Analysis of the Effect of Sensors for End-to-End Machine Learning Odometry

Carlos Marquez Rodriguez-Peral(✉) and Dexmont Peña(✉)

Intel Research and Development, Movidius Group, Kildare, Ireland
carlos.marquez.rp@gmail.com, dexmont.pena@intel.com

Abstract. Accurate position and orientation estimations are essential for navigation in autonomous robots. Although it is a well studied problem, existing solutions rely on statistical filters, which usually require good parameter initialization or calibration and are computationally expensive. This paper addresses that problem by using an end-to-end machine learning approach. This work explores the incorporation of multiple sources of data (monocular RGB images and inertial data) to overcome the weaknesses of each source independently. Three different odometry approaches are proposed using CNNs and LSTMs and evaluated against the KITTI dataset and compared with other existing approaches. The obtained results show that the performance of the proposed approaches is similar to the state-of-the-art ones, outperforming some of them at a lower computational cost allowing their execution on resource constrained devices.

Keywords: Navigation · Visual · Inertial · Odometry
Machine learning · CNN · LSTM

1 Introduction

Motion estimation is one of the main pillars of mobile robotics. It provides a robot with the capability to know its position and orientation in an unknown environment and it can be combined with mapping approaches to develop Simultaneous Localization and Mapping (SLAM), which is essential to perform human assistance and exploration tasks. A robot can use different sources of data to perform such motion estimation depending on the type of sensor: proprioceptive, when it offers the robot's internal information such as Inertial Measurement Units (IMU), or exteroceptive, when it offers information of the robot's surroundings such as cameras or LiDARs. Due to the autonomous nature of a robot, it should be able to perform such motion estimation on board in real time, often resource-limited. Thus, finding a solution that can run in an embedded device under such restrictions is desirable.

L. Leal-Taixé and S. Roth (Eds.): ECCV 2018 Workshops, LNCS 11134, pp. 82–95, 2019.
https://doi.org/10.1007/978-3-030-11024-6_6

This paper explores the performance of different end-to-end machine learning based systems depending on the type of sensor used. The contributions of this paper are:

- Three end-to-end trainable networks using different kind of sensor data are proposed.
- The proposed networks are tested on real world data and compared with other state-of-the-art approaches.

The rest of the paper is organized as follows: Sect. 2 shows existing visual and inertial odometry approaches based on both classical and machine learning techniques. Section 3 shows the architecture of the proposed approaches. In Sect. 4, the training parameters, optimizer and objective function used are explained. The results of the training as well as their performance comparison is done in Sect. 5. Section 6 presents the conclusions of this work and shows the future work that can be done.

2 Related Work

This section analyzes and highlights different works that have been done to solve the pose estimation problem with classic approaches and, more recently, with deep learning techniques.

Cameras capture the surroundings of the robot and can be used to track the robot's movement, this process is known as Visual Odometry (VO) [1]. Classic VO approaches estimate motion from geometry constraints, and can be divided into two groups: sparse feature based methods and direct methods. On one hand, sparse feature based methods extract and match feature points to estimate the motion between frames such as in LIBVISO2 [2]. In addition, some VO approaches such as ORB-SLAM [3] add and maintain a feature map in order to correct the drift suffered due to the presence of outliers and noisy images. On the other hand, direct [4] and semi-direct [5,6] methods use all the image pixels to estimate the pose by minimizing the photometric error between consecutive images.

However, classical VO approaches need external information (such as camera height or templates) to perceive the scale and recover distances in real world units. Castle et al. address this problem by combining a monocular SLAM system with object recognition [7]. Pillai et al. combine ORB-SLAM [3] with object recognition, introducing a multi-view object proposal and an efficient feature encoding method [8].

Nevertheless, VO systems are not reliable in the presence of rapid movements or when there are sudden changes in illumination. To solve this lack of reliability, the camera information can be combined with inertial sensors, which can provide acceleration and angular rate information. This sensors usually offer data at much higher frequencies (about 10 times faster) than a camera. Therefore,

inertial information can be used to overcome VO systems' weaknesses in the case of rapid camera motion.

