CUAHN-VIO: Content-and-Uncertainty-Aware Homography Network for Visual-Inertial Odometry *Supplementary Material*

Yingfu Xu and Guido C. H. E. de Croon, *Member, IEEE*

I. Network Architecture and Implementation

A. Network Architecture

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

B. Model Size

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

C. Implementation and Training

We implement the networks by PyTorch [1] v1.7.1. The training is executed on a server with GTX1080ti GPU and CUDA v10.1. The AdamW [2] optimizer with $\beta = (0.9, 0.999)$ and weight decay $\lambda = 0.01$ is utilized during the 50 training epochs. The batch size is 16. The initial learning rate is 2e-4 and it is divided by 2 after 10, 20, 30, 35, 40, and 45 epochs. The weight parameters are initialized by Kaiming initialization [3] and bias parameters are initialized to zero. The network performs inference on the validation set after each training epoch. The set of parameters that achieve the smallest average validation error are saved as the best model and evaluated on the testing set. Most networks in this article follow the above training procedure, except that Master Model has a smaller initial learning rate (5e-5).

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

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

In the training of the content-aware mask based on Laplacian probability distribution, we tried to train the network to directly output $\log b_k$ (Eq. (7) of the main article) in order to avoid zero values of b_k. However, the training became unstable. Good results were achieved when b_k is the sum of the square of the network output and a small value (2e-6).

D. Comparison of Basic Homography Networks

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

TABLE I
Comparison of Basic Homography Networks

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

* Directly regress to homography matrix

We trained two models of each architecture to exclude the influence of individual circumstances. The time consumption of network inference is measured on the GPU of an Nvidia Jetson TX2 mobile processor in Max-P ARM power mode. The shown values in the Python column are the averages

of one thousand times of inference on the validation set of the Basic Dataset when the batch size is one. The C++ time consumption values are measured when processing the downward-facing Seq. 2 of the UZH-FPV dataset. From Table I we can conclude that the architecture using pyramidal images is more accurate and faster than its peer utilizing FPE. Downsampling by average pooling is slightly faster than bilinear interpolation. Their accuracy is very close. Besides corner flow, directly regressing to the eight elements of $\boldsymbol{H}$ (the last diagonal element is 1.0) is also implemented for the first three blocks (bottom row of Table I). It does not require DLT solving so it runs faster. But obviously, it is less accurate.

Model 1 of the network using pyramidal images and average pooling in Table I is selected as the Basic Model in the main article.

E. Direct Linear Transformation (DLT) Solver

DLT solver aims at solving the homography matrix $\boldsymbol{H}$ that has 8 degrees of freedom from at least four pairs of corresponding points. As (1) shows, $\boldsymbol{H}$ relates the pixel coordinates in two images of a point lying on a planar surface.

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

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

$$\begin{aligned} \boldsymbol{A}_i &= \begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -u_2 u_1 & -u_2 v_1 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -v_2 u_1 & -v_2 v_1 \end{bmatrix} \\ \boldsymbol{h} &= \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 \end{bmatrix}^T \\ \boldsymbol{b}_i &= \begin{bmatrix} u_2 & v_2 \end{bmatrix}^T \end{aligned} \tag{2}$$

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

II. Predictive Uncertainty

A. Why Learning Requires a Teacher Network?

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

The predictive uncertainty map b_k explains the uncertainty of photometric matching that is affected by both image content and the accuracy of homography transformation. For a well-trained content-aware network, b_k mainly reflects the content information in complicated scenes as shown in Fig. 6 (main article). Because the photometric error induced by the prediction error is much smaller compared to the non-homography content. Since it is unknown whether a pixel belongs to the single non-reflective plane, it is not tractable to decouple the prediction error and the image content. Even if b_k is determined by the prediction error alone, *e.g.*, the pixels on the plane are accurately semantically segmented or the input images have no unfavorable content, it is not clear to us how to calculate the prediction uncertainty from b_k.

