other works [28,40,42,55], 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) where $\alpha = 0.85$. Multi-stage losses are calculated from each $\boldsymbol{\mathcal{I}}_i$. Their weights are respectively 0.1, 0.2, 0.3, and 0.4 from earlier to later blocks.

The planar homography network is implemented[1] 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 [47,49,51], 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++[1], measured on a TX2 processor in Max-P ARM power mode.

4.3. Content-aware learning

The major assumptions made in the self-supervised loss function Eq. (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-like [56] 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 $\boldsymbol{\mathcal{I}}_3$. 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) 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 [22]. 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 $\mathcal{I}_p$. The pixel-wise loss is weighted by E_k as shown in Eq. (5). A regularization term $\mathcal{L}_{\text{reg.}}(E_k)$ 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 $\mathcal{L}(\boldsymbol{\mathcal{I}}_{4,k})$ would be fully minimized. On the contrary, $\mathcal{L}(\boldsymbol{\mathcal{I}}_{4,k})$ would be ignored if E_k is close to zero.

$$\mathcal{L}_{\text{CA,Exp.}} = \frac{1}{|V|}\sum_{k\in V} E_k \cdot \mathcal{L}(\boldsymbol{\mathcal{I}}_{4,k}) + \lambda_{\text{reg.}} \cdot \mathcal{L}_{\text{reg.}}(E_k) \tag{5}$$

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 [28,40] and optical flow estimation [43]. Given the Laplacian probability density function (PDF)

$$p(x|\mu, b) = \frac{1}{2b} e^{\frac{-|x-\mu|}{b}}, \tag{6}$$

since we use $L1$ loss in Eq. (4), the term $|x - \mu|$ can be replaced by photometric matching loss $\mathcal{L}(\boldsymbol{\mathcal{I}}_{4,k})$ calculated from the homography prediction. The parameter b in Eq. (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 $\boldsymbol{\mathcal{I}}_4$. The learning objective is to maximize the PDF, *i.e.*, minimize the NLL loss

$$\mathcal{L}_{\text{CA,Lap.}} = \frac{1}{|V|}\sum_{k\in V} \frac{\mathcal{L}(\boldsymbol{\mathcal{I}}_{4,k})}{b_k} + \log b_k. \tag{7}$$

b_k can be understood intuitively as the uncertainty of the indirectly predicted $\mathcal{L}(\boldsymbol{\mathcal{I}}_{4,k})$, *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. (7) allows the learning process to increase the value of b_k to down-weight $\mathcal{L}(\boldsymbol{\mathcal{I}}_{4,k})$ and reduce the overall loss. $\log b_k$ prevents b_k to overgrow.

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 1. The 2nd column is the baseline network without mask prediction and trained by the original loss function Eq. (3). The network of the 3rd column is trained by Eq. (7). The remaining three columns correspond to networks trained by Eq. (5) with different hyperparameters $\lambda_{\text{reg.}}$. Their pixel-wise average values of the predicted explainability maps

Table 1
Comparison of content-aware homography networks (CAHN). The unit of average error is pixel. The three columns on the right correspond to the networks predicting the explainability map E_k. The three values in the first row are the weights $\lambda_{\text{reg.}}$ of the regularize $\mathcal{L}_{\text{reg.}}(E_k)$.

Setups	No mask	Lap.	1e−3	2e−3	3e−3
Avg. E_k	–	–	0.383	0.640	0.779
Avg. Error ↓	0.8768	0.8477	0.8483	0.8471	0.8487

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.

The lowest errors are obtained by using the context-aware masks. However, given the considerable 3-d structure in the MYNT dataset, the difference with the baseline is quite modest. As shown in the 6th row of Fig. 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 this modest difference is that the baseline DNN trained with the original loss function Eq. (3) is already by itself rather robust to non-planar scenes.

As shown in Fig. 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 the uncertainty map trained by Eq. (7) is the better choice for content-aware learning.

The 2nd and 3rd rows of Fig. 6 show the positive correlation between the predicted uncertainty maps and the photometric error maps. The 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, a content-aware mask is not ideal for semantic plane segmentation. Its only duty is to down-weight the non-homography pixels in training.

5. Network uncertainty estimation

In Section 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. Uncertainty-Aware Homography Networks (UAHN) in this section are trained and evaluated on the Basic Dataset. Content-aware learning is not involved.

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 [14,17,21,39], the six elements are modeled as independent of each other. In this work, 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 $\widetilde{\mathcal{I}}_{c,3}$ and $\mathcal{I}_p$ 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 $\widetilde{\mathcal{I}}_{c,3}$. 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 $\mathcal{I}_c$ that has the same intensity as the endpoint of $\boldsymbol{f}_{4,j}$ can be obtained by Eq. (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

$$\lambda^2 \boldsymbol{\Sigma}_{\text{total},j} = \boldsymbol{H}_{\text{integ.},3} \cdot \boldsymbol{\Sigma}_{4,j} \cdot \boldsymbol{H}^T_{\text{integ.},3}. \tag{8}$$

Note that the λ here has the same value as the λ in Eq. (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. 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 article, 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.

5.2. Predictive uncertainty

As introduced in Section 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.

