doi:10.20944/preprints202209.0276.v1

Article

Towards Interpretable Camera and LiDAR Data Fusion For Unmanned Autonomous Vehicles Localisation

Haileleol Tibebu*, Varuna De-Silva, Corentin Artaud, Rafael Pina and Xiyu Shi

Institute of Digital Technologies, Loughborough University London, 3 Lesney Avenue, London E20 3BS, UK;
V.D.De-Silva@lboro.ac.uk (V.D.S.); c.artaud2@lboro.ac.uk (C.A); r.m.pina@lboro.ac.uk (R.P), x.shi@lboro.ac.uk (X.S)
* Correspondence: H.tibebu@lboro.ac.uk

Abstract: Recent deep learning frameworks draw a strong research interest in the application of ego-motion estimation as they demonstrate a superior result compared to geometric approaches. However, due to the lack of multimodal datasets, most of these studies primarily focused on a single sensor-based estimation. To overcome this challenge, we collect a unique multimodal dataset named LboroAV2, using multiple sensors including camera, Light Detecting And Ranging (LiDAR), ultrasound, e-compass and rotary encoder. We also propose an end-to-end deep learning architecture for fusion of RGB images and LiDAR laser scan data for odometry application. The proposed method contains a convolutional encoder, a compressed representation and a recurrent neural network. Besides feature extraction and outlier rejection, the convolutional encoder produces a compressed representation which is used to visualise the network's learning process and to pass useful sequential information. The recurrent neural network uses this compressed sequential data to learn the relation between consecutive time steps. We use the LboroAV2 and KITTI VO datasets to experiment and evaluate our results. In addition to visualising the network's learning process, our approach gives superior results compared to other similar methods. The code for the proposed architecture will be released in GitHub and accessible publicly.

Keywords: Sensor fusion; Camera and LiDAR fusion; Odometry; Explainable AI

1. Introduction

Localisation is one of the critical decisions for Autonomous Ground Vehicles (AGV) to operate safely. However, a reliable GPS signal is not always available in many scenarios, which makes localisation and orientation estimation of AVGs difficult, even impossible sometimes. A fully autonomous system requires dependable navigation capability, which works in all environments, including where a strong GPS signal is available and, in a condition, where a GPS signal is unreliable. Therefore, a mobile platform's self-contained pose and orientation estimation are critical as they are fundamental steps to get vital navigation information for a mobile robot and autonomous vehicles. Autonomous systems collect essential information about their environment using onboard sensors, including cameras, LiDAR, Inertial Measuring Unit (IMU), radar and digital compasses. This information is necessary for the autonomous system to track past trajectories, acquire current location, and plan future movements. Odometry uses data from motion sensors to estimate the change in the relative position of a mobile robot over time.

Camera and LiDAR sensors are by far the two dominants chosen by the robotic community to tackle an odometry problem. Both sensors have their advantages and disadvantages. LiDAR has a superb capability to obtain different features of the environment. LiDAR is also not affected by different lighting conditions. Most of all, LiDAR gets more accurate range measurements than a camera. However, reflection heavily influences the LiDAR data, and unless using additional algorithms, LiDAR cannot detect glass [1]. Besides, the LiDAR data are very sparsely distributed and have a limited visibility range. LiDAR also collects too much data, which requires high computational power for processing. It is also more expensive than a camera.

Cameras give dense and rich data, making them ideal for obstacle avoidance problems in AGVs. Camera is not affected by the presence of glass to the same degree as LiDAR does, and also does not have a limited visibility range. However, unless it is a 360^0 camera, it will have a narrow visibility angle resulting in blind spots. Apart from RGB-D cameras, standard cameras don't measure distance. Cameras also struggle to work in adverse weather conditions and are also severely impacted by different lighting conditions.

Other odometry methods are based on the fusion of data from multiple sensors [2-6]. Feature-based algorithms use gematrically steady features and track them across consecutive frames [6,7]. The motion of the sensor is then tracked by estimating the minimised reprojection error across pairs of frames. Although this method is robust, it mostly fails in an environment which doesn't generate proper-fit features to be extracted. The other method is called a direct method [8] which traced the intensity of pixels across frames and estimated the camera's motion from photometric error.

