ViT VO - A Visual Odometry technique Using CNN-Transformer Hybrid Architecture

Jayaraj P. B[1,*], *Ebin* J[1], *Karthik* R[2], and *Pournami* P N[1]

[1]National Insitute of Technology Calicut, India,
SED/ISG, Advanced Inertial Systems, ISRO Inertial Systems Unit, Thiruvananthapuram, Kerala, India

Abstract. Localization is one of the main tasks involved in the operation of autonomous agents (e.g., vehicle, robot etc.). It allows them to be able to track their paths and properly detect and avoid obstacles. Visual Odometry (VO) is one of the techniques used for agent localization. VO involves estimating the motion of an agent using the images taken by cameras attached to it. Conventional VO algorithms require specific workarounds for challenges posed by the working environment and the captured sensor data. On the other hand, Deep Learning approaches have shown tremendous efficiency and accuracy in tasks that require high degree of adaptability and scalability. In this work, a novel deep learning model is proposed to perform VO tasks for space robotic applications. The model consists of an optical flow estimation module which abstracts away scene-specific details from the input video sequence and produces an intermediate representation. The CNN module which follows next learn relative poses from the optical flow estimates. The final module is a state-of-the-art Vision Transformer, which learn absolute pose from the relative pose learnt by the CNN module. The model is trained on the KITTI dataset and has obtained a promising accuracy of approximately 2%. It has outperformed the baseline model, MagicVO, in a few sequences in the dataset.

Keywords: Visual Odometry, Deep Learning, Optical Flow, Convolutional Neural Networks, Generative Adversarial Networks, Sequence-based Models

1 Introduction

A fundamental challenge in the domain of autonomous agent applications is precise localization and self-motion estimation of the agent. To achieve autonomous navigation, it is pertinent that the agent keeps track of its position over time while also observing and adjusting to the changes in its working environment. Wheel Odometry, Laser/Ultrasonic Odometry, Global Position System (GPS), Inertial Navigation System (INS), and Visual Odometry (VO) represent the broad spectrum of the techniques used for simultaneous localization and mapping. Autonomous robot applications require that the errors in position estimation be contained to within a few centimeters [1].

Visual Odometry is the process of estimating the trajectory of motion of an autonomous agent using only the images taken by cameras attached to it. An autonomous agent is responsible for being aware of the environment and other moving objects in the scene, as well

*e-mail: jayarajpb@nitc.ac.in

as its own relative movements. VO is unaffected by wheel slippage in uneven terrains or other unfavorable conditions. The rate of local drift under VO is smaller than the drift rate of wheel encoders and low-precision INS [2]. Unlike sensor-based localization systems, VO systems do not emit any discernible radiation into the environment. Moreover, a VO system can be easily integrated with other vision-based autonomous systems like terrain detectors to enhance them with more information. Furthermore, the use of consumer-grade cameras instead of expensive sensors allow ease of maintenance and deployability.

The model proposed in this work builds upon work done by Tejas Pandey et al. [3] and Peter Muller et al. [4]. The novel idea in this work is to augment the CNNs described in [3] and [4] with a Vision Transformer (ViT) [5], which has a sequence-based architecture. The ViT learns absolute poses from the relative poses learnt by the CNN preceding it, without accumulating errors across the time-sequence. This helps in significantly cutting down the errors in the final predicted pose values.

2 Visual Odometry Using Optical Flow Estimation

VO involves estimating the self-motion of an autonomous agent using a sequence of images captured using on-board cameras. The motion of the agent induces changes in the sequential images captured by the cameras. Optical flow is defined as the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer (the camera) and a scene (the agent's working environment) [6]. It serves as a good approximation of the true physical motion. In the captured sequence of images, certain features can be identified which help us to calculate the relative displacement of the agent between consecutive frames. For example, for an autonomous car moving on the road, these features could be markings on the road, vehicles moving in other lanes, poles on the road-side etc.

2.1 Mathematical Formulation of Optical Flow Estimation

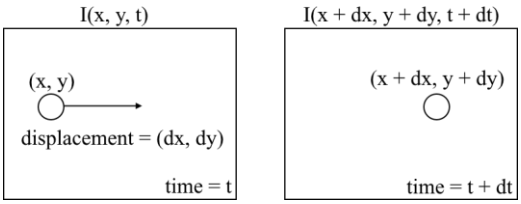

Figure 1. Optical flow of a pixel between consecutive images taken *dt* time apart. The pixel at coordinate (x, y) in the first image has translated to coordinate (x+*dx*, y+*dy*) in the second image in *dt* time.