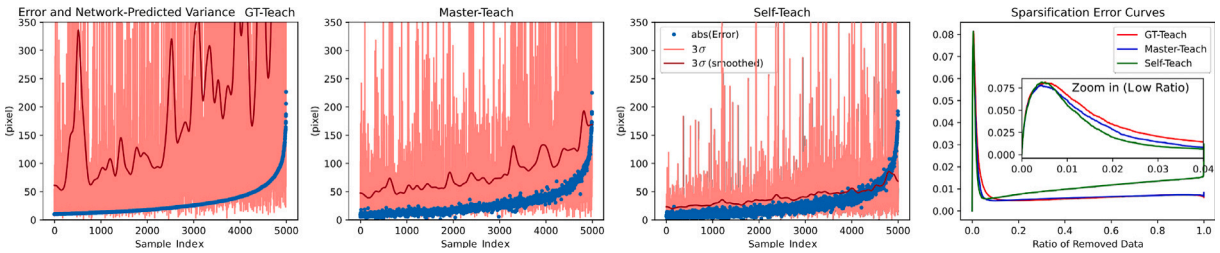


Fig. 7. Comparison of different approaches to learning predictive uncertainty. Left three subplots: three times of the predictive standard deviations σ and the corresponding prediction errors (absolute values) of GT-Teach, Master-Teach, and Self-Teach, sorted according to the errors of GT-Teach. The 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. Rightmost subplot: comparison of the sparsification error curves of the three supervision schemes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

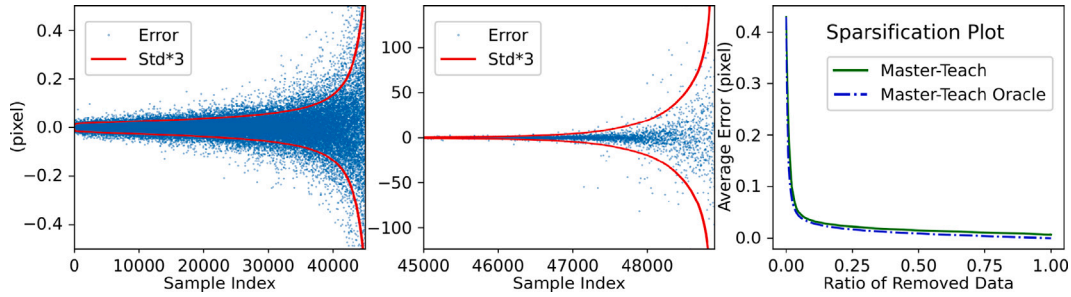


Fig. 8. Predictive uncertainty of the randomly initialized Master-Teach model (1st row of Table 3). Error-variance pairs of the testing set are sorted according to the variances. To avoid overly dense data points in plots, we show one point every ten pairs. Small and big variances are shown respectively in the left and middle subplots with different ranges of y-axis for better visualization.

Table 2
Comparison of supervision signals for predictive uncertainty.

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

[a] The networks predict rare unreasonably big variances. The averages are calculated after removing the 0.1% biggest 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

$$\mathcal{L}_{\text{Gaus.}} = \sum_{n=1}^{8} \frac{1}{2\sigma_n^2(\boldsymbol{\mathcal{I}}_3)} \|t_n - \mu_n(\boldsymbol{\mathcal{I}}_3)\|^2 + \frac{1}{2}\log\sigma_n^2(\boldsymbol{\mathcal{I}}_3). \tag{9}$$

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 $\boldsymbol{\mathcal{I}}_3$. 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. (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 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 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 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 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. 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 2. 3σ grows with error as a general trend.

Table 3
Comparison of different network initializations.

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

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 rightmost subplot of Fig. 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. 8 diagrams how the predictive variances cover the prediction errors in a different view from Fig. 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. 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 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 article are trained in Master-Teach scheme.

Besides random initialization, the student network can be initialized with the trained parameters of the Basic Model. It is clearly shown in Table 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.

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. (11) as the sum of the empirical variance of the mean value predictions $\mu_{m,n}$ and the average of predictive variances $\sigma_{m,n}^2$. m indexes over the network model samples.

$$\mu_n = \frac{1}{M}\sum_{m=1}^{M}\mu_{m,n} \tag{10}$$

$$\sigma_n^2 = \sigma_{n,\text{pred.}}^2 + \sigma_{n,\text{emp.}}^2$$
$$\sigma_{\text{pred.}}^2 = \frac{1}{M}\sum_{m=1}^{M}\sigma_{m,n}^2, \quad \sigma_{\text{emp.}}^2 = \frac{1}{M}\sum_{m=1}^{M}(\mu_{m,n}-\mu_n)^2 \tag{11}$$

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 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 [45], 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. 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 leftmost subplot shows that increasing the number of sampled network models only slightly improves prediction accuracy. While, the AUSE values shown in the 2nd subplot from left vary with the number of samples significantly, especially for deep ensembles. The same trend is observed in Inside Rate (3rd 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. 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 before dropout to obtain the same effect. The time consumption increases insignificantly as shown in the rightmost subplot of Fig. 9. Based on the overall consideration of

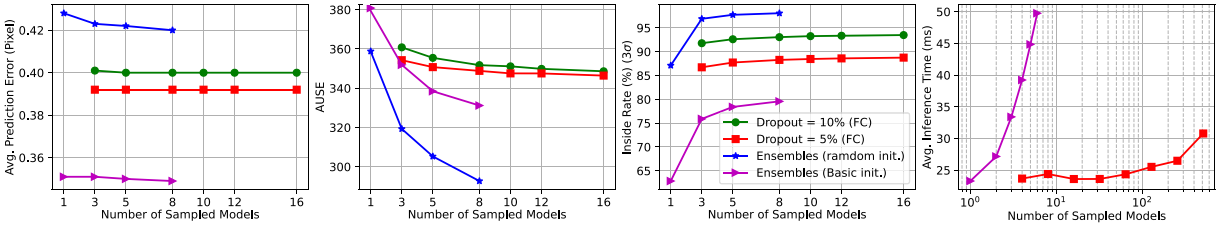


Fig. 9. Comparison of approaches for empirical uncertainty in terms of prediction accuracy, uncertainty estimation quality (AUSE and Inside Rate), and inference time consumption. The x-axis for deep ensembles is the number of independent network models. For MC-Dropout, the x-axis is the number of forward passes. Deep ensembles indicated by blue stars and magenta triangles correspond respectively to the 1st and 3rd row of Table 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. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the four metrics, we select three candidates that act as the VIO vision frontend and are compared in terms of the resulting VIO accuracy in Section 7.2. They are 1) the ensemble of randomly initialized models (indicated by the blue stars in Fig. 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).