Current research directed their effort towards deep learning-based approaches. This method outperforms conventional approaches in related domains like object detection and classification [9,10]. It also yields better performance in odometry applications. However, it needs to be investigated further as it is still a new research area.

Research in [11,12] stated that the fusion method performs better odometry than a single sensor approach. Both papers use CNN and LSTM network architecture. While [11] addresses the problem as a classification problem, [12] deals with this problem as a regression problem. Both methods yield better performance than conventional approaches. In this research, we collected a unique multimodal dataset – LboroAV2, from multi-sensors, including a camera, LiDAR, electronic compass, rotary encoder, and ultrasonic sensors, suitable for research related to localisation and mapping. The data in LboroAV2 is acquired using our bespoke AGV while driving in multiple paths and sequences with ground truth data. We proposed an efficient data-driven neural network architecture combining LiDAR and camera for odometer applications. The proposed architecture is tested using KITTI dataset and LboroAV2.

The rest of the paper is organised as follows. A review of the literature is presented in the related work section. The methodology section describes details about the research methods and the developed localisation algorithms, followed by the results and discussion section. Lastly, we summarise our findings in the conclusion section.

2. Related work

Ego-motion, aka pose estimation, is one of the primary challenges in simultaneous localisation and mapping (SLAM). Pose estimation plays a significant role in various applications such as autonomous cars [2], Unmanned air vehicles (UAV) [3], robotic hands [4] and 3D reconstruction [5]. Pose estimation is the key to tracking position over time. There are two approaches to odometer: traditional methods (geometry model) and learning methods. Traditional methods have shown superb performance, especially in reducing noise. However, they lack the adaptability to all challenging scenarios. Moreover, conventional methods depend on hand-craft modules. To overcome the shortcomings of the traditional approach, researchers are proposing learning methods with promising results [6]–[9].

Most previous research based on learning methods uses data from either range measurement or visual sensors. Only a few research uses fusion of LiDAR and camera for odometry application.

2.1 Visual odometry

There are two different approaches to achieve this; feature-based and direct methods [10]. These approaches target estimating the pose of a robot based on images. Feature-based methods follow the standard pipeline, including feature extraction, matching, estimation of motion, and local and/or global optimisations [11][12]. Assuming that scenes are stationery, the direct method aims to minimise a photometric error by considering all the pixels in the image [13][14]. The assumption in a direct method that scenes are stationery is not conclusive enough to fully satisfies. Hence most direct methods have lower performance than feature-based methods.

Following the advancement of deep learning, researchers are exploring deep learning-based approaches for odometry application using visual data, aka visual odometry (VO) [6-9], [15], [16]. This technique requires a ground-truth-labelled data. Some researchers use a pretrained FlownetS model [17] by remodelling it for the prediction of relative poses [18][19]. The FlownetS model using a single architecture, given two consecutive RGB images, executes feature extraction and matching for pose estimation. This technique assumes that features learned from the two successive images for optical flow estimation can also be used to predict the relative pose between the images.

Another approach uses optical flow as input and reverts the pose based on learned information from the past [19][20][6]. Deep learning approaches are not only learning visual signs, but they are also learning camera intrinsic features.

Wang et al. [21] argued that while CNNs are suitable for capturing useful learning features, they are not satisfiable enough to extract motion dynamics. Hence Wang suggested that CNN models are not good enough for Visual Odometry (VO). Therefore, Wang proposed sequential based modelling with Deep Recurrent Convolutional Neural Networks (RCNN). While the FlownetS architecture is augmented with LSTM which outputs pose estimation for each time step, LSTM is not suitable for high dimensional data. [22] proposes a stacked autoencoder to capture optical flow and predict relative poses of the camera by forming a multi-object loss function to minimise inaccuracy. This method performs better than traditional monocular approaches like [12]. Nevertheless, the multi-objective may not share favourable features for both reconstruction and pose estimation.