Estimating the optical flow from a sequence of images is a fundamental task in VO. For this, an image is represented as function of 3 variables - the *X*-coordinate, the *Y*-coordinate and time. So, an image $I(x, y, t)$ gives the intensity at position (x, y) at time t (Figure 1). Optical flow algorithms start by assuming that the image intensity of each visible point does not change considerably between consecutive frames (the *constant intensity* assumption). The typical frame rate of consumer-grade cameras is around 30 FPS, and so, consecutive images are captured tenths of a second apart. Since this interval is very short, it is highly unlikely that pixel intensities change considerably and hence the assumption is justified.

Mathematically, the *constant intensity* assumption states that:

$$\frac{dI(x, y, t)}{dt} = 0$$

$$\frac{dI}{dt} = \frac{\partial I}{\partial x}.\frac{dx}{dt} + \frac{\partial I}{\partial y}.\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

$$\frac{\partial I}{\partial x}.u + \frac{\partial I}{\partial y}.v + \frac{\partial I}{\partial t} = 0$$

Here $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$. u and v are called the *flow vectors*. So the objective is to solve this differential equation for u and v.

Based on *how* they operate, optical flow algorithms can be classified as feature-based methods or direct methods. Feature-based methods require a pre-processing step in which a set of anchors should be identified to map the motion across the sequence. Direct methods compute flow vectors directly from the pixel intensity values. Another classification based on the *type* of output produced, is as sparse-flow methods and dense-flow methods. Sparse optical flow algorithms compute flow vectors for only some "interesting points" in the images. The model proposed in this work uses a dense-flow method.

2.2 Representing Optical Flow

Optical flow is calculated for consecutive image pairs. Calculating the optical flow simply means finding the tuple (u, v) for each pixel, where u is the flow value in X-direction and v is the flow value in Y-direction. Consider two consecutive images I_t and I_{t+1} taken by a moving agent at time t and $t + 1$ respectively. To calculate the optical flow for this image pair, we have to find the (u, v) tuple for each pixel in I_{t+1} with respect to the corresponding pixel in I_t.

Optical flow can be represented in two ways: as an (u, v) image or as a color-coded image. An (u, v) image consists of two channels. For each pixel P in the (u, v) image, the intensity value of P in the first channel represents the flow value in X-direction and the intensity value of P in the second channel represents the flow value in the Y-direction.

3 Proposed Design

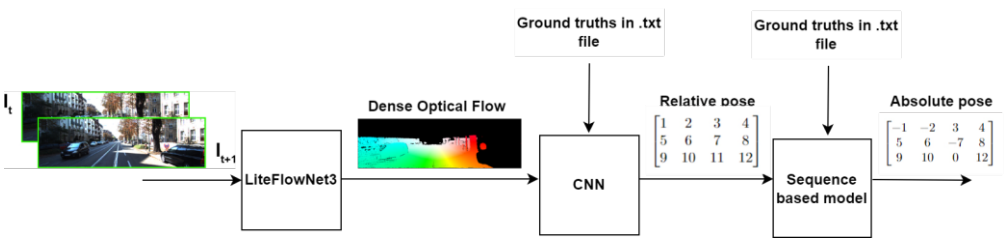

Figure 2. The proposed design

The proposed model consists of three modules - LiteFlowNet3 module, CNN module and a Sequence-based model. The description of each module is given in the following paragraphs. The proposed design is as shown in Figure 2.

FlowNet [7] was proposed in 2015 by Alexey Dosovitskiy, Eddy Ilg *et. al* as the first CNN-based approach for optical flow estimation. It infers flow fields from a pair of consecutive images. FlowNet was followed up by FlowNet2 [8] in 2016, LiteFlowNet [9] in 2018 and LiteFlowNet3 [10] in 2020. Each new variant improved upon the previous variants in terms of number of model parameters and accuracy of estimation.

