CNN-based Ego-Motion Estimation for Fast MAV Maneuvers

Yingfu Xu and Guido C. H. E. de Croon

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

I. INTRODUCTION

Indoor flight of Micro Air Vehicles (MAVs) is an attractive but challenging task. Towards the goal of autonomy, robust state estimation is one of the most essential modules of the MAV’s flight control system. A camera captures rich information in a big field of view. Since being small and power-efficient, it is an ideal onboard sensor [1]. Its combination with the high-sample-frequency IMU is not only suitable for environment perception but also for real-time ego-motion estimation. Visual [2], [3] and visual-inertial [4]–[8] odometry (VO/VIO) systems contribute to MAVs’ autonomy in generic environments by achieving real-time efficiency on onboard processors with decent accuracy.

Being constrained by the limited battery life, increasing flight speed is a direct way to enlarge an MAV’s operation range and efficiency. However, it also introduces challenges for perception, and notably VO/VIO. Detection and tracking of handcrafted interest-point-based features [9]–[11] is the standard in state-of-the-art VIO systems [6]–[8]. However, such systems lack robustness in the presence of motion blur occurring during fast maneuvers. Robust Visual Inertial Odometry (ROVIO) [5] directly uses photometric errors of multilevel image patches around FAST feature points [10] to be more robust against image blur than point features, partly because the texture of the tracked image patch is taken into account. However, Foehn et al. point out that, at larger speeds the state estimation of ROVIO suffers from drift [12]. When the speed gets larger, feature points and patches can move out of the camera’s field of view sooner. We believe that the bigger visual disparities between images and the consequent lower number of frames in which features can be tracked is another adverse condition besides motion blur. Since ROVIO takes features’ 3-dimensional (3-d) positions as states of an extended Kalman filter (EKF), having fewer visual observations decreases the accuracy. Other VIO systems such as [4], [6] that estimate feature positions by multiple observations can also suffer from high-speed motion [13].

CNNs are state of the art in many computer vision tasks and are promising for VO as well. Various networks have been proposed to estimate the pose change (rotation and translation) between two or more subsequent views. There are not only supervised pose networks trained by limited ground truth [14]–[16] but also self-supervised ones trained together with other networks including a depth estimation network [17]–[21]. Evaluated by the KITTI dataset [22], these networks obtain highly accurate performances and rival VO [3] with a traditional vision method [11].

There are also pose networks considering the application to an MAV’s ego-motion estimation. For example, in [16], a recurrent CNN is trained on the EuRoC MAV dataset [23] to regress the 6 degree-of-freedom (6DoF) motion. The network manages to learn the more complex (compared with a car) MAV’s dynamics but the accuracy is limited by the small amount of training data. Differently, PRGFlow [24]...
focuses on the essential function of estimating the 3-d translational velocity of the MAV with a downward-facing camera, assuming a planar ground. Aided by the attitude estimated by IMU measurements, via image warping, the task is simplified to the pixel-level similarity transformation estimation. Although PRGFlow thoroughly studied CNN-based ego-motion estimation, the focus is on the low-speed flight (about 0.5m/s on average), with motion blur lacking from the artificially generated training images.

Hence, it is currently still an open question of how good CNNs perform during fast maneuvers. To gain insight into this matter, in this article we study networks predicting 3-d relative translation of MAVs in fast maneuvers with a downward-facing camera. The networks are trained and tested on images with significant motion blur and big visual disparities. Our main contributions are that we: (1) Extend and further improve the performance of the network framework proposed in [24] to fit fast maneuvers, and (2) Investigate how well the networks can deal with faster motion in comparison with traditional feature-point-based methods. According to our knowledge, this is the first work showing networks’ superior performance in fast motion when traditional feature-point-based methods have high failure rates.