Supplementary Document shows the magnitudes of $\sigma^2_{\text{emp.}}$ and $\sigma^2_{\text{pred.}}$ and their correlation for interested readers.

6. Visual-inertial odometry

6.1. Homography network as the vision frontend

We have introduced how to train CAHN (Section 4) and UAHN (Section 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 frontend 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. (7), different from the Master Model in Section 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 Section 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 article, 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 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 $k_{\text{var.}}$.

6.2. EKF-based backend

The VIO backend 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. 2. The EKF state vector is defined as:

$$\boldsymbol{x} := \left[\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{v}, \boldsymbol{b}_a, \boldsymbol{b}_g, \boldsymbol{f}_j\right], \quad j \in \{ul, bl, br, ur\}. \tag{12}$$

$\boldsymbol{p}$ 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. (13), IMU measurements are modeled 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 $\boldsymbol{w}_g$).

$$\boldsymbol{a}_m = \hat{\boldsymbol{a}} + \boldsymbol{b}_a + \boldsymbol{w}_a, \quad \boldsymbol{\omega}_m = \hat{\boldsymbol{\omega}} + \boldsymbol{b}_g + \boldsymbol{w}_g \tag{13}$$

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 [6].

$$\begin{aligned}
\dot{\boldsymbol{p}} &= -[\hat{\boldsymbol{\omega}}]_\times \boldsymbol{p} + \boldsymbol{v} + \boldsymbol{w}_p, \\
\dot{\boldsymbol{v}} &= -[\hat{\boldsymbol{\omega}}]_\times \boldsymbol{v} + \hat{\boldsymbol{a}} + \boldsymbol{R}(\boldsymbol{q})^{-1}\boldsymbol{g}, \\
\dot{\boldsymbol{q}} &= \frac{1}{2}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \hat{\boldsymbol{\omega}} \end{bmatrix}, \\
\dot{\boldsymbol{b}}_a &= \boldsymbol{w}_{\boldsymbol{b}_a}, \quad \dot{\boldsymbol{b}}_g = \boldsymbol{w}_{\boldsymbol{b}_g}, \\
\dot{\boldsymbol{f}}_j &= -(\boldsymbol{I} - (\boldsymbol{c}_j + \boldsymbol{f}_j)\boldsymbol{e}_z^T)\boldsymbol{H}(\boldsymbol{c}_j + \boldsymbol{f}_j)
\end{aligned} \tag{14}$$

Eq. (14) shows the IMU-driven state dynamics ($\dot{\boldsymbol{x}}$). $[\hat{\boldsymbol{\omega}}]_\times$ represents the skew-symmetric matrix associated with $\hat{\boldsymbol{\omega}}$. $\boldsymbol{w}_p$ 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 [57] for quaternion-related calculation. The propagation of $\boldsymbol{f}_j$ is based on the continuous homography transformation. The formula derivation can be found in [53]. $\boldsymbol{c}_j$ 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:

$$\boldsymbol{H} = [\hat{\boldsymbol{\omega}}_c]_\times + \frac{1}{d_c}\boldsymbol{v}_c\boldsymbol{\mu}_c^T \tag{15}$$

where

$$\begin{aligned}\hat{\boldsymbol{\omega}}_c &= \boldsymbol{R}_{CI}\hat{\boldsymbol{\omega}},\\ \boldsymbol{v}_c &= \boldsymbol{R}_{CI}(\boldsymbol{v} + [\hat{\boldsymbol{\omega}}]_\times \boldsymbol{t}_{IC}),\end{aligned} \tag{16}$$

and

$$\begin{aligned}\boldsymbol{\mu}_c^T &= \boldsymbol{R}_{CI}\boldsymbol{R}^{-1}(\boldsymbol{q})\boldsymbol{e}_z,\\ d_c &= -\boldsymbol{e}_z^T\boldsymbol{R}(\boldsymbol{q})(\boldsymbol{p} + \boldsymbol{t}_{IC}).\end{aligned} \tag{17}$$

$\hat{\boldsymbol{\omega}}_c$ 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. 2. In the cases where the z-axis points up, minus signs should be added to the right of the equal signs in Eq. (17).

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

$$\boldsymbol{z}_{j,t} = \boldsymbol{f}_{j,t|t-1} + \boldsymbol{w}_{j,t} \tag{18}$$

where $\boldsymbol{f}_{j,t|t-1}$ is the *a priori* estimation of $\boldsymbol{f}_j$ propagated by Eq. (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. 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.

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 [3,6], 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 [1]. 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 [30] 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 [31], 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 [58] 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 [58]. It reveals how well the scale of the estimated trajectory matches the metric scale. The *Sim3* alignment widely used by other works cannot.

Table 4
Comparison with SOTA VIO approaches OpenVINS [6], LARVIO [7], MSCKF [3,6], ROVIO [11], and VINS-Fusion [5]. **Bold** represents the best and underline represents the best of SOTA approaches.

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

[a] Without online calibration.

Table 5
Comparison with the learning-based DROID-SLAM [32]. The accuracy metric is calculated after *Sim3* trajectory alignment. **Bold** represents better.