2.2 LiDAR odometry

LiDAR odometry is the process of estimating the relative pose of two 3D point cloud frames [10]. The LiDAR apparatus is known for its accurate depth information. Nevertheless, using LiDAR data for odometry is challenging due to the point cloud data's sparsely nature. Most LiDAR-based odometry works are based on traditional pipelines and achieved promising performance [23] [24]. Martin et al. [25] proposes a method to overcome the shortcoming of LiDAR point clouds sparsity by grouping the LiDAR points into a polar bin and generating a line segment for each bin. Although it produces a better result than traditional methods, it is computationally expensive and cannot be applied in real-time.

Deep learning method has also been used for the problem of LiDAR-based odometry. However, this method is still very challenging. For instance, [26] and [27] uses CNN to perform LiDAR based odometry. In this method, the original data is transformed into a dense matrix with three channels. While [26] only estimates translational data, [27] encodes the point cloud into data matrices to feed into the network, and direct data processing is more practical. [28] uses a deep neural network by replacing each component of the traditional pipeline. Other methods, such as DeepLO [29], LO-Net [27], and DeepVCP [30] design different types of end-to-end deep learning-based frameworks and have been conducted using the supervised learning method.

2.3 Fusion of Visual and LiDAR odometry

Research conducted by fusing visual and LiDAR data takes advantage of both the LiDAR and visual information to predict the translation and rotation of a mobile robot. Again, most of these methods are conducted using a traditional approach. The V-LOAM [31] combines visual odometry and LiDAR odometry for better model optimisation using a conventional feature-based pipeline. The LIMO [32] proposes a feature-based visual odometry, which takes scale estimation from LiDAR data to attain pose estimation.

Recent emerging research uses learning techniques to fuse LiDAR and visual data for odometry purposes. [33] uses high-resolution images to improve 3D point clouds using deep learning method. [34] propose self-supervised visual odometry for estimation of scale aware poses. However, they use only images as input to the network. The LiDAR data is added as a supplement. [10] propose a self-supervised LiDAR and visual odometry which takes monocular images and depth maps obtained from the LiDAR point clouds as network input to estimate pose.

This research proposes a deep learning architecture for real-time odometer applications which can also be used to explain the learned features and process at the compressed representation mode. To the best of our knowledge, this is the first architecture that fuses LiDAR and camera data in the compressed representation node of the mono-modal encoders. We propose a new network architecture that provides a better result compared to similar approaches. The network architecture can be used to tackle other similar type of problems. We also collected a unique dataset named LboroAV2. The LboroAV2 data can be used for different localisation problems. The data is acquired in both structured and non-structured outdoor environments. We use LboroAV2 and KITTI dataset for training and testing our model.

3. Method

This section first describes the LboroAV2 and KITTI VO datasets, followed by a detailed explanation of the proposed approaches. The proposed method combines the Convolutional encoder with LSTM model to estimate translation and rotation between the fusion of two consecutive LiDARs scan and camera images.

3.1. Dataset

A review of existing datasets that are potentially suitable for this research has been conducted. Except for a few, the majority of exiting datasets are mono modal. Amongst a very few who have a multimodal dataset, none of the datasets has both structured and unstructured environments. Hence, this research investigates two datasets: the publicly available KITTI [35] and LboroAV2. The KITTI dataset contains 20 sequences of LiDAR and a camera; however, the ground truth data is only available for 10 sequences.

The LboroAV2 multimodal dataset is collected specifically for this project. An AGV [36] is used to collect the dataset. The AGV was autonomously driven throughout the data collection period, with minimal human interaction, on structured and unstructured roads on the privately-owned Here East compound in Queen Elizabeth Olympic Park, London. Multiple sensors are embedded in the AVG platform, including a Camera, LiDAR, ultrasound, rotary encoder, and electronic compass.

The VLP-16 is a low powered compact optical sensor with a useable range of up to 100m. It utilises 16 pairs of emitter detectors which measure a total of 300,000 data points per scan.