II. METHODOLOGY

A. Homography Transformation

As shown in Eq. 1, for a fixed point laying on a plane observed by two cameras, it has been proven in [25] that the projective coordinates \( x_1, x_2 \) of the same point in the camera frames are related by the homography matrix \( H \) that depends only on the 6-d relative pose of the cameras and the unit normal vector of the plane \( n \). \( R \) denotes the rotation matrix between the camera frames and \( t \) denotes the translation vector expressed in the second camera’s frame pointing from the second camera to the first one. The scalar \( d \) is the distance from the first camera to the plane.

\[
x_2 = Hx_1, \quad H = R + \frac{tn^T}{d}
\]

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

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

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

B. Cascaded Network Blocks Connected by Image Warping

Sanket et al. adopt the inverse compositional spatial transformer networks (ICSTN) [26] as the framework of their networks [24]. The ICSTN has multiple network blocks that predict the image deformation that benefits the final goal. Based on the homography transformation, a new image can be synthesized by warping the original image using the method proposed in [27]. As shown in Fig. 2, a network block is made up of multiple convolutional layers followed by a fully-connected layer regressing the translation. With multiple cascaded network blocks, each block takes the concatenated original image 1 and the image 2 warped by the newest pose prediction as input and combine its output into the pose prediction. As the pose prediction is refined by more blocks, there is less relative motion between the concatenated images. Each block predicts a part of the total relative translation, making the problem more tractable. The network can also make use of an initial guess of the relative pose, which can be available from the MAV’s dynamic model. PRGFlow has compared network architectures inside one block. Focusing on fast maneuvers, we study higher-level architectures applying to the pyramidal images and feature maps to enlarge the receptive field which is important for dealing with the big visual disparities.

The loss functions of the networks are the mean of the Charbonnier [28] loss of the predicted 3-d translation’s error in supervised learning and the mean of the Charbonnier loss

![Fig. 2. An ICSTN-based network with 3 blocks for relative pose prediction.](image)

The dashed line frame indicates the basic functional unit that can be sequentially stacked one or multiple times. The dotted frame indicates a network block that takes (downsampled) concatenated images as input and predicts the 3-d distance-scaled relative translation.
of the valid pixels’ photometric error in self-supervised learning. For data augmentation, we feed the network with image pairs concatenated in both orders to perform bidirectional training.

We implement the networks in Python 3.6.9 with the PyTorch [29] 1.1.0 library. The Adam optimizer [30] with $\beta = (0.9, 0.999)$ is utilized during the 25 training epochs. The batch size is 16. The initial learning rate is 0.0002 and it is divided by 2 after 5, 10, 15, and 20 epochs. The weights of convolutional layers are initialized by Glorot initialization [31] with a gain of 1. The weights of fully-connected layers are initialized by the (Pytorch default) uniform distribution $U(-\sqrt{k}, \sqrt{k})$ where $k$ is the multiplicative inverse of the number of input features.

### C. Dataset Generation

We use the Microsoft COCO dataset [32] as the source of a large variety of textures to generate a big number of image pairs thanks to the homography transformation. A source image is treated as a plane above which a simulated camera is moving. For one plane, one image pair is generated. Costante et al. and Kendall et al. test their pose estimation networks with respectively artificial Gaussian blur [14] and motion blur [33] added to images. Their blur is uniform over the whole image and thus not ego-motion-related. In order to obtain realistic blur that is caused by the camera’s motion within the exposure duration, we simulate a moving camera whose time step for the kinetic integration is 0.1 millisecond (ms) and the exposure duration is 10ms. In each integration step during the exposure, an image is sampled from the homography transformation of the plane. The blurry image is the average of the 100 sampled images. The poses of both the exposure starting step and ending step of an image are recorded. Except for subsection III-C, the pose of the starting step is used as the ground truth in supervised learning. 30 frames per second (fps) are recorded to simulate a common global shutter camera. The images are in grayscale with the resolution of $320 \times 224$ pixels. The intrinsics of the simulated camera are $f_x = 160$, $f_y = 160$, $c_x = 160$, $c_y = 112$.

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

### III. NETWORKS

#### A. ICSTN-based Networks

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

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

Networks with a single block are shown in the first 3 rows of Table I. The 1st network is the pose estimation network proposed in [17]. The 2nd and 3rd networks respectively have skip connections [34], [35] for better performance in their deeper architectures. They have smaller model sizes, higher accuracy, but slower speed. The 2nd network with 18 convolutional layers and densely connected architecture has the highest accuracy. In the case of multiple blocks, for each block, there is an image warping operation that has unneglectable time consuming. Since a network’s inference speed is required to be around 100fps, the total number of layers in the whole network decreases when the number of
blocks increases. The accuracy gets worse when there are more than 3 blocks mainly because the blocks are too shallow and the total capacity of the whole network decreases. As shown by the 6th and 8th networks, bigger strides lead to fewer layers achieves higher inference speed and accuracy, and thus fewer parameters in the fully-connected layer. Besides, it increases the receptive field. Since our dataset has image pairs with big visual disparities, a bigger receptive field can capture more feature correspondences and improve the accuracy.