VIO/VO	RMSE of Absolute Translation Errors (ATE) ↓					
	Seq. 2	Seq. 4	Seq. 9	Seq.12	Seq.13	Seq.14
6-3 of Table 7	**0.3355**	**0.2941**	**0.3382**	**0.5800**	**0.4057**	**1.7672**
DROID	1.3882	1.1096	1.4307	7.3421	1.2826	4.3682

7.1. Accuracy comparison with SOTA VIO

In Table 4, our approach is compared with the open-sourced SOTA VIO approaches. The tests are run on the six 45-degree downward-facing sequences of the UZH-FPV dataset. Based on the ablation study shown later, three variants of our approach (6–3, 6–4, and 4–4 in Table 7) are selected. 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 7. 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,[1] *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 [2] 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 DROID-SLAM [32], 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 DROID-SLAM and UAHN-VIO, as shown in Table 5. The preprocessing and frame rate of the input videos are the same for DROID-SLAM and UAHN. The global bundle adjustment of DROID-SLAM was disabled in our testing. Table 5 shows that UAHN has an advantage in accuracy over DROID-SLAM. Regarding time efficiency, we run DROID-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 DROID-SLAM did not run in real-time. As to be shown later, our approach ran in real-time

Table 6
Comparison with SOTA VIO approaches DVIO-Homo [10], MSCKF [3,6], ROVIO [11], and VINS-Fusion [5]. The seven testing sequences were collected by us and referred to as the MYNT-CyberZoo dataset. **Bold** represents the best and underline represents the second best. "-" means very big error in trajectory estimation.

VIO	RMSE (meter) of Absolute Translation Errors (ATE) ↓						
	hover	circle1	circle2	shuttle1	shuttle2	lowTex	varyLight
6-3 of Table 7 ($k_{\text{var.}} = 1$)	0.0153	0.2296	0.2773	0.0939	0.1996	0.4960	0.4894
6-3 of Table 7 ($k_{\text{var.}} = 5$)	0.0164	0.3885	0.3110	0.0903	**0.1402**	0.6727	0.5318
6-3 of Table 7 ($k_{\text{var.}} = 25$)	**0.0134**	0.4001	0.3023	**0.0860**	0.1790	0.6750	**0.4490**
DVIO-Homo	0.4148	0.4727	0.2067	0.6878	2.6687	0.6070	–
MSCKF	6.4930	**0.1819**	0.3071	0.1647	0.1989	**0.1824**	–
ROVIO	–	0.2097	–	0.3167	–	–	0.7231
VINS-Fusion	0.1054	0.7077	**0.1588**	0.2538	0.4036	1.8864	0.6508

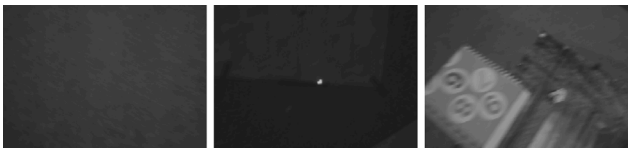


Fig. 10. Example images of the MYNT-CyberZoo dataset. From left to right, they are respectively from *lowTex*, *varyLight*, and *circle2*.

on a small-size mobile GPU processor. Thus our approach outperforms DROID-SLAM in terms of time efficiency.

Table 6 shows the comparison between our approach and SOTA VIO approaches on seven flight sequences collected in our flight arena CyberZoo. The MAV and sensor for data collection are the same as those introduced in Section 7.3. We refer to it as the MYNT-CyberZoo dataset. DVIO-Homo [10] assumes a single planar surface in the field of view. It estimates the ego-motion by an iterative EKF that matches pixel intensities between two frames according to the continuous homography. It is a good counterpart to compare with our approach because they both assume a single planar surface and utilize the planar homography. We also tried to run DVIO-Homo on the UZH-FPV dataset but it had big estimation errors. Tuning the parameters in the configuration file failed to produce significant improvement. Therefore DVIO-Homo is not shown in Table 4. In the *hover* sequence, the MAV took off and stayed still at a point just above the takeoff point. In *circle1* and *circle2*, the MAV had circle flight trajectories. In *shuttle1* and *shuttle2*, the MAV performed shuttle flights between two waypoints. For *lowTex*, the flight trajectory was also a circle but there were much less texture on the ground, as shown in Fig. 10. During the collection of *varyLight*, the illumination was varied by moving the curtain and turning on/off the lights. As shown in Table 6, our approach has the best or the second-best accuracy in five out of seven sequences of the MYNT-CyberZoo dataset.

6–3 of Table 7 is a variant of UAHN-VIO that is introduced in the next subsection. UAHN-VIO is trained on the Basic dataset, which is synthetic by simulation. Therefore, the flight arenas of the UZH-FPV and MYNT-CyberZoo datasets are strictly unknown to the DNN of UAHN-VIO, and tests on the two real-world datasets are out-of-domain tests. As shown in Table 4, 6–3 of Table 7 is more accurate than SOTA VIOs on five out of six sequences. 6–3 of Table 7 is also evaluated in Tables 5 and 6, and achieves competitive accuracy. These results demonstrate the good generalization capacity of our approach.

By Table 6, we study how the hyperparameter $k_{\text{var.}}$ affects accuracy. $k_{\text{var.}}$ scales the network estimation of measurement variance. It is the only tunable hyperparameter in the VIO backend. Table 6 shows that the accuracy does not vary much even if $k_{\text{var.}}$ is multiplied by five twice. This experimental result demonstrates the easy-to-tune nature of our approach.

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 7. The 2nd column shows the network acting as the vision frontend 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 Section 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 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. It is also demonstrated in Table 6 that varying $k_{\text{var.}}$ affects the accuracy to a very small extent.

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).

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 Section 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 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. (3) instead of Eq. (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 Section 4.3.

Table 7
Evaluation of VIO variants by indoor 45-degree downward-facing sequences of UZH-FPV. **Bold** represents the best result.