The data were logged using two ways; directly from the sensors to a laptop and using data logger mounted on the AVG. There were 864K (at 60fps) frames captured by the wide-angle camera camera, 432K frames (at 30fps) captured by the Ricoh Theta V 360° camera. The rotary encoder and the ultrasound sensor captured 62k (5fps) scans respectively. The rotary encoder and electronics compass captured 144K (10fps) data sets each. Given that each sensor has its own limitations, the mixture offered by multimodality has excellent potential, to sum up a positive contribution to the intended capability of any machine. Fig 1. shows the AGV used for data collection and the onboard sensor's locations.

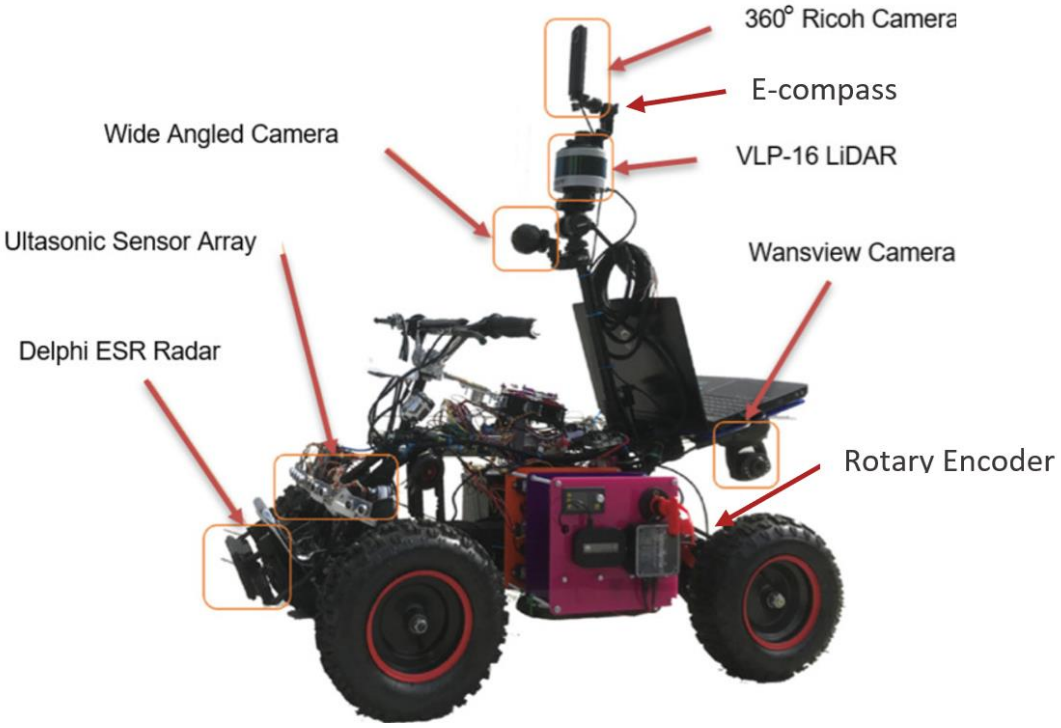


Fig. 1 The Unman ground vehicle used for LboroAV2 data collection [36]

The LboroAV2 dataset were collected in four separate routes. The total distance of data collection covers over 1.2 km. The dataset contains multiple outdoor environments at different times of the day. The data collection period cover 11 months to include a wide range of class features, including, pedestrians, cyclist, and vehicle traffic in many different environmental conditions.

The dataset considered in this research is selected based on the assortment of foreground and background objects, and the overall scene layout. The sensor data which are not selected for this research are compiled for future work.

Although the AVG platform's operation speed can reach up to 22km/h, its speed during the data collection period is limited to 4km/h due to the License and Permit approved by the HereEast management.

Sensor parts that are subject to a build-up of contaminants, such as lenses, were sufficiently maintained throughout the data collection period. A unique label for each sequence was added to the captured data—labels for accessible grouped data collected in the same sequence. Care was taken to ensure that all data captured were meticulously archived, labelled, and stored securely following data protection acts.