For the networks with multiple blocks, instead of only using the loss of the final prediction (end-point loss) in training [24], we weighted sum the losses of every prediction after the loss weight distributions of blocks are respectively [0.3, 0.7], [0.2, 0.3, 0.5], and [0.1, 0.2, 0.3, 0.4] for networks with 2, 3, and 4 blocks. The accuracy is compared by the 4th, 5th, and 7th networks of Table I. Multi-stage losses produce higher accuracy. For the 6th and 8th networks, we only show the results of multi-stage losses. All the networks in the rest part of this article are trained with their multi-stage losses.

B. Pyramidal Images and Feature Maps in ICSTN

From Table I, we find that a bigger receptive field can benefit accuracy. When the kernel size and stride keep the same, another way to increase the receptive field is using pyramidal images. Networks using pyramidal images or feature maps with lower resolution are shown in Table II. The downsampled image at each pyramid level has half the size of the image of its adjacent lower level. So the lowest-resolution image of the network that has 4 pyramids has one-eighth the width and height of the original image. The number of network blocks is the same as the number of pyramids. The first pose prediction block uses the images at the highest pyramid level with the lowest resolution. The predicted pose is used to warp the original image. Then the warped image is downsampled to the next lower pyramid level and input to the next network block. For image downsampling, we compared bilinear image interpolation [27] and average pooling. They have similar accuracy, but average pooling is faster in our Pytorch implementation.

Since the EPE’s standard deviation is enough to reflect the accuracy of the network predictions, the medians of EPE’s absolute values are not shown in Table II. Comparing the 1st network of Table II with the 5th of Table I, with the same kernel size and stride, the pyramidal network that has fewer layers achieves higher inference speed and accuracy, thanks to the bigger receptive fields of the first two blocks. Comparing the 2nd network of Table II with the 6th network of Table I, the pyramidal network has slightly lower accuracy. We think it is because when the receptive fields are big

| Network | Num. Blocks | Num. Conv./Kernel/Stride | Num. Params | FPS (intrpl. / avg. pooling) | Inlier Rate(%) | EPE’s Standard Deviations (1e-3) | Receptive Field | EPE’s Standard Deviations (1e-3) |
|---------|-------------|--------------------------|------------|-----------------------------|---------------|---------------------------------|----------------|---------------------------------|
| 1 [17]  | 1           | 8/ 7, 5/ 2.2             | 1.583M     | 215                         | 263           | 90.14                           | (13.44, 13.88, 19.23) | (7.83, 8.11, 11.93) |
| 2 [34]  | 1           | 18/ 3.3/ 2.2             | 1.441M     | 105                         | 759           | 91.77                           | (6.70, 6.94, 9.63)   | (3.85, 4.03, 6.12)   |
| 3 [35]  | 1           | 2/ 3.3/ 2.2              | 1.477M     | 103                         | 975           | 92.35                           | (7.32, 7.58, 10.57)  | (4.33, 4.48, 6.66)   |
| 4 (ours)| 2           | 9/ 7, 5/ 2.2             | 1.385M     | 104                         | 647           | 89.92                           | (3.35, 3.30, 3.91) / (2.71, 2.64, 3.28) | (1.89, 1.89, 2.61) / (1.49, 1.49, 2.18) |
| 5 (ours)| 3           | 5/ 7, 5/ 2.2             | 1.367M     | 101                         | 71            | 87.59                           | (3.33, 3.15, 4.29) / (2.28, 2.22, 3.08) | (1.91, 1.83, 2.90) / (1.24, 1.22, 2.11) |
| 6 (ours)| 3           | 5/ 7, 5/ 2.2             | 1.252M     | 101                         | 135           | 92.18                           | (2.06, 2.05, 2.90)   | (1.16, 1.13, 2.00)   |
| 7 (ours)| 4           | 3/ 7, 5/ 4.4             | 1.421M     | 95                          | 55            | 85.48                           | (3.58, 3.51, 4.75) / (2.20, 1.93, 3.06) | (2.20, 2.20, 3.20) / (1.28, 1.26, 2.13) |
| 8 (ours)| 4           | 3/ 9, 5/ 4.4             | 1.276M     | 95                          | 85            | 87.35                           | (2.10, 2.09, 3.01)   | (1.25, 1.22, 2.09)   |