ID	Network	Initial-ization	Num. of Blocks	$\boldsymbol{R}_{\text{meas.}}$ / $k_{\text{var.}}$	Empirical Uncertainty	Avg. Time Cost[a] ↓	RMSE (m) 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[b])	26.075	0.4664	0.4497	0.3494	0.5725	0.4156	1.7731
4-2	CUAHN	rand.	3	5.0	Ensem. (2[b])	–	0.3493	0.2846	0.3206	0.5865	0.3937	1.7880
4-3	UAHN	rand.	3	0.5	Ensem. (3[b])	31.519	0.4264	0.3863	0.3335	0.5749	0.4196	1.7720
4-4	CUAHN	rand.	3	5.0	Ensem. (3[b])	–	0.3475	**0.2739**	0.3200	0.5826	0.3985	1.7903
5-1	UAHN	Basic	3	65.0	Ensem. (2[b])	–	0.3575	0.3170	0.3825	0.6186	0.4047	1.8008
5-2	CUAHN	Basic	3	50.0	Ensem. (2[b])	–	0.3445	0.3179	0.3477	0.6059	0.4035	1.7852
5-3	UAHN	Basic	3	50.0	Ensem. (3[b])	–	0.3662	0.3126	0.4199	0.6148	0.4033	1.8052
5-4	CUAHN	Basic	3	30.0	Ensem. (3[b])	–	0.3544	0.3072	0.3353	0.5992	0.4042	1.7811
6-1	UAHN	Basic[c]	4	10.0	Drop. 5% (16[b])	23.605	0.3577	0.3607	0.3623	0.5976	**0.3871**	1.7903
6-2	CUAHN	Basic[c]	4	10.0	Drop. 5% (16[b])	–	0.3906	0.3437	0.3356	0.6090	0.3945	1.7816
6-3	UAHN	Basic[c]	3	10.0	Drop. 5% (16[b])	22.356	**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[c]	3	10.0	Drop. 5% (16[b])	-	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[c]	2	5.0	Drop. 5% (16[b])	19.419	0.3533	0.3015	0.3359	0.5935	0.3964	1.7881
6-6	CUAHN	Basic[c]	2	5.0	Drop. 5% (16[b])	–	0.3556	0.3044	0.3214	0.5842	0.4057	1.7759
6-7	UAHN	Basic[c]	1	5.0	Drop. 5% (16[b])	**17.455**	0.5862	0.3612	0.3887	0.6050	crash	6.1602
6-8	CUAHN	Basic[c]	1	5.0	Drop. 5% (16[b])	–	0.4185	0.3692	0.3432	0.6005	0.4430	8.8627

[a] 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.
[b] The number of independent network models in an ensemble or the number of forward passes with dropout.
[c] Only to initialize the convolutional layers. The FC layers are randomly initialized and have dropout layers before them.

Fourth, we assess the effects of exploiting *a priori* homography for image pre-warping as shown in Fig. 3. In Table 7, 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 (~1 ms). 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. 11. 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.

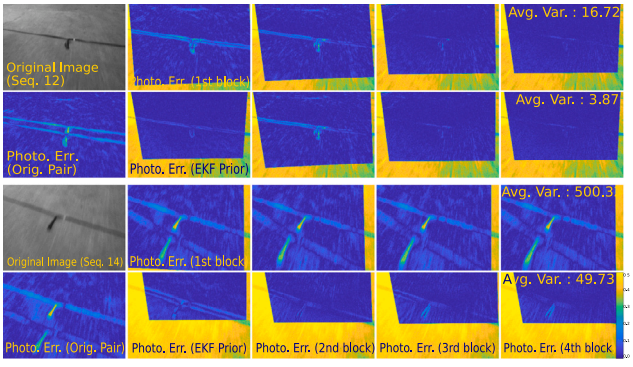


Fig. 11. The top two rows show an example image pair captured at a relatively slow speed (Seq. 12 of UZH-FPV). The bottom two rows show a high-speed example (Seq. 14). The two image pairs film the same scene. The 1st column shows the original images and the original photometric error maps of the image pairs. The 2nd column shows the photometric error maps of $(\widetilde{\mathcal{I}}_{t,1}, \mathcal{I}_{t-1})$ or $(\widetilde{\mathcal{I}}_{t,\text{prior}}, \mathcal{I}_{t-1})$. The 3rd to 5th columns show the photometric error maps of $(\widetilde{\mathcal{I}}_{t,i}, \mathcal{I}_{t-1})$, i is the index of network block and ranges from 2 to 4. The performance of the network 6–2 in Table 7 is shown in the 1st and 3rd rows, and network 6–4 in the 2nd and 4th rows.

Table 8
Time consumption indicators measured on a TX2 processor by running 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	320 × 224	2	×	24.00	1.48	0.12	25.61	3.32	**0.44**	26.11	0.3544
UAHN	320 × 224	3	×	27.30	**1.46**	**0.12**	28.89	**2.66**	0.78	26.11	**0.3380**

7.3. Onboard deployment for feedback control

UAHN-VIO, 6–5 of Table 7, 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 Section 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 90-degree downward-facing. The image stream and IMU measurement stream are published at 30 Hz and 200 Hz 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[1] 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,[1] tracking a circle trajectory,[1] 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. 14. The link to the flight video is in Appendix A.

7.4. Time efficiency and processing latency

We have shown the network inference time consumption in Table 7. In the following, we further discuss the detailed time consumption and processing latency of the whole VIO system. The 1st row of Table 8 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 30 Hz video (33.3 ms). The 3rd column from the right shows the percentage of frames that take more than 33.3 ms 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.

The MSCKF [3] in Table 8 is implemented by [6]. We choose it instead of other SOTA approaches because our C++ implementation is based on the open-sourced code of [6]. 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 8 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 8 has the same settings as the one in Table 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 33.3 ms, 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 33.3 ms 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 ∼3.3 ms 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 8 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 Tables 4 and 8, 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.