3.2 Data encoding

The data coming from LiDAR and camera sensors are required to be pre-processed before feeding to the proposed network architecture. The original image data from the KITTI and LboroAV2 datasets is transformed into a size of 416x128 pixels. The primary aim of compressing the datasets is to reduce the computational time of the training while not losing valuable features. The LiDAR data is encoded in two methods and each method is tested in the experiment. In the first method, we encode the LiDAR dataset based on [37]. The raw LiDAR data is initially binned with a resolution of 0.10. The 3D LiDAR data is then encoded in to a 1D vector. We use the average value of point clouds that belongs to the same bins as the encoded value, as defined in (1). The final vector has 3601 elements, which stores each bin's depth information.

$$\mathcal{P}_i = \frac{1}{N} \sum_{k=0}^{N} \alpha_k \, , \forall i \in [0, 3600] \tag{1}$$

where $\mathcal{P}_i$ represents the encoded value of ith bin, α is raw depth information and N is the total number of LiDAR data points in a single scan.

Once each image and LiDAR data is encoded as described, each sequential LiDAR and camera data scan is concatenated. The images are concatenated channel-wise to give a sequence of 6x128x416. The LiDAR data become 2x3601 size after the concatenation of two consecutive scans. The concatenated LiDAR shape enables the network to pass a convolutional layer for feature extraction.

The second LiDAR data encoding method is based on the approaches in [38]. The 3D LiDAR data is projected to a 2D image as seen in Fig 2. The projected 2D image contains the local geometry of the 3D voxel space. This feature helps the model to learn the translation and rotation while reducing the input size of the model. We limit the projection of the LiDAR to the front side 180^0 field of view.

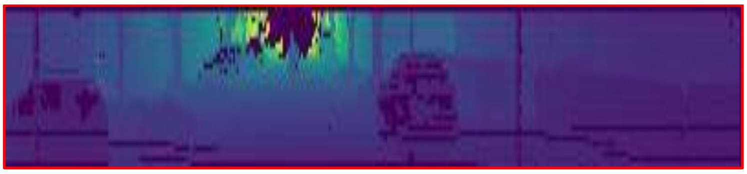


Fig. 2 LiDAR point clouds projected into a panoramic image

3.3 Deep learning architecture

Deep learning-based algorithms have become a common practice for image classification due to the availability of large-scale datasets. Applications such as odometry require a high volume of data suitable for learning geometric features representing the surrounding environments. Unlike traditional methods, deep learning approaches do not require manual data calibration. Although current deep learning approaches outperform traditional methods, their architectural designs are not good enough to grasp all the useful underlining features of the input data. Moreover, it is impossible to explain the underline feature extraction process between neurons as the machine learning process is thought as a black box problem. To overcome this limitation, we propose a new design of deep learning architecture, as shown in Fig.3, that can better extract features representing image and LiDAR data for odometer applications. The architecture contains two parts, an encoder that includes compressed representation node and a sequential modelling with RNN. The compressed representation node space of this model can also be used to gain insight into the deep learning network.