In the proposed design, the LiteFlowNet3 module takes a pair of consecutive images I_t and I_{t+1} taken at time t and $t + 1$, respectively and produces the optical flow image corresponding to the pair. This optical flow image is given to the CNN module. The purpose of the CNN module is to learn the relative pose of the camera at time $t + 1$ with respect to the coordinate system in image I_{t+1}. The ground truth relative pose values will be fed to the CNN in a *.txt* file. The architecture will consist of several convolutional layers, followed by a flattening layer and finally some fully-connected layers. The relative pose values estimated by the CNN are forwarded to a sequence-based model which learns and outputs the absolute pose of the camera at time $t+1$ with respect to the coordinate system at the origin. It is known that sequence-based models are better at learning relationships that are spread across time, which is exactly the case at hand. The ground truth absolute pose values will be fed to the CNN in a *.txt* file.

3.1 Datase usedt

The dataset to be used for this work is the KITTI suite [11] created by Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago. It consists of sequences of images taken in different settings from a moving car. The ground truths are available for various Computer Vision related research areas like Depth Estimation, Semantic Segmentation, Optical Flow, Visual Odometry etc.

The pose of a camera is represented as a 3×4 (12 entries) transformation matrix, which determines the changes along all the 6 degrees of freedom between two instances of time t and $t + 1$. Relative pose means the changes are calculated with respect to the previous image in the sequence. Absolute pose means the changes are calculated with respect to the origin. In the Visual Odometry dataset provided by the KITTI suite, the ground truths are stored in a *.txt* file as an $N \times 12$ table, where N is the number of frames in the sequence. The i^{th} row denotes the row major representation of the i^{th} absolute pose. The 3×4 transformation matrices take a point in the i^{th} coordinate system (coordinate system of the i^{th} frame) and project it into the first coordinate system (coordinate system of the first frame).

3.2 Pre-processing

The dataset for this work was acquired from the KITTI benchmark suite which contains separate datasets with ground truths for Optical Flow Estimation and Visual Odometry, amounting to a total size of around 66 GB. The ground truths are stored in *.txt* files. For Visual Odometry, the ground truths provided are only absolute pose values. However, relative pose values are also necessary as per the proposed design. So, as a pre-processing step, relative pose values were generated from the absolute pose values.

3.3 Implementation

To start implementing the proposed design, we used Caffe and PyTorch libraries. An implementation of LiteFlowNet3 in PyTorch was re-purposed to implement the first module of the proposed design. Auxiliary code was added to allow the module to fully utilize the GPU runtime environment of Google Colab. Using this module, optical flow estimates were generated

for every pair of images in the sequences 00 through to 05 in the KITTI dataset. Furthermore, the optical flow estimates were converted into color-coded flow images.

For the next module, a CNN was implemented in PyTorch that takes as input the color-coded flow images and regresses the relative camera pose. The CNN consists of 7 convolution layers. Max pooling is applied after every layer, excluding the last one, to reduce the size of the feature map. A Rectified Linear Unit (ReLU) was applied as the activation function. The Mean Squared Error (MSE) loss was used to minimize the euclidean distance between the ground-truth pose and the prediction.

Layer	Kernel Size	Strides	Channels
Conv2D	3	1	16
MaxPool		2	
Conv2D	3	1	32
MaxPool		2	
Conv2D	3	1	64
MaxPool		2	
Conv2D	3	1	128
MaxPool		2	
Conv2D	3	1	128
MaxPool		2	
Conv2D	3	1	256
MaxPool		2	
Conv2D	3	1	256

Table 1. Model Summary of CNN

In the third and final module, a Vision Transformer (ViT) was implemented. As mentioned early, ViT is an adaptation of the Transformer architecture [12] for sequence-based tasks in the domain of Computer Vision. The ViT architecture diagram is shown in Figure 3.

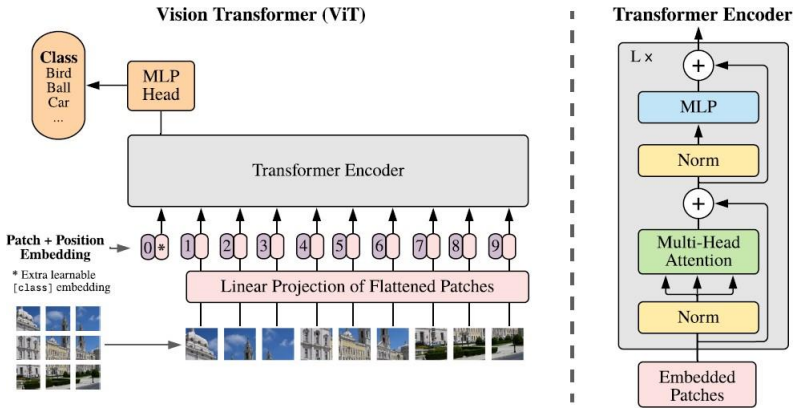

Figure 3. The Vision Transformer (ViT) architecture [5]