It is clear that for both approaches in Table 8, 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

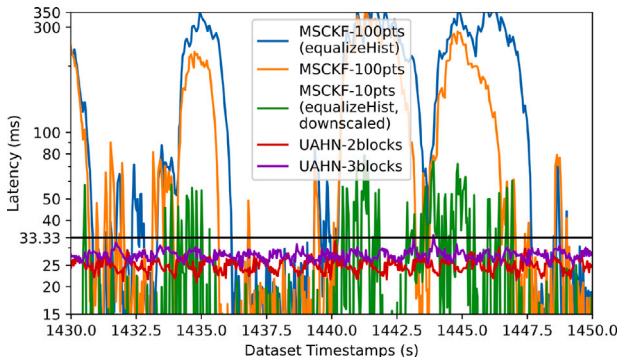


Fig. 12. Processing latency within 20 s of Seq. 2, UZH-FPV dataset. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

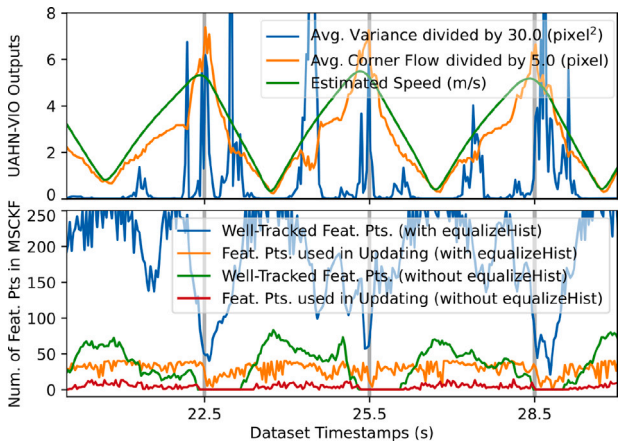


Fig. 13. The outputs of UAHN-VIO (6–5 of Table 7) and MSCKF (2nd row of Table 8) processing the two-waypoint shuttle flight sequence. The average of the network-estimated variances of the eight elements of the corner flow and the average of the absolute values of the eight elements of the corner flow are downscaled for better illustration. The well-tracked points are the ones that fulfill the epipolar constraint calculated within a RANSAC scheme. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

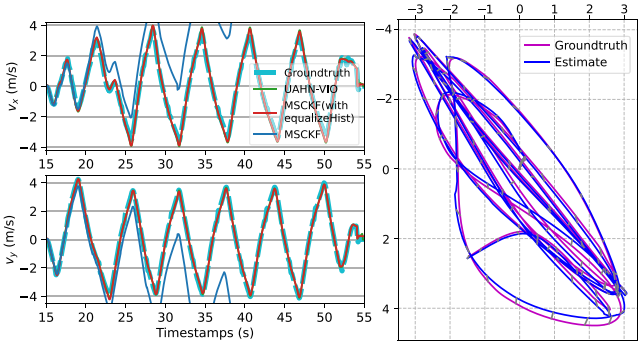


Fig. 14. UAHN-VIO (6–5 in Table 7) and MSCKF (100 feature points) with and without histogram equalization are evaluated on the two-waypoint shuttle flight sequence, running on a TX2 processor. The left subplot shows the velocity expressed in the body frame. The right subplot is the trajectory evaluation of UAHN-VIO plotted by [58]. 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. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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. 12 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 corresponds to the big variance of time consumption shown in Table 8. 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 [59], and learning-based feature points [60–62]. VIO approaches based on feature points adopting such techniques can achieve lower and scene-independent stable latency in their vision frontends. But the complicated backends that utilize the pixel trajectories of the vanished points [3], BA [5], or iterative EKF [11] 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.

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. 11, 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 frontend 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. 14) 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.

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. 13 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. 14. Fig. 15 shows an image captured when the optical flow is close to a peak. It is too blurry for FAST feature point [63] 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 be captured by the network. As shown by the photometric error map (bottom right of Fig. 15), the network is able to retrieve a reasonable homography transformation that aligns the images well.

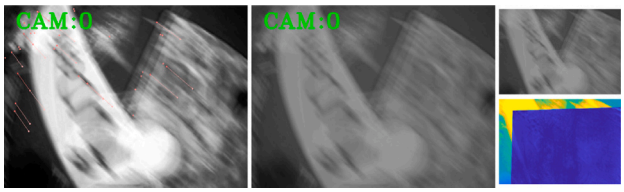


Fig. 15. An example of the highly blurry images captured during the two-waypoint shuttle flight (the *shuttle2* sequence of MYNT-CyberZoo dataset shown in Table 6). Feature point tracking results of MSCKF with and without histogram equalization are shown on the left and middle respectively. These two images are from the visualization module of [6]. 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 7. 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 fed into the VIO approaches. The network uses the smaller resolution. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

A clear phenomenon shown by the top subplot of Fig. 13 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. 11 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 *Supplementary Document* 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 backends compensate for the effect of fewer points to some extent. We did not use datasets with long periods of ongoing motion blur in this article, 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 accuracy of mean prediction is noticeably affected. Also because of the higher accuracy of UAHN-VIO than other approaches in most tests in this article, we believe that the proposed network has advanced robustness toward motion blur.

7.6. Potential improvements

About reaching the end of this article, 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 frontend. Compared with other VIO solutions, the EKF backend 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. It is possible to apply the Keyframe mechanism that achieved big success in many VIO solutions to CUAHN-VIO following the similar scheme proposed in [10]. Loop closure and relocalization functions that are important for a SLAM system have not been developed in this work. A keyframe mechanism may contribute to the loop closure and relocalization functions, which is an interesting topic for future research. 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 paper 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 backend. An interested reader can refer to [10,53], 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 training, the four network blocks are trained to handle the whole homography 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.

8. Conclusions

In this article, we propose CUAHN-VIO. Its vision frontend is a homography transformation network with uncertainty awareness and the backend 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 article. 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.

CRediT authorship contribution statement

Yingfu Xu: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Guido C.H.E. de Croon:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary material

A video of CUAHN-VIO runtime performance is available at https://youtu.be/_NgDkgON-nE. *Supplementary Document* and the software code developed for CUAHN-VIO are stored at https://github.com/tudelft/CUAHN-VIO.

