Towards fast ego-motion estimation using a two-stream network

Jose Arturo Cocoma-Ortega
Instituto Nacional de Astrofisica, Optica y Electronica (INAOE)
Luis Enrique Erro 1, Tonantzintla, Puebla, Mexico
cocoma@inaoep.mx

Jose Martinez-Carranza
Instituto Nacional de Astrofisica, Optica y Electronica (INAOE)
Luis Enrique Erro 1, Tonantzintla, Puebla, Mexico
carranza@inaoep.mx

Abstract

Autonomous navigation is a challenging task that requires solving individual problems such as the camera pose. Even more, if agile motion is performed in the navigation, the problem becomes complex due to the necessity to know the pose (trajectory generated) as fast as possible to continue with the agile motion. Several works have proposed approaches based on geometric algorithms and deep learning-based solutions that reduce error in estimation. Still, the prediction is performed at 30Hz on average in most cases. It is desirable to increase the frequency of operation to allow cameras with higher frame rates to capture the motion correctly. Motivated by the latter, we propose a two-stream network that allows predicting pose at a frame rate up to $78fps$ *with an average relative error of* $2.21m$.

1. Introduction

The estimation of motion performed by an object (such as a robot, vehicle, or person) using only visual information (obtained from a camera or cameras) is challenging in robotics, augmented reality, and wearable computing.

In recent years, many approaches have been proposed using deep learning to offer robust solutions with the least possible error [1, 21, 22]. Despite the efforts to offer robust solutions, they have become complex having an operating frequency of around 30Hz in most cases. Nevertheless, it is desirable to use cameras with a higher frame rate for agile motion to capture all the movement performed without losing its detail.

We propose a methodology based on a two-stream CNN architecture for the latter. Our main idea is to feed the two-stream network with two stacks of 6 consecutive images, one with grey-scale images and another with edge images using the well known Canny edge detector.

The paper has been organised as follows; Section 2 reviews the most related works. Section 3 briefly describes our proposed methodology. The experiments are shown in section 4. Finally, section 5 presents the final remarks of our proposal.

2. Related work

Since it was introduced in the early 2000s, visual SLAM (Simultaneous Localisation and Mapping) has been one of the mos used and improved algorithms for pose estimation. Traditional computer vision algorithms propose the use of different feature extractors such as ORB descriptors [16] in which a 3D dense map is generated with the features extracted. Also, the optical flow has been combined with extractors to improve estimation [2, 12, 13].

With the growth of Deep Learning, proposals based on CNN architectures have been increased. DeepVO [20] uses a stack of two consecutive RGB images to pose estimation, achieving an average translation error (t_{err}) of 5.96%.

Exploring different techniques, Zhou et al. [25] design a cascade architecture of multiple sub-networks, achieving a low relative error of 1.4% (t_{err}), but with a computational cost high; they need 129.56ms (7.7FPS) to estimate each pose. Some works estimate optical flow to aid the correction of camera pose [24]. Despite can achieve a competitive error of 3.41% (t_{err}), they can perform it only at 12.5Hz.

Lu et al. [11] use CannyLines to enhance RGB-D frames and make predictions up to 35Hz.

3. Methodology

As shown in the work of Cocoma et al. [6], using only monocular grey-scale images allow 3D pose estimation for a specific context. Motivated by this idea, we create a stack of 6 consecutive grey-scale images to input a CNN for extracting features and estimating 3D camera pose.

The most predominant element in images is the floor, which may not contribute significantly as other parts in the scene do. To avoid the extraction of many features from the floor and motivated by the work of Klein [10] that shows that prominent elements such as edges are preserved in agile motion and can represent better the motion, we propose to use an additional stack of 6 consecutive edge images. Then, we propose a two-stream network based on inception modules [18], that takes as input a stack of grey-scale images in the first branch and edge images in the other.

First, we extract four patches of size 56x56 pixels from each input that feeds each branch. Then, we combine each branch output. A final block of 3 inception modules precedes a final layer formed by two MLP (multi-layer perceptron) layers with 1024 neurons and three neurons, respectively, to perform regression.

We built the grey-scale stack using the frames from the dataset already in grey-scale, and we resized them to 224x224 pixels. Using the Canny algorithm [4] provided by OpenCV [3], we obtain the edge images.

Our network learns to predict the relative motion between the last two frames in the stack. To convert world pose (p^w) to relative motion (p^c), we use the following equation:

$$p_i^c = Rot^T(q_{i-1})(p_i^w - p_{i-1}^w) \tag{1}$$

where p_i^c is the relative motion between p_i^w, p_{i-1}^w; p_i^w is the position in the world; q_{i-1} is the orientation in the world and Rot^T is the transpose rotation matrix of q.

To estimate the world pose, we use the world orientation q^w, thus:

$$p_i^w = Rot(q^w) * p_i^c + p^w \tag{2}$$

4. Results

To test our proposal, we experiment with a Drone racing dataset. In this section, we described all the details of the experimentation performed.