Visual-Inertial Odometry (VIO) systems take advantage of visual and inertial information to provide position and orientation estimations. In state-of-the-art methods, the visual-inertial data fusion is done by using probabilistic filter approaches such as Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF), which are compared in [9]. In [10], the visual-inertial data fusion is performed with an EKF based system, which they used to compare different fusion models using only gyroscope data, or gyroscope and accelerometer data. Other variations of the EKF have been used for this purpose, such as the Multi-state Constraint Kalman Filter (MSCKF). Mourikis et al. implemented a MSCKF system in which several past camera poses were used to detect static features and add a constraint to the state vector [11].

In recent years, deep learning approaches have overcome the weaknesses of classic VO approaches, such as lack of robustness to blurred or noisy images or when changes in illumination or occlusion occurs. Convolutional Neural Networks (CNN) have shown to perform well even with blurred and noisy images, providing a robust method for extracting image features [12]. CNNs have been also used to compute the Optical Flow between two consecutive images [13,14]. The Optical Flow represents the change in location of the objects on the camera view [15], therefore it is related to the motion that the camera has experienced between two consecutive frames. The image features extracted by the Optical Flow network in [13] have been used in [16] along with two Long Short Term Memory (LSTM) [17] layers to implement a monocular VO system in an end-to-end deep learning manner, clearly outperforming a classic monocular VO approach based on LIBVISO2 [2].

VIO approaches based on probabilistic filters for sensor fusion may require a hard and complex calibration process in order to bring camera and IMU measurements to the same reference coordinate system [18,19]. In [20] the calibration process is done in real time while a tracking system is running, adding complexity to the filtering process. Moreover, some IMU's parameters are difficult to model, such as the noise scaling over the measurements found in most commercial IMUs [21]. Deep Learning techniques have been used in order to solve the issues with the sensor fusion process. In [22], Rambach et al. use an LSTM to track past IMU raw measurements (accelerometer and gyroscope) to estimate the pose of a robot, which is then fused with a VO system. LSTMs have been also used in VINet [23] to extract encoded features from IMU's raw measurements. These encoded features are combined in a features vector with features extracted from a CNN, being this features vector tracked over time by a second LSTM, which provides a pose estimation of a robot. VINet approach outperforms OKVIS [24], which is an optimization-based sensor fusion approach.

3 Proposed Approaches

This work explores the performance of different end-to-end trainable neural network architectures, varying the amount and type of the input. Three different

networks have been trained for this purpose. The first one takes as input RGB images, the second one takes as input IMU raw measurements and the last one is a combination of the previous networks, taking as input both RGB images and IMU raw measurements. All the networks are trainable in an end-to-end manner, eliminating any need of calibration or preprocessing.

The networks have been trained to produce at every frame a pose estimation relative to the previous frame. Each pose estimation represents a transformation, which is usually represented as elements of the Special Euclidean Group of transformations SE(3), which is described in [25]. All the transformations represented in SE(3) (Eq. 1) are composed of a rotation matrix and a translation vector, being that rotation matrix part of the Special Orthogonal group SO(3), described in [25].

$$SE(3) : (R|T),\ R \in SO(3),\ T \in \mathbb{R}^3 \tag{1}$$

Finding a transformation in the SE(3) is not straightforward for the network because R has to be orthogonally constrained. Thus, to make easier the learning process, the estimated transformations are represented in the Lie Algebra se(3) (Eq. 2) of SE(3).

$$se(3) : (\omega|t),\ \omega \in so(3),\ t \in \mathbb{R}^3 \tag{2}$$

The pose estimations in se(3) are 6-D vectors and are not orthogonally constrained. Once estimated, the poses in se(3) can be converted into transformations of the SE(3) by doing an exponential mapping: $se(3) \rightarrow SE(3)$ (Eq. 10) as described in [25].

$$\theta = \sqrt{\omega^T \omega} \tag{3}$$

$$A = \frac{sin\theta}{\theta} \tag{4}$$

$$B = \frac{1 - cos\theta}{\theta^2} \tag{5}$$

$$C = \frac{1 - A}{\theta^2} \tag{6}$$

$$\omega_x = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \tag{7}$$

$$R = I + A\omega_x + B\omega_x^2 \tag{8}$$

$$V = I + B\omega_x + C\omega_x^2 \tag{9}$$

$$se(3) \rightarrow SE(3) : exp(\omega|t) = (R|Vt) \tag{10}$$

Matrices R and V can be calculated using Eqs. 8 and 9, respectively. A, B, C and θ can be obtained through Eqs. 4, 5, 6 and 3. ω_x matrix is composed by ω values (Eq. 7).