Data availability

Data will be made available on request.

References

[1] Jeffrey Delmerico, Titus Cieslewski, Henri Rebecq, Matthias Faessler, Davide 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.

[2] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, Juan D. Montiel, Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam, IEEE Trans. Robot. 37 (6) (2021) 1874–1890.

[3] Mingyang Li, Anastasios I. Mourikis, High-precision, consistent EKF-based visual-inertial odometry, Int. J. Robot. Res. 32 (6) (2013) 690–711.

[4] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, Paul Furgale, Keyframe-based visual–inertial odometry using nonlinear optimization, Int. J. Robot. Res. 34 (3) (2015) 314–334.

[5] Tong Qin, Peiliang Li, Shaojie Shen, VINS-mono: A robust and versatile monocular visual-inertial state estimator, IEEE Trans. Robot. 34 (4) (2018) 1004–1020.

[6] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, Guoquan Huang, Openvins: A research platform for visual-inertial estimation, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2020, pp. 4666–4672.

[7] Xiaochen Qiu, Hai Zhang, Wenxing Fu, Lightweight hybrid visual-inertial odometry with closed-form zero velocity update, Chin. J. Aeronaut. (2020).

[8] Jakob Engel, Thomas Schöps, Daniel Cremers, LSD-SLAM: Large-scale direct monocular SLAM, in: European Conference on Computer Vision, Springer, 2014, pp. 834–849.

[9] Jakob Engel, Vladlen Koltun, Daniel Cremers, Direct sparse odometry, IEEE Trans. Pattern Anal. Mach. Intell. 40 (3) (2017) 611–625.

[10] Shangkun Zhong, Pakpong Chirarattananon, An efficient iterated EKF-based direct visual-inertial odometry for MAVs using a single plane primitive, IEEE Robot. Autom. Lett. 6 (2) (2020) 486–493.

[11] Michael Bloesch, Michael Burri, Sammy Omari, Marco Hutter, Roland Siegwart, Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback, Int. J. Robot. Res. 36 (10) (2017) 1053–1072.

[12] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, Davide Scaramuzza, SVO: Semidirect visual odometry for monocular and multicamera systems, IEEE Trans. Robot. 33 (2) (2016) 249–265.

[13] Wenshan Wang, Yaoyu Hu, Sebastian Scherer, TartanVO: A generalizable learning-based VO, in: Conference on Robot Learning, PMLR, 2021, pp. 1761–1772.

[14] Brandon Wagstaff, Emmett Wise, Jonathan Kelly, A self-supervised, differentiable Kalman filter for uncertainty-aware visual-inertial odometry, 2022, arXiv preprint arXiv:2203.07207.

[15] Yingfu Xu, Guido CHE de Croon, CNN-based ego-motion estimation for fast MAV maneuvers, 2021, arXiv preprint arXiv:2101.01841.

[16] Gabriele Costante, Michele Mancini, Paolo Valigi, Thomas A. Ciarfuglia, Exploring representation learning with cnns for frame-to-frame ego-motion estimation, IEEE Robot. Autom. Lett. 1 (1) (2015) 18–25.

[17] Sen Wang, Ronald Clark, Hongkai Wen, Niki Trigoni, End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks, Int. J. Robot. Res. 37 (4–5) (2018) 513–542.

[18] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, Niki Trigoni, Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31, (no. 1) 2017.

[19] Chethan M. Parameshwara, Gokul Hari, Cornelia Fermüller, Nitin J. Sanket, Yiannis Aloimonos, DiffPoseNet: Direct differentiable camera pose estimation, 2022, arXiv preprint arXiv:2203.11174.

[20] Liming Han, Yimin Lin, Guoguang Du, Shiguo 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.

[21] Chunshang Li, Steven 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.

[22] Tinghui Zhou, Matthew Brown, Noah Snavely, David 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.

[23] Zhichao Yin, Jianping 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.

[24] Yuhua Chen, Cordelia Schmid, Cristian Sminchisescu, Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 7063–7072.

[25] Reza Mahjourian, Martin Wicke, Anelia 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.

[26] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, Ian Reid, Unsupervised scale-consistent depth and ego-motion learning from monocular video, in: Advances in Neural Information Processing Systems, vol. 32, 2019.

[27] Ruihao Li, Sen Wang, Zhiqiang Long, Dongbing Gu, Undeepvo: Monocular visual odometry through unsupervised deep learning, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 7286–7291.

[28] Nan Yang, Lukas von Stumberg, Rui Wang, Daniel 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.

[29] Yasin Almalioglu, Mehmet Turan, Muhamad Risqi U Saputra, Pedro PB de Gusmão, Andrew Markham, Niki Trigoni, SelfVIO: Self-supervised deep monocular visual–Inertial odometry and depth estimation, Neural Netw. 150 (2022) 119–136.

[30] Andreas Geiger, Philip Lenz, Raquel 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.

[31] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W. Achtelik, Roland Siegwart, The EuRoC micro aerial vehicle datasets, Int. J. Robot. Res. 35 (10) (2016) 1157–1163.

[32] Zachary Teed, Jia Deng, Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras, in: Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 16558–16569.

[33] Alex Kendall, Yarin 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] David A. Nix, Andreas S. Weigend, Estimating the mean and variance of the target probability distribution, in: Proceedings of 1994 Ieee International Conference on Neural Networks, Vol. 1, ICNN'94, IEEE, 1994, pp. 55–60.

[35] Radford M. Neal, Bayesian Learning For Neural Networks (Ph.D. thesis), University of Toronto, 1995.

[36] Balaji Lakshminarayanan, Alexander Pritzel, Charles Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, Adv. Neural Inf. Process. Syst. 30 (2017).