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

B. Comparison of Different Output Dimensions

In addition to predicting eight different values for each $\log \sigma_n^2$ (8-d variance), inspired by the concise approach utilized in [6], we also implement predicting a generalized single value that represents the overall uncertainty (1-d variance) and duplicating it eight times to fill in all dimensions. It assumes that noise in the input affects all dimensions of $\boldsymbol{f}_4$ equally, neglecting the difference among them. Two types of variance representation under different supervisions are compared in Table II. The 4th block of the student model is randomly initialized. The best model in one training attempt is the one achieving the smallest average imitation error on the validation set. In terms of the prediction accuracy of the mean values, 1-d variance is slightly better. While 8-d variance has obvious advantages in predictive uncertainty as evidenced by lower AUSE and higher inside rate.

TABLE II
Comparison of Predictive Uncertainty

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

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

C. Difficult Testing Samples

Fig. 1 shows several difficult testing samples from which we explore what leads to high prediction error and high predictive uncertainty. Comparing the Basic Model and the Master Model, the Master Model achieves accurate predictions except for the 1st and 5th images. Significant blur (1st and 2nd columns) and lack of texture (4th and 5th columns) cause big prediction errors of the Basic Model that has less model capacity. The master model achieves high accuracy on the images of the 2nd and 4th columns, while the errors of the Master-Teach student model are big. It indicates that the student fails to imitate the teacher well. From the appearance of the images, we speculate that the difficulty in imitation is due to the unfavorable image contents from which extracting desired information is hard for a network. That is to say, the big

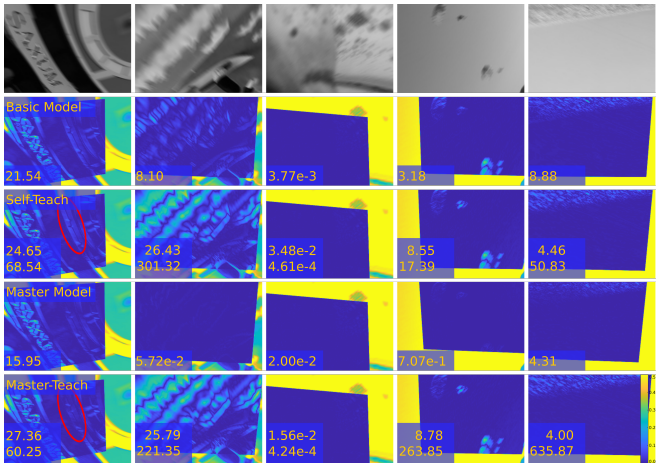

Fig. 1. Five testing samples from the Basic Dataset that contain unfavorable content: significant blur (1st to 4th) and lack of texture (4th and 5th). 1st row shows one of the input images $\mathcal{I}_p$. The 2nd to 5th rows show the photometric error map of ($\mathcal{I}_p$, $\mathcal{I}_c$), where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

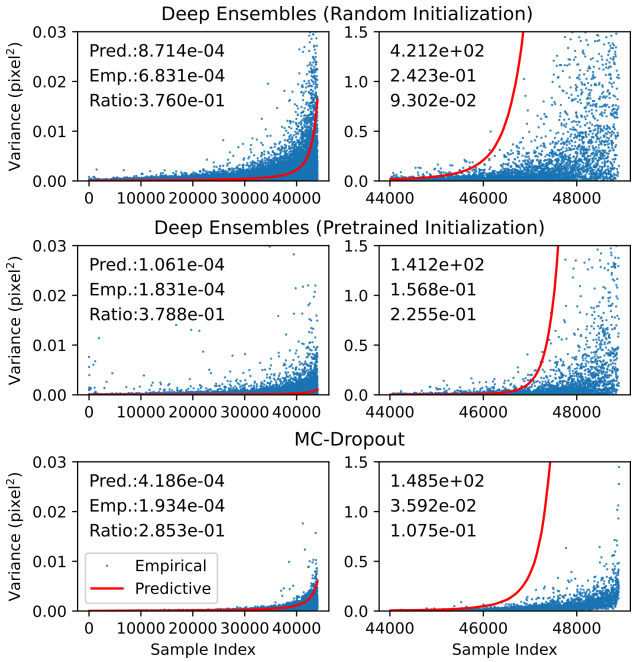

Fig. 2. Predictive variance $\sigma^2_{\text{pred.}}$ and empirical variance $\sigma^2_{\text{emp.}}$ of three network models in an ensemble. The $\sigma^2_{\text{pred.}}$- $\sigma^2_{\text{emp.}}$ pairs of each model are sorted according to $\sigma^2_{\text{pred.}}$ and separated into two groups. 90%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

predictive variances are caused by unfavorable image contents. It corroborates the definition of predictive uncertainty, which is that it captures the uncertainty in network output caused by the content-determined observation noise. Based on our observations, the noise is mainly caused by textureless image content and blur.