3.1 Network 1: Visual Odometry

The first proposed network is illustrated in Fig. 1. It takes as input two consecutive RGB images, which are stacked composing an input tensor of size 512×384 with 6 channels. This image size has been used because it has shown to contain enough features while resulting in a light CNN. FlowNetS [13] has been used to extract images' features, as its good performance for motion estimation was shown in [16,23]. This network was trained on a synthetic dataset to learn how to estimate the Optical Flow between frames, which represents the motion undergone by the robot over time.

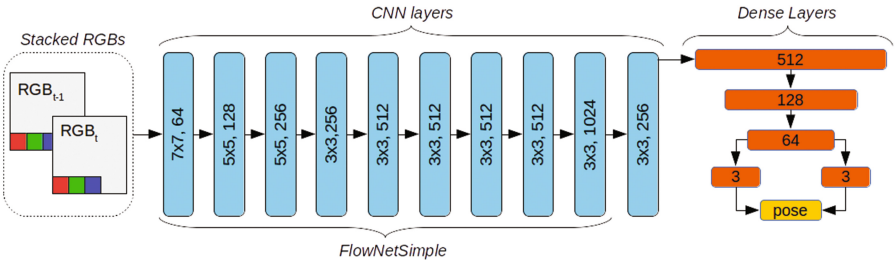


Fig. 1. Architecture of the proposed visual odometry network. All the layers are followed by a LeakyReLU activation layer except the last two FC layers.

FlowNetS is taken up to its 9th convolutional layer, followed by an additional convolutional layer to reduce the output size of the CNN to a $2 \times 3 \times 256$ tensor. After the CNN, a series of Fully Connected layers combine the extracted features to produce an output 6-D vector pose that represents the transformation of the current frame (t) relative to the previous frame $(t-1)$, expressed in the Lie Algebra of SE(3).

3.2 Network 2: Inertial Odometry

The second proposed network is shown in Fig. 2. In this case, only inertial data is used as input to the network. Specifically, the input is a subsequence composed by 10 6-D vectors with the x-y-z raw data components from accelerometer and gyroscope. This subsequence of 10 measurements is ordered in time, being the last one the most up to date, encoding the motion that the sensor has experienced over time.

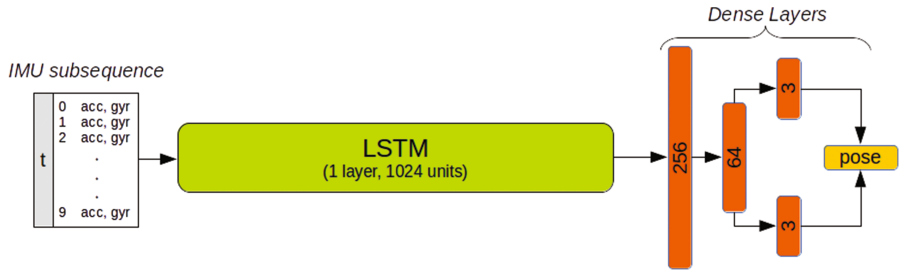


Fig. 2. Architecture of the proposed inertial odometry network. The first two FC layers are followed by a LeakyReLU activation layer.

An LSTM is used as a regression layer to track measurements over the subsequence and extract motion information as it is able to store in its hidden states short and long term dependencies produced by past inputs. Each input is combined with the hidden state as it passes through the LSTM, finding temporal correspondences between the current and past measurements. The LSTM used has 1 layer, 1024 units and is followed by 4 Fully Connected layers that output a 6-D vector representing the transformation undergone by the robot from the last to the first element of the subsequence. These architecture and parameters have been selected as a result of their performance in a montecarlo analysis, which was done to explore different combination of layers.

3.3 Network 3: Visual-Inertial Odometry

The third proposed network is shown in Fig. 3. It combines the networks 1 and 2, taking advantage of both visual and inertial sensors. The input is a pair of consecutive RGB images and a subsequence of 10 inertial measurements following the idea of the VIO networks, which combine inertial and visual data to overcome the weaknesses of each other, as stated in Sect. 1. The structure of the VO network remains up to its third Fully Connected layer. Similarly, the Inertial Odometry (IO) network is used up to its second FC layer. The idea behind this is to maintain both VO and IO networks until the last layer that provides useful features.