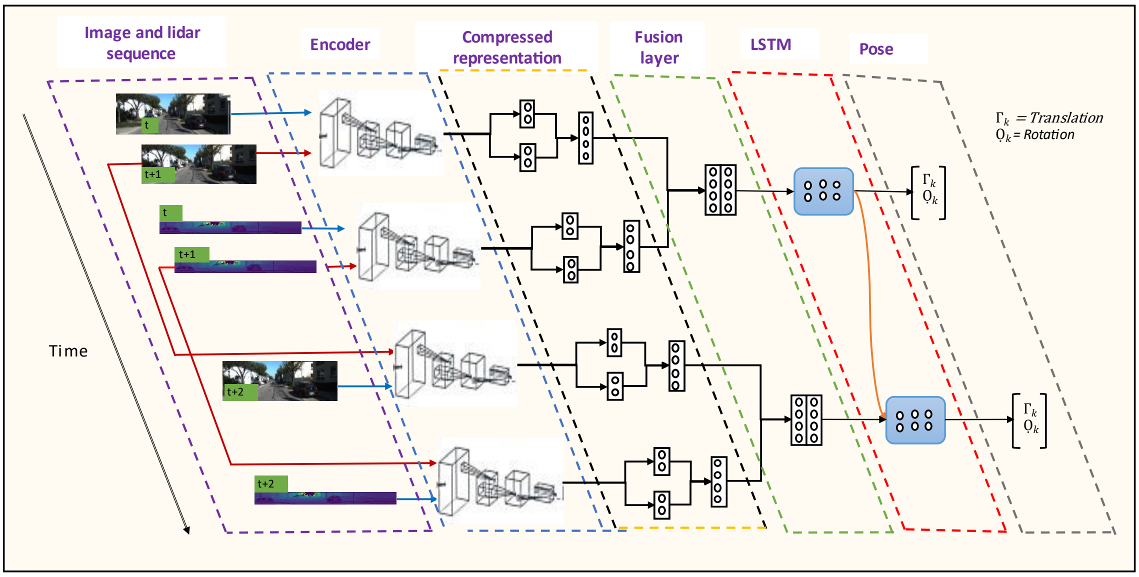


Fig. 3 Architecture of the proposed deep learning method

Three different architectures have been experimented with in the research. Figure 4 shows the architectures for fusion of camera and LiDAR point cloud data and fusion of camera and LiDAR data projected in 2D images. Figure 5 shows the camera only architecture. The architectural details of each model are discussed below

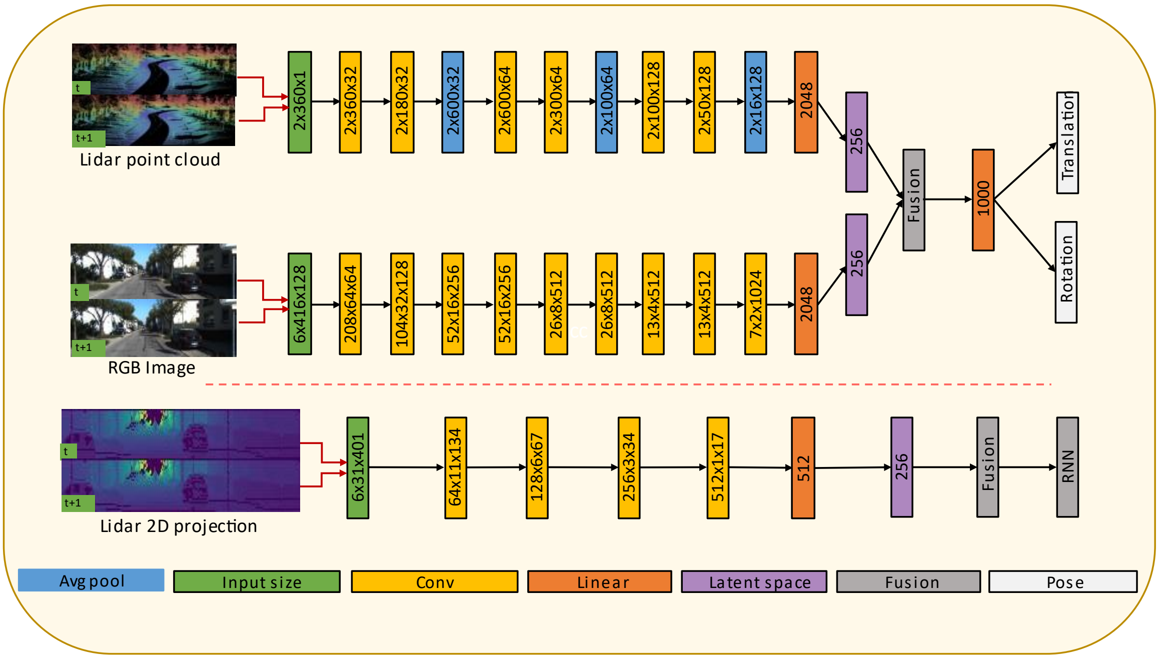


Fig. 4 Architectures for fusion of camera and LiDAR point cloud data (upper part), and fusion of camera and LiDAR data projected in 2D images (lower part)

3.3.1 Encoder

Popular deep learning architectures like Alexnet [39] and VGG [40] are designed to perform well for tasks like object detection and classification, primarily dependent on objects' appearance in the image. However, training data for odometry applications requires a deeper extraction of geometric features from consecutive scans. We use different feature extraction architectures for the camera and LiDAR, which work based on the same principle, referred to encode hereafter. The detail of the encoder for each modality is described below.