[37] Wesley J. Maddox, Pavel Izmailov, Timur Garipov, Dmitry P. Vetrov, Andrew Gordon Wilson, A simple baseline for Bayesian uncertainty in deep learning, Adv. Neural Inf. Process. Syst. 32 (2019) 13153–13164.

[38] Yarin Gal, Zoubin Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, in: International Conference on Machine Learning, PMLR, 2016, pp. 1050–1059.

[39] Elia Kaufmann, Mathias Gehrig, Philipp Foehn, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, Davide 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] Maria Klodt, Andrea 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] Valentin Peretroukhin, Brandon Wagstaff, Jonathan Kelly, Deep probabilistic regression of elements of SO (3) using quaternion averaging and uncertainty injection., in: CVPR Workshops, 2019, pp. 83–86.

[42] Matteo Poggi, Filippo Aleotti, Fabio Tosi, Stefano 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] Eddy Ilg, Ozgun Cicek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, Thomas 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] Nitin J. Sanket, Chahat Deep Singh, Cornelia Fermüller, Yiannis Aloimonos, Ajna: Generalized deep uncertainty for minimal perception on parsimonious robots, Science Robotics 8 (81) (2023) eadd5139.

[45] Alex Kendall, Roberto Cipolla, Modelling uncertainty in deep learning for camera relocalization, in: 2016 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2016, pp. 4762–4769.

[46] Gabriele Costante, Michele Mancini, Uncertainty estimation for data-driven visual odometry, IEEE Trans. Robot. 36 (6) (2020) 1738–1757.

[47] Daniel DeTone, Tomasz Malisiewicz, Andrew Rabinovich, Deep image homography estimation, 2016, arXiv preprint arXiv:1606.03798.

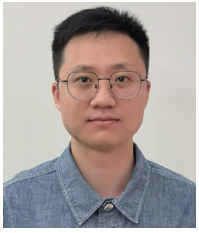
[48] Farzan Erlik Nowruzi, Robert Laganiere, Nathalie Japkowicz, Homography estimation from image pairs with hierarchical convolutional networks, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2017, pp. 913–920.

[49] Hoang Le, Feng Liu, Shu Zhang, Aseem Agarwala, Deep homography estimation for dynamic scenes, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 7652–7661.

[50] Ty Nguyen, Steven W. Chen, Shreyas S. Shivakumar, Camillo Jose Taylor, Vijay Kumar, Unsupervised deep homography: A fast and robust homography estimation model, IEEE Robot. Autom. Lett. 3 (3) (2018) 2346–2353.

[51] Jirong Zhang, Chuan Wang, Shuaicheng Liu, Lanpeng Jia, Nianjin Ye, Jue Wang, Ji Zhou, Jian Sun, Content-aware unsupervised deep homography estimation, in: European Conference on Computer Vision, Springer, 2020, pp. 653–669.

[52] Max Jaderberg, Karen Simonyan, Andrew Zisserman, Koray Kavukcuoglu, Spatial transformer networks, in: Advances in Neural Information Processing Systems, vol. 28, 2015, pp. 2017–2025.
[53] Shangkun Zhong, Pakpong Chirarattananon, Direct visual-inertial ego-motion estimation via iterated extended kalman filter, IEEE Robot. Autom. Lett. 5 (2) (2020) 1476–1483.
[54] Simon Baker, Ankur Datta, Takeo Kanade, Parameterizing Homographies, 2006.
[55] Clément Godard, Oisin Mac Aodha, Michael Firman, Gabriel J. Brostow, Digging into self-supervised monocular depth estimation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 3828–3838.
[56] Olaf Ronneberger, Philipp Fischer, Thomas 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.
[57] Joan Sola, Quaternion kinematics for the error-state Kalman filter, 2017, arXiv preprint arXiv:1711.02508.
[58] Zichao Zhang, Davide 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.
[59] S. Heymann, K. Müller, A. Smolic, B. Froehlich, T. Wiegand, SIFT implementation and optimization for general-purpose GPU, 2007.
[60] Axel Barroso Laguna, Krystian Mikolajczyk, Key. Net: Keypoint detection by handcrafted and learned CNN filters revisited, IEEE Trans. Pattern Anal. Mach. Intell. (2022).
[61] Daniel DeTone, Tomasz Malisiewicz, Andrew 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.
[62] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, Andrew 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.
[63] Edward Rosten, Tom Drummond, Machine learning for high-speed corner detection, in: European Conference on Computer Vision, Springer, 2006, pp. 430–443.



Yingfu Xu received his M.Sc. degree in Aerospace Engineering from the Harbin Institute of Technology, Harbin, China, in 2018. From 2018, he pursued a Ph.D. degree in robotics at the Faculty of Aerospace Engineering, Delft University of Technology, Delft, the Netherlands, and graduated in 2023. He conducted research on vision-based ego-motion estimation, with an emphasis on approaches based on self-supervised deep learning, applied to flying robots with limited computational power. Since 2022, Yingfu Xu has been working at the hardware-efficient artificial intelligence team, imec the Netherlands, Eindhoven, the Netherlands. His research focuses on hardware-aware neural network optimization for low-power neuromorphic processors.

Guido C. H. E. de Croon received his M.Sc. and Ph.D. in the field of Artificial Intelligence (AI) at Maastricht University, the Netherlands. His research interest lies in computationally efficient algorithms for robot autonomy, with an emphasis on computer vision and evolutionary robotics. Since 2008 he has worked on algorithms for achieving autonomous flight with small and lightweight flying robots, such as the DelFly flapping wing MAV. In 2011-2012, he was a research fellow in the Advanced Concepts Team of the European Space Agency, where he studied topics such as optical flow-based control algorithms for extraterrestrial landing scenarios. Afterwards, he returned to TU Delft, focusing on research topics such as bio-inspired and neuromorphic AI for small drones. Currently, he is a full professor at TU Delft and scientific lead of the Micro Air Vehicle Lab (MAVLab) of TU Delft.