Then, vision and inertial feature vectors are concatenated into a 128-D vector and passed through three FC layers to output a pose estimation. As before, each

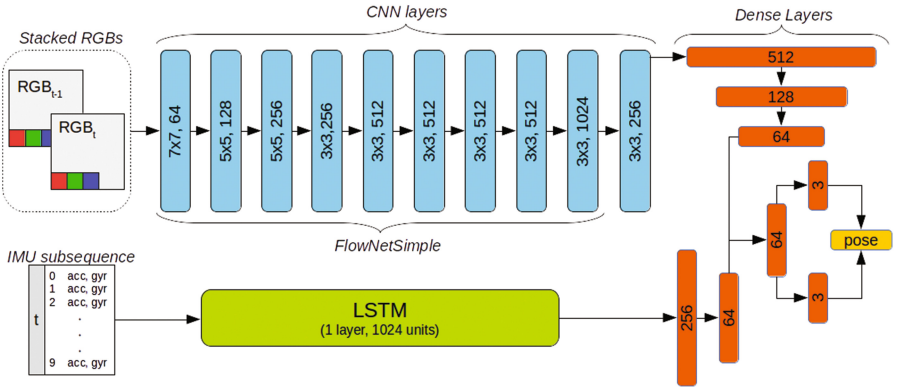


Fig. 3. Architecture of the proposed sensor fusion odometry network. The structure of the VO and IO networks remains, so LeakyReLU activation layers are applied after every layer except after the last two FC layers.

pose estimation represents the transformation undergone by the robot at the current frame with respect to the previous one.

4 Training Setup

This section describes the parameters used to train the networks as well as the dataset structure. All the networks described have been trained separately but maintaining the same parameters and training data in order to be able to do a fair comparison between them.

The data used for training is part of the raw data section of KITTI Vision Benchmark Suite [26], which involves a car based odometry problem suitable for the analysis carried out in this paper. The odometry dataset is composed by 22 sequences, being the first 11 of these provided with its groundtruth transformations. Sequences 11–22 are intended to be used as evaluation, so no groundtruth is provided. Sequences 00, 02, 08 and 09, which contain the highest number of frames, are used for training and sequences 05, 07 and 10 for evaluation. The training data is augmented by randomly applying gaussian noise, gaussian blur and changes in intensity to the images as follows:

- 2/3 of the data: gaussian noise (mean = 0, standard deviation = [0, 32]) and change in pixels intensity [−25%, 25%]
- 1/3 of the data: gaussian blur with kernels 3, 5 and 7.

After augmenting the data, the training dataset has a total of 22912 image frames. The images recorded in the dataset are sampled at 10 Hz as well as the groundtruth. The IMU data arrives at 100 Hz, meaning that there are 10 IMU measurements per image frame. However, there are frames where some IMU data are missing. In that case, the first IMU measurement of the frame is used to pad the missing measurements to fill the subsequence.

The loss function (Eq. 11) used represents the euclidean distance between every estimated relative pose and its respective groundtruth, expressed in se(3). This loss function was inspired by VINet paper [23].

$$\mathcal{L}_{se(3)} = \Sigma||\omega - \hat{\omega}|| + \beta||t - \hat{t}|| \tag{11}$$

ω, $\hat{\omega}$, t and $\hat{t}$ represent the estimated and groundtruth rotation and translation in *se(3)*, respectively. The parameter β is useful to balance the different magnitude order between ω and t, and it is fixed to 0.1 in all trainings. Nesterov Accelerated Gradient (NAG) [27] is used as optimizer (Eqs. 12 and 13). It speeds up the convergence with respect to the standard Gradient Descent, as stated in [28], measuring the gradient of the loss function not at the local position but slightly ahead in the direction of the momentum, m.

$$m = \beta m + \lambda \nabla (w^{se(3)} + \beta m) \tag{12}$$

$$w^{se(3)} = w^{se(3)} - m \tag{13}$$

β acts as a friction factor, preventing the momentum from growing too large and λ is the learning rate. The weights $w^{se(3)}$ are then updated according to m. For training, a friction factor $\beta = 0.9$ was used. Senior et al. performed an empirical study of different learning rate schedules [29], showing that implementing an exponential schedule (Eq. 14) leads to a faster convergence and it is easier to implement in comparison with other methods such as the performance schedule.

$$\lambda(t) = \lambda_0 2^{-t/r} \tag{14}$$

An initial learning rate (λ_0) of 10^{-5} and a step (r) of 50 were used. With these parameters, the learning rate is divided by 2 every 50 iterations. All the networks have been implemented on TensorFlow and trained using a NVIDIA GeForce GTX Titan X GPU. In order to reduce the training time, FlowNetS' [13] weights were frozen during training.