Table II

ICSTN-based Networks Using Pyramidal Images or Feature Maps

| Num. Pyramid | Num. Layers/Kernel/Stride | Num. Params | FPS (intrpl. / avg. pooling) | Receptive Field | EPE’s Standard Deviations (1e-3) |
|--------------|---------------------------|------------|-----------------------------|----------------|---------------------------------|
| 3            | 3/ 7, 5/ 2.2; 4/ 7, 5/ 2.2; 5/ 7, 5/ 2.2 | 1.388M | 103 / 108 | 23×4; 39×2; 71 | (2.13, 2.05, 2.93) / (2.11, 2.09, 2.91) |
| 3            | 3/ 7, 5/ 4.2; 4/ 7, 5/ 4.2; 5/ 7, 5/ 4.2 | 1.272M | 104 / 111 | 39×4; 71×2; 135 | (2.11, 0.92, 2.94) / (2.09, 0.96, 2.94) |
| 3            | 4/ 7, 5/ 2.2; 4/ 7, 5/ 4.2; 4/ 7, 5/ 4.4 | 1.316M | 104 / 109 | 39×4; 71×2; 119 | (2.07, 2.06, 2.93) / (2.07, 2.07, 2.90) |
| 4            | 2/ 7, 5/ 2.2; 2/ 7, 5/ 4.2; 3/ 7, 5/ 4.2; 3/ 7, 5/ 4.4 | 1.386M | 93 / 98 | 15×8; 23×4; 39×2; 55 | (2.03, 2.02, 2.87) / (2.02, 2.02, 2.85) |
| 3            | [FPE: 3/ 7, 5/ 2.2] + [2; 3; 4] | 1.455M | 100 | 71; 71; 71 | (2.18, 2.12, 3.12) |
| 3            | [FPE: 3/ 7, 5/ 4.2] + [2; 3; 4] | 1.340M | 100 | 135; 135; 135 | (2.00, 1.97, 3.08) |

Table III

Comparison between Supervised and Self-Supervised Learning
enough, the pyramidal version receives less information due to the downsampling. The 8th network of Table I has a big receptive field. Also with 4 blocks, the 4th network of Table II has decreasing receptive fields with the increasing of image resolution. Although 3 out of 4 blocks have smaller receptive fields than the 8th network of Table I, this 4-stage coarse-to-fine refinement gets better accuracy. The 2nd and 3rd networks of Table II have the same total number of layers. The 3rd one having a deeper block at the lowest resolution achieves slightly higher accuracy.

The pyramidal feature maps network is based on the feature pyramid extractor (FPE) inspired by the PWC-Net [36]. The results are shown in the last 2 rows of Table II. The general principle is extracting multiple feature maps at different resolutions (pyramid levels) of each image respectively by the same convolutional feature extractor network. One of the feature maps is warped and then concatenated along the channel dimension with the other feature map of the same size. The concatenated feature maps are the input of the pose prediction blocks. The networks we design have 3 levels of pyramidal feature maps and 3 pose prediction blocks that have 2, 3, and 4 convolutional layers respectively. The FPE network at the last row of Table III has the highest accuracy in the $x$-axis and $y$-axis.

C. Self-Supervised Learning

Self-supervised learning is based on the photometric error between the image warped by the predicted relative pose and the other image. We use a mask to not count the photometric errors of the pixels whose locations to interpolate lie outside the image frame. By the results shown in the 1st row of Table III, we notice that self-supervised learning with a basic photometric loss gets better accuracy than supervised learning. The reason behind it worth further studying. For now, we think it is mainly because the target relative poses used in supervised learning are calculated from the poses at the starting time points of the image exposure. While the simulated camera keeps moving within the exposure duration, motion blur appears and the image gets a different appearance from the start of exposure, and thus there will be small photometric errors between the images warped by the target relative pose. This means the network is trained to regress to a target not perfectly matching the feature correspondences. This discrepancy can “confuse” the network. While in the case of self-supervised learning, the network tries to minimize the photometric error affected by the blur and is more likely to converge to the “accurate” relative pose that best matches the feature correspondence. When we evaluate the self-supervised network, we use the poses at the start of exposure as the ground truth, to which the network does not learn to converge. But the effect of it is smaller than the “confusion” induced by the discrepancy.