III. Correlation between Predictive and Empirical Uncertainty

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

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

In order to gain more insight, we log the $\sigma^2_{\text{pred.}}$ and $\sigma^2_{\text{emp.}}$ on UZH-FPV and show the average values in Table III. The light yellow group shows the results of VIO 6-2 in Table V of the main article. The light green group shows the results of a variant of 4-4 with four blocks and without *a priori*. The three sequences on the left were recorded when there are crowded low objects on the ground. The other three sequences were filmed in the same flight arena but the ground is relatively cleaner from objects. We respectively sort the sequences of the two environments in ascending order of speed. Because the network input image can vary with *a priori* homography

TABLE III
Uncertainty Estimation on UZH-FPV.

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

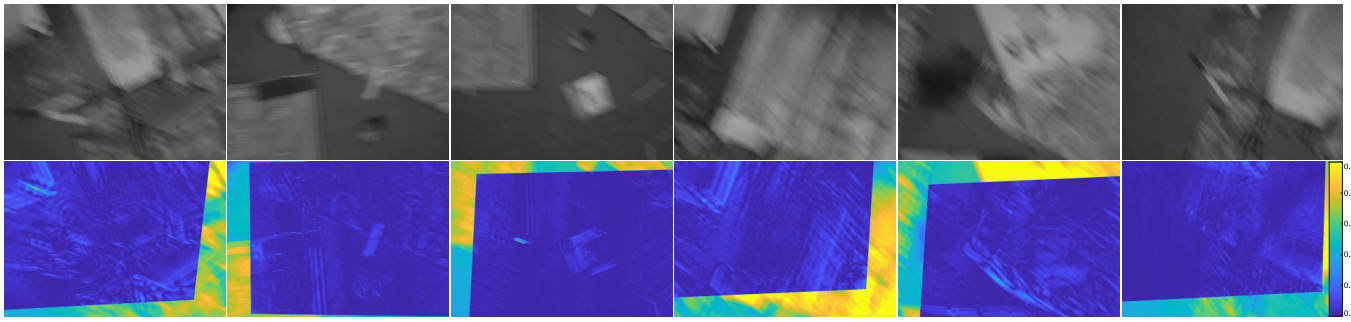

Fig. 3. Example images from the high-speed two-waypoint shuttle flight sequence. The top row shows the original images and the bottom row shows the photometric error maps of the image pairs warped according to the network predictions of homography transformation. Significant motion blur causes big predictive uncertainty. The average values of the variance estimations of the eight elements of the corner flow are respectively 394.99, 106.41, 59.80, 183.90, 190.22, 221.72 (pixel^2) for the six examples.

transformation, we run all four network blocks and the inputs are the original images. Thus different networks have identical inputs. The data in Table III tells that both predictive and empirical variances increase with speed. The growth in $\sigma^2_{\text{pred.}}$ is more significant and thus the ratio of $\sigma^2_{\text{emp.}}$ decreases with speed. It is related to the motion blur considered as observation noise by the network, as discussed before. The positive correlation between $\sigma^2_{\text{pred.}}$ and $\sigma^2_{\text{emp.}}$ is observable. For both $\sigma^2_{\text{pred.}}$ and $\sigma^2_{\text{emp.}}$, deep ensembles produce bigger values, implying more comprehensive uncertainty estimation that contributes to better AUSE and Inside Rate, as shown in Fig. 9 in the main article.

Snapshot Ensemble [7] learns multiple models faster than deep ensembles. Since it shares the disadvantage of high inference time cost with deep ensembles (limited by our current implementation), it is not separately discussed. The approach proposed in [8] that enhances model diversity in the ensemble by altering hyperparameters is not discussed for the same reason.