5 Results

This section shows the evaluation results of all the networks. Initially, the VO and VIO are compared separately with existing approaches that use the same type of data. Then, the evaluation performance of all the networks proposed in this paper are compared.

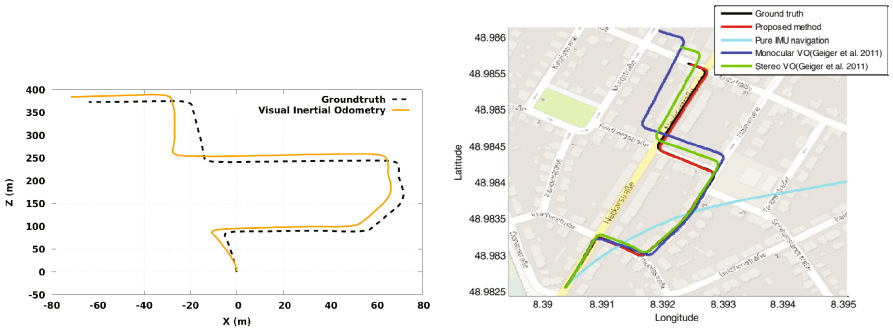
The proposed VO network has been evaluated using the metrics proposed in KITTI's odometry development kit. According to these metrics, the network is executed on sequences 05, 07 and 10, getting the absolute pose for every frame with respect to the first one. Then, the Root Mean Squared Error (RMSE) is calculated for different trajectory lengths (100 m, 200 m, 300 m, ...800 m) over the sequence. These results are shown in Table 1 along with VISO2_M and DeepVO for comparison.

Table 1. All the errors represent the average RMSE for all the possible sequence lengths. t_{rel} is translation error and r_{rel} is rotation error. DeepVO and VISO2_M results are taken from [16].

Seq	t_{rel} (%)			r_{rel} (deg/m)		
	Proposed VO	VISO2_M	DeepVO	Proposed VO	VISO2_M	DeepVO
05	14.03	19.22	2.62	0.10	0.17	0.03
07	28.6	23.61	3.91	0.21	0.29	0.04
10	11.83	41.56	8.11	0.08	0.32	0.08

The VO network proposed in this paper outperforms VISO2_M in terms of both translation and rotation errors for Sequences 05 and 10, being slightly worse in translation for Sequence 07.

The proposed VIO network has been compared with the method proposed by Hu and Chen in [30], which uses a MSCKF to perform monocular VIO. They evaluate their VIO method in the first section of sequence 00. Therefore, in order to do a fair comparison, the proposed VIO network has been trained again eliminating the first 1000 frames of sequence 00 from the training dataset. Then, the trained network has been evaluated in frames 0–800 of sequence 00, which involve a total translation distance of 556.1 m. The estimated trajectory is shown in Fig. 4 and the end point translation and rotation errors are shown in Table 2.



(a) Proposed VIO method estimated trajectory.

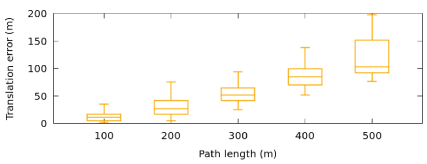
(b) Hu and Chen VIO method estimated trajectory.

Fig. 4. Estimated trajectory comparison between the proposed VIO and Hu and Chen VIO methods. The evaluation trajectories have been made over the first frames of sequence 00. Subfigure (b) has been taken from [30], in which *Proposed method* refers to the VIO method proposed by Hu and Chen.