For each image *I* in a sequence in the dataset, the final layer of the CNN module produces a feature map consisting of 256 channels. These channels are considered as the *image patches*

Model	AEE
FlowNetS [7]	9.1
HD^3 [13]	1.4
IRR-PWC [14]	1.6
SelFlow [15]	1.5
Proposed	**1.3**

Table 2. AEE results of optical flow estimation module compared with state-of-the-art methods.

corresponding to the image *I*. The ViT module takes a sequence of image patches as input and outputs the absolute pose (the 12 elements of a 3x4 transformation matrix in row-major form) for each image *I* in a sequence in the dataset.

The ViT implementation consists of a Patch Embedding layer, followed by 7 Transformer Encoder layers, followed by a Fully-Connected layer with 12 outputs. All the layers are defined as classes in Python. The Patch Embedding layer learns the positional embedding (a learnable vector) of each image patch. Each image patch is flattened and is concatenated with its learned positional embedding and forwarded to the first Transformer Encoder layer. Finally, the output prediction is obtained from the outputs of the fully-connected layer. Each Transformer Encoder layer consists of two sub-layers: a Multi-head Attention layer and a Multi-layer Perceptron. Both these sub-layers are implemented using library functions available in PyTorch.

4 Results and Discussion

The network was trained in Google Colab, which provides a single core hyper threaded Intel Xeon Processor at 2.3Ghz and a 12GB NVIDIA Tesla K80 GPU.

The LiteFlowNet3 module was used to generate the optical flow color-coded images for sequences 00 through to 05, each consisting of around 4500 images. The accuracy of the generated optical flow estimates is evaluated using Average End Point Error metric. The Average End Point Error (AEE) is defined as the Euclidean distance between the estimated optical flow vector and the ground truth optical flow vector, averaged over all pixels. The AEE results of the optical flow estimation module are summarized and compared with similar state-of-the-art works in Table 2.

The majority of optical flow CNNs are 2-frame methods and use the same datasets for training. However HD^3 is pre-trained on ImageNet. SelFlow uses multi-view extensions of KITTI for self-supervised training and also uses more than two frames to boost the flow accuracy. Even so, the proposed method clearly outperforms the state-of-the-art optical flow CNN models, as can be observed from Table 2. The sequences 00, 02 and 04 were used as training set. The model was trained end-to-end for 100 epochs, with a batch size of 64. While training, Adam optimizer was used with a learning rate of 0.001. The model is evaluated on sequences 01, 03 and 05 to realize the generalization of our network. The evaluation is based on the KITTI odometry benchmark. Translation and rotation errors are calculated using the averaged root-mean-squared error metric.

The results of this work is compared against MagicVO [16] - a deep learning approach which has been used as a baseline for many related works, recently ([3], [17], [4], [18]). The results are obtained using the evaluation script and is summarized in Table 3. The implementation of the proposed model comes close to the baseline MagicVO on unknown environments. Although straight trajectories show minimal variation from the ground truth, errors

Sequence	MagicVO		Proposed	
	t (%)	r (°)	t (%)	r (°)
01	4.95	2.44	4.85	2.54
03	1.63	2.25	2.89	1.22
05	2.61	1.08	2.56	2.15

Table 3. Comparison of results with the baseline, MagicVO. *t* represents average translational drift (in percentage) and *r* represents average rotational drift (in degrees).

around turns and corners can be expected to be high. Table 3 indicates the errors are maximum in sequence 01. This may be due to the fact that the trajectory followed by the car in sequence 01 consists of a lot of turns and loop closures.

The developed model has achieved a promising accuracy of 82%. The translation and rotation errors for all possible sub-sequences of length 100 to 800 meters have been evaluated. As can be observed from the plots, there is a peak in the error when the path length is around 300-400 metre mark. The peak appears sharply in translation error compared to rotation error. Another point worth noting is that the error rate has almost stagnated towards the latter half of the sequences, that is around 500-800 metres. This could be interpreted as the model producing more accurate results in longer sequences than shorter ones.