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

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

D. Networks for Tilt Angle Prediction

It is known that one can estimate the tilt of the camera relative to the plane in the view from the optical flow field [37]. Since tilt is a property of the flow field and hence affects both images, it cannot be estimated iteratively by our ICTSTN-based framework that warps only one image (Fig. 2). For this preliminary investigation, we employ a single deep network block to predict tilt angles from a pair of derotated images, supervised by ground truth. Shown in Table IV, the best network’s EPE’s standard deviation is around 4 degrees for both angles. Although the prediction is noisy, it may serve as an unbiased absolute information source of attitude.

IV. EVALUATION

The 4th network of Table III is chosen for evaluation and comparison to traditional feature-based methods. Note that the network is trained only with the simulation dataset described in subsection II-C without any fine-tuning to highlight the generalizability. We use MATLAB functions for traditional feature detection and matching. More feature points can be detected by tuning the parameters of the functions. Here we only show the results of the default parameters. 50 uniformly distributed ones are selected when there are enough detected features. The translation is obtained by calculating the similarity matrix (with known in-plane rotation, 3 DoF left) based on linear least squares and random sample consensus (RANSAC).
A. Simulated Dataset

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

The norms of the error vectors of the estimated distance-scaled translations and their local standard deviations are shown in Fig. 4. We use linear fitting to show their trends. The local standard deviations are calculated with the window size of 10% of the total number of inliers. For feature-based methods, if there are less than 2 inlier matchings in RANSAC, this pair is treated as an outlier. For the estimated pose, we apply the same local outlier rejection as Section III with the window size of 500. The final inlier rates of the network, SURF [38], ORB [11], and FAST [10] are shown in Fig. 4. The network has the highest inlier rates partly because it does not rely on the number of matches so it can perform prediction on every image pair. Fig. 4 shows that the network is most accurate with both datasets. Its performance is barely affected by the growing motion blur while feature-based methods more or less provide more noisy results. For the increasing disparity, the network is also least affected.

The root mean square errors (RMSEs) of the distance-scaled translational velocity of 4 one-minute flights.

The results of the fastest flight are shown in Fig. 5. SURF’s result is noisy in some parts of the flight mainly because of the big motion blur and scenes lacking texture. For 8.6% of image pairs, SURF has less than 2 inlier matchings. We use zero vectors to show its results in this case. For the other 3 slower flights, SURF has enough matches all the time. The root mean square errors (RMSEs) of the distance-scaled velocity vector’s components along the world frame’s 3 axes are shown in Table V. When using original images, the network outperforms SURF more in faster flights where fewer points are detected. In histogram equalized images, more SURF points are detected and the accuracy is slightly higher than the network. The network performs better on original images than histogram equalized images since the images in the training set are without pre-processing.

![Fig. 4. Comparison between feature-based methods and the network. The top row shows how their accuracy changes with the amount of motion blur. The bottom row shows the effects of the increasing visual disparity. “v/d” denotes the norm of the distance-scaled velocity of the simulated camera.](image)

![Fig. 5. The ratio of velocity to height (v/d) of the number 4 flight, expressed in the world frame. Both methods use original images.](image)

B. Flight Dataset

To obtain sensor data in flight, an MYNT EYE D1000-120 visual-inertial sensor is downward-facing mounted on an Eachine Wizard X220 FPV Racing Drone carrying an NVIDIA Jetson TX2. Its IMU measurements (200Hz) and monocular gray-scale images (30fps) with an exposure duration of 20ms are collected. The images are undistorted and transformed to have the same size and intrinsic matrix as the training set. The top left of Fig. 1 shows an example. The camera’s attitude is estimated by the Madgwick filter [39] using the IMU measurements. The ground-truth velocity is obtained from the OptiTrack motion tracking system at 120Hz. The first column of Table V shows the average and maximum distance-scaled translational velocity of 4 one-minute flights.