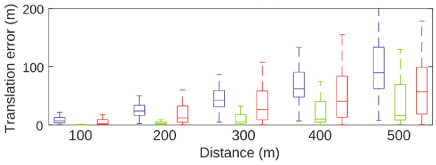
Although the proposed VIO method translation error is considerably bigger than the one obtained by Hu and Chen, most of the contribution to that error comes from Y axis error, which represents the height of the robot. X, Y and

Table 2. Final point position and orientation error for the proposed VIO method and the method proposed by Hu and Chen [30]. The translation error is shown both in terms of absolute error of the final point and of percentage of that error with respect of the total distance covered in frames 0–800.

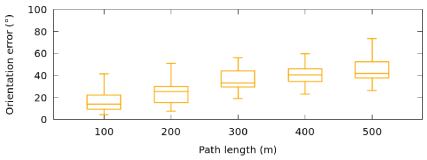
	Proposed VIO method	Hu and Chen VIO [30]
Translation (m)/(%)	37.20/6.68	6.44/1.15
Rotation (deg)	15.64	1.05



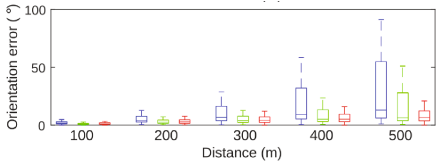
(a) Proposed VIO method translation error.



(b) VINet method translation error.



(c) Proposed VIO method rotation error.



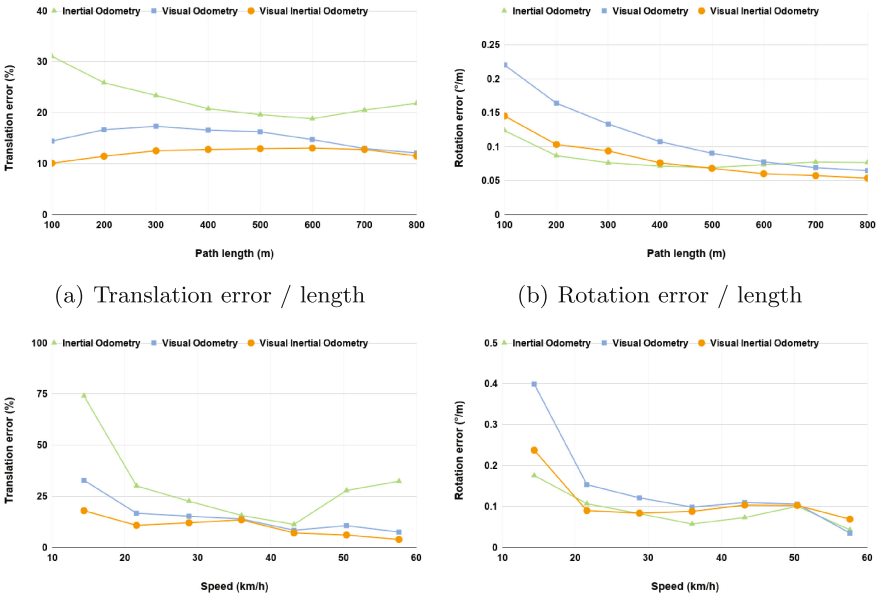
(d) VINet method rotation error.

Fig. 5. The boxplot graphs show the translation and rotation errors distribution for the different path lengths. In VINet error graphs, three different approaches are shown for each path length . According to VINet paper [23], from left to right: VINet vision only, VINet and EKF+Viso2. Subfigures (b) and (d) have been taken from VINet paper.

Z errors for the end point are 6.90, 34.59 and 11.84 respectively. Moreover, the translation percentage errors show that the translation error of the proposed VIO method represents a 6.68% with respect to the total distance covered, meanwhile this percentage is 1.15% for the Hu and Chen VIO approach.

In addition, the VIO network has been compared with VINet [23]. For this purpose, the network has been executed over the whole sequence 10 and the evaluation results have been calculated with KITTI metrics for paths of length: 100 m, 200 m, 300 m, 400 m and 500 m. These results are shown in Fig. 5 along with VINet results.