4.1. Dataset

WWe used the UZH-FPV Drone racing indoor dataset [7]. This dataset contains sequences from indoor and outdoor scenes recorded from the first-person-view quadrotor flown aggressively by an expert pilot.

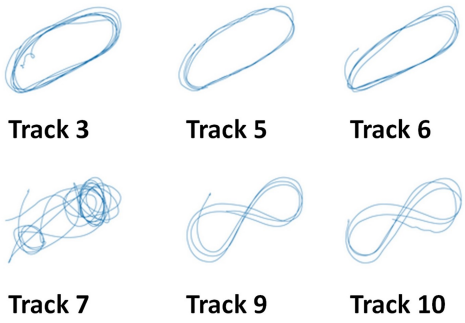

Figure 1. Sequences for the indoor dataset. Besides the dataset containing nine sequences, only six have ground-truth labels. Tracks nine and ten were selected for our experiments.

Dataset provides ground-truth labels for some sequences to allow learning.

We selected the indoor dataset for our experiments formed by nine tracks. Some tracks follow an oval shape, others an eight, and one with a free-form trajectory. Only six of the nine sequences contain ground-truth labels (see Fig. 1). Due to the challenging nature of the dataset, we selected tracks that describe the shape of an eight (tracks 9 and 10). Track 9 was used for training and track 10 for evaluation. Track nine contains three laps in the sequence with 1007 images and labels. Similarly, track ten has 828 images with their respective labels.

4.2. Implementation

We use a laptop with an i5-9300 CPU (octa-core), 32 GB of RAM, and a Geforce RTX 2060 with 6GB VRAM for training and testing. For the implementation of our methodology, we used the Keras API 2.6.0 [5] and TensorFlow 2.6.2 framework [8] implemented in Python 3.6.9 [19]. We use ROS [17] to communicate the camera topics with CNN on Ubuntu 18.04 with CUDA 11.2 [15].

4.3. Training process

To train our network, we selected the Adam optimiser ([9]) with a learning rate = 0.01. To perform the regression, we use ReLU ([14]) as activation function, and for loss function, we use the euclidean distance (see Eq. (3)).

$$loss(I) = \|\hat{x} - x\|_2 \tag{3}$$

We trained our network for 100 epochs with a batch size = 8.

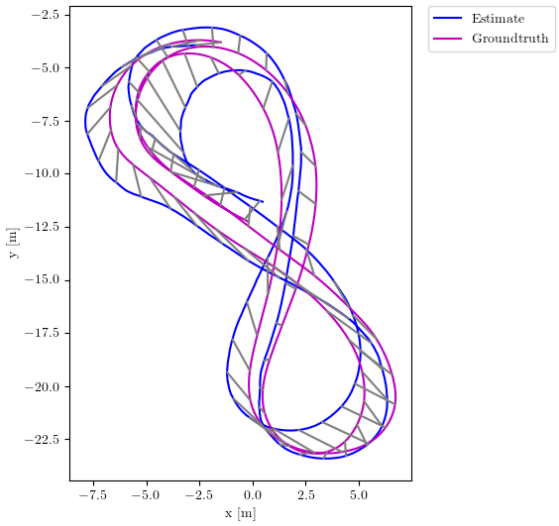

Figure 2. Top view trajectory 10. Our proposal estimates the blue trajectory, and the red one is the ground-truth trajectory. As can be seen, the estimated follows the ground-truth trajectory, with the max errors at the turns

4.4. Experiments

The experiments were conducted as follows, first, training our network with track 9, then the unseen track 10 was used for evaluating the prediction. We perform an ablation study of the effect of different thresholds for the Canny detector. We vary the lower threshold from 50 to 250 and the upper one from 100 to 300. Table X shows the results of the ablation study.

We can notice from Tab. 1 that the better results are with, the lower value and the upper value, being this last the best.

We used the evaluation trajectory provided by the dataset's authors [23]. Figure 2 shows the top view of the trajectory estimated compared with the ground-truth trajectory. As can be seen, the estimated trajectory (blue) follows the shape of the trajectory showing the max errors at the turns due to the aggressive change in motion. Additionally, Fig. 3 shows the relative errors for segments of the trajectory.

To calculate the time needed to perform estimation, we measured the time expended in each step of the methodology; this includes the Canny edge detector, rescaling image, generating the two stacks (grey and edge, respectively) and network prediction. We calculated the times in the ablation study, given an average time of $12.8ms$ (milliseconds) to estimate each pose.

As it can be seen, besides the t_{rel} is around $2m$ (metres), the estimation follows the ground-truth trajectory closely, and it can be estimated up to 78Hz, which surpasses most of the works reported in the literature. With a fast estimation frequency, it is possible to develop systems that respond in less time when high-speed flights or with agile movements are carried out. The error may be acceptable if it is considered that very high precision may not be required in agile or high-speed flights.