In contrast to the translation/rotation error versus path length plots, when the errors are compared to the vehicle speed, the errors steadily increase as the vehicle attains greater speed. This is most likely due to the fact that there is minimal data in each sequence where the vehicle is traveling that quickly. With fewer samples of fast driving, there is more significance to the amount of error that is present when the car does achieve those speeds. Augmenting the network with a Vision Transformer has yielded positive results. One of the scenarios where many of the deep learning based Visual Odometry models struggle is when there are abrupt rotational drifts. Such rotational drifts are frequent in sequence 03. The proposed model was able to outperform the baseline in sequence 03 in terms of rotational errors, because it was able to extract more contextual information using the ViT, which is a sequence-based architecture.

5 Conclusion

Visual Odometry (VO) is the process of estimating the motion of an agent (e.g., autonomous vehicle, human, robot etc.) using only the images taken by cameras attached to it. Among other applications, VO can be used in autonomous robots like pathfinders and rovers that perform extraterrestrial surface explorations.

Deep Learning approaches have shown tremendous efficiency and accuracy in tasks that require high degree of adaptability and scalability. The task of self-motion estimation of an autonomous agent using VO can be posed as a Deep Learning problem. In this work, a Deep Learning approach to perform VO tasks for space robotic applications has been proposed. The design consists of three modules - a optical flow estimation module, a relative pose regressor module and a sequence-based model to infer temporal dependencies among relative poses.

References

[1] D. Scaramuzza, F. Fraundorfer, IEEE Robotics Automation Magazine **18**, 80 (2011)

[2] A. Howard, *Real-time stereo visual odometry for autonomous ground vehicles*, in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2008), pp. 3946–3952

[3] T. Pandey, D. Pena, J. Byrne, D. Moloney, Sensors **21** (2021)

[4] P. Muller, A. Savakis, *Flowdometry: An Optical Flow and Deep Learning Based Approach to Visual Odometry*, in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2017), pp. 624–631

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly et al., CoRR **abs/2010.11929** (2020), 2010. 11929

[6] D. Warren, E. Strelow, *Electronic Spatial Sensing for the Blind: Contributions from Perception, Rehabilitation, and Computer Vision*, Nato Science Series E: (Springer Netherlands, 1985), ISBN 9789024732388, https://books. google. co. in/books?id=-I_Hazgqx8QC

[7] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox, CoRR **abs/1504.06852** (2015), 1504. 06852

[8] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, T. Brox, CoRR **abs/1612.01925** (2016), 1612. 01925

[9] T. Hui, X. Tang, C.C. Loy, CoRR **abs/1805.07036** (2018), 1805. 07036

[10] T. Hui, C.C. Loy, CoRR **abs/2007.09319** (2020), 2007. 09319

[11] M. Menze, A. Geiger, *Object Scene Flow for Autonomous Vehicles*, in *Conference on Computer Vision and Pattern Recognition (CVPR)* (2015)

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, *Attention is All You Need*, in *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Curran Associates Inc., Red Hook, NY, USA, 2017), NIPS'17, p. 6000–6010, ISBN 9781510860964

[13] Z. Yin, T. Darrell, F. Yu, CoRR **abs/1812.06264** (2018), 1812. 06264

[14] J. Hur, S. Roth, CoRR **abs/1904.05290** (2019), 1904. 05290

[15] P. Liu, M.R. Lyu, I. King, J. Xu, CoRR **abs/1904.09117** (2019), 1904. 09117

[16] J. Jiao, J. Jiao, Y. Mo, W. Liu, Z. Deng, CoRR **abs/1811.10964** (2018), 1811. 10964

[17] Y. Almalioglu, M.R.U. Saputra, P.P. De Gusmao, A. Markham, N. Trigoni, *GANVO: Unsupervised deep monocular visual odometry and depth estimation with generative adversarial networks*, in *2019 International conference on robotics and automation (ICRA)* (IEEE, 2019), pp. 5474–5480

[18] S. Rao, *SuperVO: A Monocular Visual Odometry based on Learned Feature Matching with GNN*, in *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)* (2021), pp. 18–26

[19] Y. Lu, G. Lu, *Deep Unsupervised Learning for Simultaneous Visual Odometry and Depth Estimation*, in *2019 IEEE International Conference on Image Processing (ICIP)* (2019), pp. 2571–2575