In order to compare the performance of the networks proposed, all of them have been evaluated on sequences 05, 07 and 10. The results of this evaluation are presented in Figs. 6 and 7. The checkpoint size of the proposed IO, VO and VIO networks are 34 MB, 173 MB and 207 MB respectively. Figure 6 shows graph errors for translation and rotation depending on the path length and speed. On one hand, the IO network outperforms the VO network in terms of rotation and it gets similar results to the VIO network. The reason behind this result



(a) Translation error / length

(b) Rotation error / length

(c) Translation error / speed

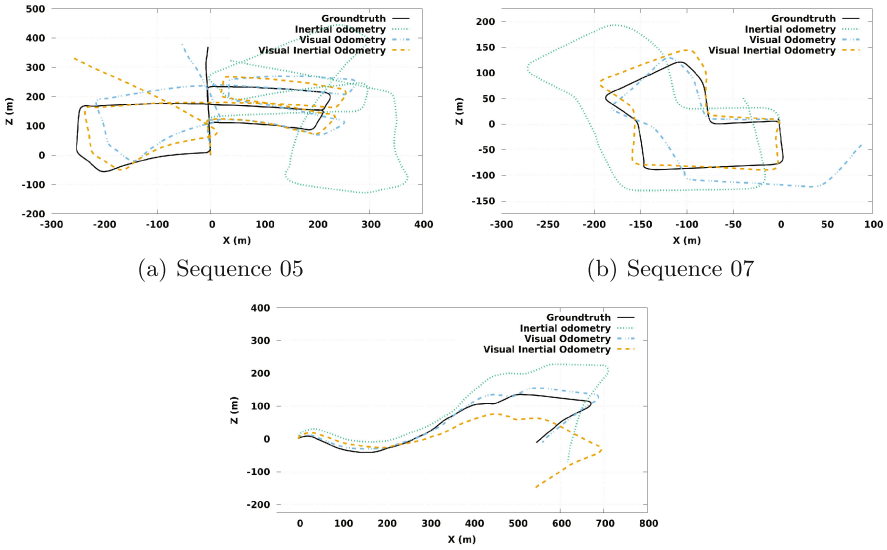
(d) Rotation error / speed

Fig. 6. Average translation and rotation RMSE for different path lengths (100 m to 800 m) and speeds.

might be that inertial information is better suited for movements that imply big changes in the frontal view of the robot. That is because when the robot is turning, the images captured by the camera suffer from temporary changes in illumination and blurring, causing the Optical Flow extracted by the CNN to become unreliable.

On the other hand, the VO network outperforms the IO network in terms of translation error. This result shows that pure IMU based odometry suffer from drift over time. This can be seen in Fig. 7, which shows the estimated trajectory for every network on Sequences 05, 07 and 10.

Analyzing Sequence 07, the IO network, while maintaining a good performance in rotation, gets drifted in translation. In contrary, the VO network performs better in terms of translation, but fails in estimating rotations. The best performance, in sequences 05 and 07, is achieved by the VIO network. The trajectories obtained with this network show that the combination of visual and inertial information allows the network to provide better estimations both in term of translation and rotation, maintaining a better transformation scale over time.



(a) Sequence 05

(b) Sequence 07

(c) Sequence 10

Fig. 7. Evaluation trajectories for the three network proposed run on Sequences 05, 07 and 10.

6 Conclusions and Future Work

This work proposed, trained and evaluated three end-to-end approaches for Odometry estimation. A performance comparison between them was carried out in order to show how different combinations of a camera and an IMU can lead to different results. The Inertial Odometry network has shown a large drift error over time. However, when it is combined with the Visual Odometry network, the drift is considerably reduced. Moreover, the Visual Inertial Odometry showed a better performance when the robot is turning, outperforming the Visual Odometry network. This showed how the IMU compensates the large displacement of the objects in the camera. These networks have been compared with existing approaches, showing promising results and outperforming (in the case of the Visual Odometry network) classical methods at a smaller memory footprint than existing approaches.

Nevertheless, the proposed networks performance may be improved by increasing the amount of data used for training to include indoor scenarios and drone flying as they present a bigger challenge for Visual Inertial Odometry systems due to the sudden and unstable movements. Further research is being performed on the introduction of sensor reading failures and approaches to overcome them.