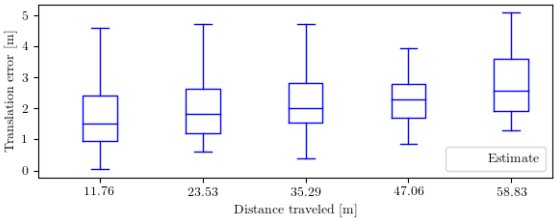

Figure 3. Relative translation error for estimated trajectory 10.

lower t_h	upper t_h	$t_{rel}(m)$	RMSE
50	100	2.26	1.75
50	150	2.55	1.97
50	200	2.42	1.94
50	250	2.73	2.08
50	300	3.08	2.08
100	150	2.43	1.82
100	200	2.43	1.84
100	250	2.53	1.93
100	300	2.52	2.02
150	200	2.42	1.96
150	250	2.43	1.88
150	300	2.37	1.79
200	250	2.48	1.94
200	300	2.74	2.07
250	**300**	**2.21**	**1.70**

Table 1. Ablation study for Canny thresholds. Both errors are in metres.

5. Conclusions

We have explored using a stack of consecutive grey-scale images and an additional stack of edge images to pose prediction in this work. The combination of grey-scale and edge images to feed a two-stream network brings a better prediction than using only grey-scale. Besides the average error in the prediction of 2.21m, we showed that prediction follows the trajectory shape with the advantage of performing estimation at a frequency-time up to 78Hz, which improves most of the related works.

We will continue investigating combining grey-scale images with edges and testing our proposal with more tracks available in the UZH FPV Drone racing Dataset.

References

[1] Debaditya Acharya, Ruwan Tennakoon, Sundaram Muthu, Kourosh Khoshelham, Reza Hoseinnezhad, and Alireza Bab-Hadiashar. Single-image localisation using 3d models: Combining hierarchical edge maps and semantic segmentation for domain adaptation. *Automation in Construction*, 136:104152, 2022. 1

[2] J. Bang, D. Lee, Y. Kim, and H. Lee. Camera pose estimation using optical flow and orb descriptor in slam-based mobile ar game. In *2017 International Conference on Platform Technology and Service (PlatCon)*, pages 1–4, Feb 2017. 1

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 2

[4] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986. 2

[5] François Chollet et al. Keras. https://keras.io, 2015. 2

[6] J. Arturo Cocoma-Ortega and J. Martinez-Carranza. A compact cnn approach for drone localisation in autonomous drone racing. *J Real-Time Image Proc*, 2021. 2

[7] Jeffrey Delmerico, Titus Cieslewski, Henri Rebecq, Matthias Faessler, and Davide Scaramuzza. Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6713–6719, 2019. 2

[8] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 2

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 2

[10] Georg Klein and David Murray. Improving the agility of keyframe-based SLAM. In *Proc. 10th European Conference on Computer Vision (ECCV'08)*, pages 802–815, Marseille, October 2008. 2

[11] Junxin Lu, Zhijun Fang, Yongbin Gao, and Jieyu Chen. Line-based visual odometry using local gradient fitting. *J. Vis. Commun. Image Represent.*, 77:103071, 2021. 1

[12] S. Mansur, M. Habib, G. N. P. Pratama, A. I. Cahyadi, and I. Ardiyanto. Real time monocular visual odometry using optical flow: Study on navigation of quadrotors uav. In *2017 3rd International Conference on Science and Technology - Computer (ICST)*, pages 122–126, July 2017. 1

[13] V. More, H. Kumar, S. Kaingade, P. Gaidhani, and N. Gupta. Visual odometry using optic flow for unmanned aerial vehicles. In *2015 International Conference on Cognitive Computing and Information Processing(CCIP)*, pages 1–6, March 2015. 1

[14] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress. 2

[15] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020. 2

[16] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011. 1

[17] Stanford Artificial Intelligence Laboratory et al. Robotic operating system. 2

[18] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[19] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. 2

[20] S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050, May 2017. 1

[21] YanTong Wu, Yang Liu, and XueMing Li. Position estimation of camera based on unsupervised learning. In *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence*, PRAI 2018, pages 30–35. ACM, 2018. 1

[22] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[23] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018. 3

[24] Baigan Zhao, Yingping Huang, Hongjian Wei, and Xing Hu. Ego-motion estimation using recurrent convolutional neural networks through optical flow learning. *Electronics*, 10(3), 2021. 1

[25] Wenhui Zhou, Hua Zhang, Zhengmao Yan, Weisheng Wang, and Lili Lin. Decoupledposenet: Cascade decoupled pose learning for unsupervised camera ego-motion estimation. *IEEE Transactions on Multimedia*, pages 1–1, 2022. 1