IV. Network Uncertainty and Velocity

As supplementary to Fig. 12 in the main article, Fig. 4 illustrates the statistically positive correlation between the magnitude of optical flow and network uncertainty estimation. We do not know the exact magnitude of optical flow but it is determined by the speed divided by the distance to the ground and the rotation rate. Since the rotation in the dataset is mostly slow, the height-scaled speed can serve as an approximate equivalent of the magnitude of optical flow. For both sequences, estimated variance is more likely to have a big value when the speed is high. In the slow sequence, most estimated variances have very small values thanks to the clear and sharp images. While in the fast sequence where blurry images are common, the positive correlation between the estimated variance and height-scaled speed is obvious. Fig. 3 shows cases where the network has big variance estimations.

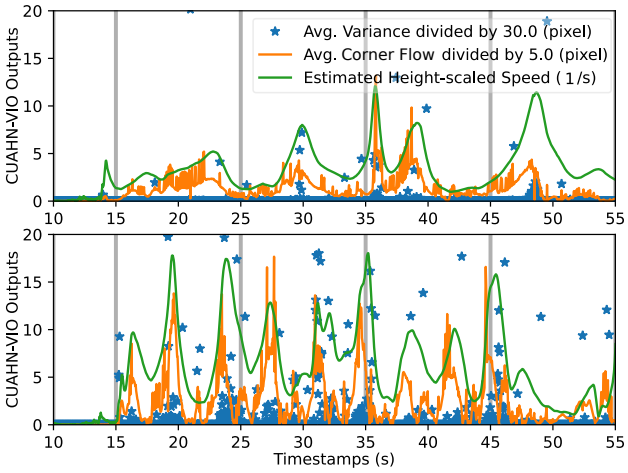

Fig. 4. The outputs of CUAHN-VIO (6-4 of Table V in the main article) evaluated on a slow sequence (Seq. 12) and a fast sequence (Seq. 14) of UZH-FPV dataset. Both sequnces were collected in the same environment.

V. UAHN-VIO for Feed-Back Control

Two of the common practices for VIO to obtain better performance in autonomous flights are 1) publishing the estimated states in higher frequency, *e.g.*, IMU measurement frequency, and 2) propagating the states to the current time using the IMU measurements that come after the latest image to compensate for the image processing latency. Since we aim to evaluate the VIO instead of pursuing the overall performance of autonomous flight, the above methods are not deployed in this article. Nevertheless, thanks to the short and almost constant processing time of UAHN-VIO, the MAV performed stable controlled flights. The RMSE of the ATE

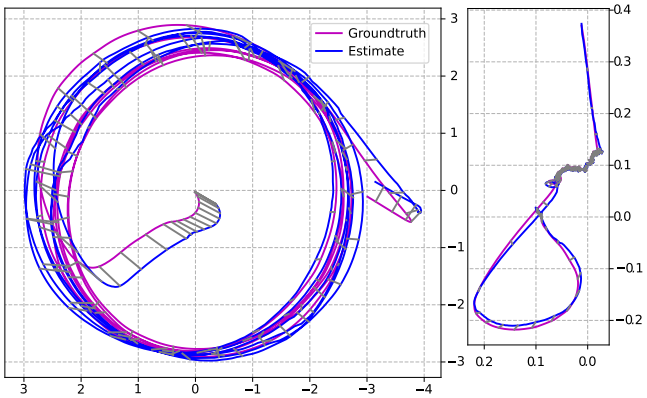

Fig. 5. Trajectories of circle trajectory tracking and fixed-point hovering.

of the estimated trajectory by UAHN-VIO is 0.1521 in the two-waypoint shuttle flight (Fig. 13 of the main article). The trajectories of autonomous flights of tracking a circle trajectory and fixed-point hovering are shown in Fig. 5. The RMSEs of the ATEs are respectively 0.2775 and 0.0155.