References

1. Nistér, D., Naroditsky, O., Bergen, J.: Visual odometry. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004, vol. 1, pp. I-I. IEEE (2004)
2. Geiger, A., Ziegler, J., Stiller, C.: Stereoscan: dense 3d reconstruction in real-time. In: 2011 IEEE Intelligent Vehicles Symposium (IV), pp. 963–968. IEEE (2011)
3. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Trans. Robot. **31**(5), 1147–1163 (2015)
4. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: DTAM: dense tracking and mapping in real-time. In: 2011 IEEE International Conference on Computer Vision (ICCV), pp. 2320–2327. IEEE (2011)
5. Engel, J., Sturm, J., Cremers, D.: Semi-dense visual odometry for a monocular camera. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1449–1456 (2013)
6. Forster, C., Pizzoli, M., Scaramuzza, D.: SVO: fast semi-direct monocular visual odometry. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 15–22. IEEE (2014)
7. Castle, R.O., Klein, G., Murray, D.W.: Combining monoslam with object recognition for scene augmentation using a wearable camera. Image Vis. Comput. **28**(11), 1548–1556 (2010)
8. Pillai, S., Leonard, J.: Monocular slam supported object recognition. arXiv preprint arXiv:1506.01732 (2015)
9. D'Alfonso, L., Lucia, W., Muraca, P., Pugliese, P.: Mobile robot localization via EKF and UKF: a comparison based on real data. Robot. Auton. Syst. **74**, 122–127 (2015)
10. Bleser, G., Stricker, D.: Advanced tracking through efficient image processing and visual-inertial sensor fusion. Comput. Graph. **33**(1), 59–72 (2009)
11. Mourikis, A.I., Roumeliotis, S.I.: A multi-state constraint Kalman filter for vision-aided inertial navigation. In: 2007 IEEE International Conference on Robotics and Automation, pp. 3565–3572. IEEE (2007)
12. Mayer, N., et al.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4040–4048 (2016)
13. Dosovitskiy, A., et al.: Flownet: learning optical flow with convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2758–2766 (2015)
14. Zhu, Y., Lan, Z., Newsam, S., Hauptmann, A.G.: Guided optical flow learning. arXiv preprint arXiv:1702.02295 (2017)
15. Raudies, F.: Optical flow. http://www.scholarpedia.org/article/Optic_flow. Accessed 26 June 2018
16. Wang, S., Clark, R., Wen, H., Trigoni, N.: Deepvo: towards end-to-end visual odometry with deep recurrent convolutional neural networks. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 2043–2050. IEEE (2017)
17. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
18. Lobo, J., Dias, J.: Relative pose calibration between visual and inertial sensors. Int. J. Robot. Res. **26**(6), 561–575 (2007)

19. Petersen, A., Koch, R.: Video-based realtime IMU-camera calibration for robot navigation. In: Real-Time Image and Video Processing 2012, vol. 8437, p. 843706. International Society for Optics and Photonics (2012)
20. Weiss, S., Achtelik, M.W., Lynen, S., Chli, M., Siegwart, R.: Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 957–964. IEEE (2012)
21. Skog, I., Händel, P.: Calibration of a MEMS inertial measurement unit. In: XVII IMEKO World Congress, pp. 1–6 (2006)
22. Rambach, J.R., Tewari, A., Pagani, A., Stricker, D.: Learning to fuse: a deep learning approach to visual-inertial camera pose estimation. In: 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 71–76. IEEE (2016)
23. Clark, R., Wang, S., Wen, H., Markham, A., Trigoni, N.: VINet: visual-inertial odometry as a sequence-to-sequence learning problem. AAAI **I**, 3995–4001 (2017)
24. Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., Furgale, P.: Keyframe-based visual-inertial odometry using nonlinear optimization. Int. J. Robot. Res. **34**(3), 314–334 (2015)
25. Eade, E.: Lie groups for 2d and 3d transformations, revised December 2013. http://ethaneade.com/lie.pdf
26. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: the kitti dataset. Int. J. Robot. Res. **32**(11), 1231–1237 (2013)
27. Nesterov, Y.: A method for unconstrained convex minimization problem with the rate of convergence o 1/k̂2. In: Doklady AN USSR, vol. 269, pp. 543–547 (1983)
28. Gron, A.: Hands-on machine learning with scikit-learn and tensorflow: concepts, tools, and techniques to build intelligent systems (2017)
29. Senior, A., Heigold, G., Yang, K., et al.: An empirical study of learning rates in deep neural networks for speech recognition. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6724–6728. IEEE (2013)
30. Hu, J.S., Chen, M.Y.: A sliding-window visual-imu odometer based on tri-focal tensor geometry. In: 2014 IEEE international conference on Robotics and automation (ICRA), pp. 3963–3968. IEEE (2014)