The network outperforms SURF more in faster flights where fewer points are detected. In histogram equalized images, more SURF points are detected and the accuracy is slightly higher than the network. The network performs better on original images than histogram equalized images since the images in the training set are without pre-processing.

![Fig. 5. The ratio of velocity to height (v/d) of the number 4 flight, expressed in the world frame. Both methods use original images.](image)

V. CONCLUSION

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

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

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

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “Orb-slam: a versatile and accurate monocular slam system,” IEEE transactions on robotics, vol. 31, no. 5, pp. 1147–1163, 2015.

[4] M. Li and A. I. Mourikis, “High-precision, consistent ekf-based visual-inertial odometry,” The International Journal of Robotics Research, vol. 32, no. 6, pp. 690–711, 2013.

[5] M. J. M. Montiel, O. D. Faugeras, and J. D. Tardos, “Robust visual inertial odometry using a direct ekf-based approach,” in 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 298–304, IEEE, 2015.

[6] T. Qin, P. Li, and S. Shen, “Vins-mono: A robust and versatile monocular visual-inertial state estimator,” IEEE Transactions on Robotics, vol. 34, no. 4, pp. 1004–1020, 2018.

[7] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, “Robust stereo visual inertial odometry for fast autonomous flight,” IEEE Robotics and Automation Letters, vol. 3, no. 2, pp. 965–972, 2018.

[8] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, “Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam,” arXiv preprint arXiv:2007.11989, 2020.

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

[10] M. Trajković and M. Hedly, “Fast corner detection,” Image and vision computing, vol. 16, no. 2, pp. 75–87, 1998.

[11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Osb: An efficient alternative to sift or surf,” in 2011 International conference on computer vision, pp. 2564–2571, Ieee, 2011.

[12] P. Fohr, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muegos, and D. Scaramuzza, “Alphapilot: Autonomous drone racing,” arXiv preprint arXiv:2005.12813, 2020.

[13] 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.

[14] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia, “Exploring representation learning with cnns for frame-to-frame ego-motion estimation,” IEEE robotics and automation letters, vol. 1, no. 1, pp. 18–25, 2015.

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

[16] S. Wang, R. Clark, H. Wen, and N. Trigoni, “End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks,” The International Journal of Robotics Research, vol. 37, no. 4-5, pp. 513–542, 2018.

[17] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, “Unsupervised learning of depth and ego-motion from video,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1567–1574, 2017.

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

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

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

[21] J. Bian, Z. Li, N. Wang, H. Zhan, C. Shen, M.-M. Cheng, and I. Reid, “Unsupervised scale-consistent depth and ego-motion learning from monocular video,” in Advances in Neural Information Processing Systems, pp. 35–45, 2019.

[22] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3354–3361, IEEE, 2012.

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

[24] N. J. Sanket, C. D. Singh, C. Fermüller, and Y. Aloimonos, “Prgbflow: Benchmarking swap-aware unified deep visual inertial odometry,” arXiv preprint arXiv:2006.06753, 2020.

[25] O. D. Faugeras and F. Lustman, “Motion and structure from motion in a piecewise planar environment,” International Journal of Pattern Recognition and Artificial Intelligence, vol. 2, no. 03, pp. 485–508, 1988.

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

[27] M. Jaderberg, K. Simonyan, A. Zisserman, et al., “Spatial transformer networks,” in Advances in neural information processing systems, pp. 2017–2025, 2015.

[28] D. Sun, S. Roth, and M. J. Black, “A quantitative analysis of current practices in optical flow estimation and the principles behind them,” International Journal of Computer Vision, vol. 106, no. 2, pp. 115–137, 2014.

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

[30] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[31] X. glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp. 249–256, 2010.

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

[33] A. Kendall, M. Grimes, and R. Cipolla, “Posenet: A convolutional network for real-time 6-dof camera relocalization,” in Proceedings of the IEEE international conference on computer vision, pp. 2938–2946, 2015.

[34] C. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4700–4708, 2017.

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

[36] 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.

[37] G. De Croon, H. Ho, C. De Wagter, E. Van Kampen, B. Remes, and Q. Chu, “Optic-flow based slope estimation for autonomous landing,” International Journal of Micro Air Vehicles, vol. 5, no. 4, pp. 287–297, 2013.

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

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