VI. VIO Backend

A. State Propagation

In our implementation, we use the simple zeroth-order integration (3). When propagating the states from t to $t+1$, we use the average of the IMU measurements sampled at the two time points (4).

$$\begin{aligned}
\boldsymbol{p}_{t+1} &= \boldsymbol{p}_t + \Delta t(-[\overline{\boldsymbol{\omega}}]_\times \boldsymbol{p}_t + \boldsymbol{v}_t),\\
\boldsymbol{v}_{t+1} &= \boldsymbol{v}_t + \Delta t(-[\overline{\boldsymbol{\omega}}]_\times \boldsymbol{v}_t + \overline{\boldsymbol{a}} + \boldsymbol{R}^{-1}(\boldsymbol{q}_t)\boldsymbol{g}),\\
\boldsymbol{q}_{t+1} &= \boldsymbol{q}_t \otimes \boldsymbol{q}\left\{\overline{\boldsymbol{\omega}}\Delta t\right\} = \boldsymbol{q}_t \otimes \begin{bmatrix} \cos(\|\overline{\boldsymbol{\omega}}\|\,\Delta t/2) \\ \dfrac{\overline{\boldsymbol{\omega}}}{\|\overline{\boldsymbol{\omega}}\|}\sin(\|\overline{\boldsymbol{\omega}}\|\,\Delta t/2) \end{bmatrix}, \quad (3)\\
\boldsymbol{b}_{a,t+1} &= \boldsymbol{b}_{a,t}, \quad \boldsymbol{b}_{g,t+1} = \boldsymbol{b}_{g,t},\\
\boldsymbol{f}_{j,t+1} &= \boldsymbol{f}_{j,t} - \Delta t(\boldsymbol{I} - (\boldsymbol{c}_j + \boldsymbol{f}_{j,t})\boldsymbol{e}_z^T)\boldsymbol{H}(\boldsymbol{c}_j + \boldsymbol{f}_{j,t})
\end{aligned}$$

$$\overline{\boldsymbol{a}} = 0.5 \cdot (\hat{\boldsymbol{a}}_t + \hat{\boldsymbol{a}}_{t+1}), \quad \overline{\boldsymbol{\omega}} = 0.5 \cdot (\hat{\boldsymbol{\omega}}_t + \hat{\boldsymbol{\omega}}_{t+1}) \quad (4)$$

B. a Priori Homography

As shown in Fig. 10 of the main article, the variance estimations for both image pairs are smaller when utilizing *a priori* homography. In the 2nd row, a noticeable phenomenon is that the photometric error map of $(\widetilde{\mathcal{I}}_{t,2}, \mathcal{I}_{t-1})$ is bigger than $(\widetilde{\mathcal{I}}_{t,\text{prior}}, \mathcal{I}_{t-1})$. The reason can be that, in training, the networks are trained to infer the whole homography transformation from the original image pairs. The 2nd block often handles bigger disparities in training than the ones between $\widetilde{\mathcal{I}}_{t,\text{prior}}$ and $\mathcal{I}_{t-1}$ and thus tends to output bigger values.

C. Iterative EKF

Inspired by [9], [10], we tried the iterative EKF scheme. After the first EKF update, the corner flow is updated. The image pair is then warped according to the updated corner flow and input to the network again to estimate the remaining visual disparities. And then EKF is updated once more according to the outputs of the second network inference. However, this scheme fails to improve accuracy.

VII. Parameter Tuning of SOTA VIO Approaches

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

At the beginning of each sequence of UZH-FPV, the MAV was swung by the human operator and then put on the ground. For sequences No. 2, 9, and 12, when we started ROVIO at the very beginning of the sequences, the estimated trajectories showed huge drifts. By changing the starting time of ROVIO to after the MAV is swung and before taking off, we obtained much better results on these sequences. We tried the IMU noise parameters given by UZH-FPV and the parameters in the ROVIO's configuration file for EuRoC. After several attempts, we found that the combination of IMU noise densities for EuRoC and the IMU random walk parameters given by UZH-FPV yielded good results.

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

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

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

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

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

References

[1] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[2] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

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

[4] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8934–8943, 2018.

[5] Y. Xu and G. C. de Croon, "Cnn-based ego-motion estimation for fast mav maneuvers," *arXiv preprint arXiv:2101.01841*, 2021.

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

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

[8] Z. Y. Ding, J. Y. Loo, V. M. Baskaran, S. G. Nurzaman, and C. P. Tan, "Predictive uncertainty estimation using deep learning for soft robot multimodal sensing," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 951–957, 2021.

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

[10] S. Zhong and P. Chirarattananon, "Direct visual-inertial ego-motion estimation via iterated extended kalman filter," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1476–1483, 2020.