3.3.2 Camera feature extraction:

We modify the Flownets [17] optical flow estimation method, which uses two consecutive RGB images to extract useful features that help estimate the pixel flow between two images. The architecture is composed of 9 convolutional layers. Except for the last layer, all the convolutional layers are followed by rectified linear unit (ReLU) activation function. We gradually decrease the kernel size from 7x7 to 3x3. We use a stride of 2 for Conv 1-3, Conv 5, Conv 7, and Conv 9; for the rest, we use a stride of 1. We also apply padding that is decreased gradually from 3 to 1. At the end of the 9th convolution, the extracted features are further reduced to a vector of 256-element, known as compressed representation node, as shown in Figure 4. This compressed representation node is a systematically condensed representation of the features in the original input data.

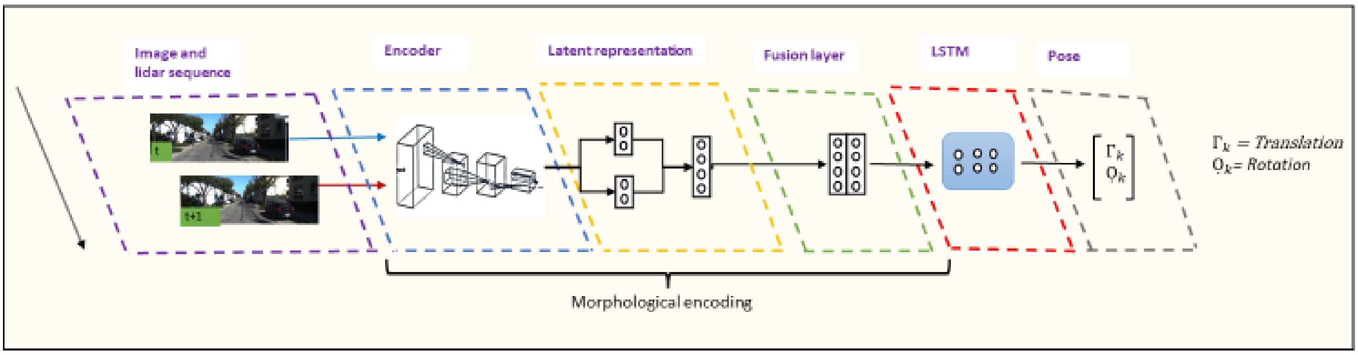


Fig. 5 Proposed mono camera odometry architecture

3.3.3 Point cloud feature extraction

As described in the section of data encoding process (Section 3.2), the concatenated two LiDAR point cloud data scans have a shape of 2x3061. This data is then sequentially fed to a 1D convolution layer. We use six 1D convolutional layers following the method in [41]. Each convolutional layer is followed by a ReLU activation. Average polling is applied after each convolutional layer to further reduce the computational time. At the end of the last convolutional layer, the extracted features are further reduced to a vector of 256 elements, as shown in Fig 4, so that both the camera and the LiDAR feature extraction have the same compressed representation node shape.

3.3.4 LiDAR projected 2D image feature extraction

The projected LiDAR 2D image size is smaller than the RGB camera image. Hence, we reduced the LiDAR feature extractor architecture. The projected 2D image has a shape of 3x31x401. The final size of the channel-wise concatenated 2D projected LiDAR image is 6x31x401. The proposed LiDAR projected 2D image feature extraction architecture is presented in Fig 4.

3.3.5 LiDAR and camera fusion

Once the LiDAR and camera data have the same size compressed representation node through their feature extractions, the fusion is conducted by concatenating each modality's compressed representation nodes. The rotation and the translation have the same sized compressed representation node. The learned features in the compressed representation node have an effectively encoded representation and can enhance the efficacy of the sequential training method.

3.3.6 RNN sequential modelling

The first RNN based visual odometry was proposed by [21]. RNNs are a branch of deep networks suitable for understanding the core features in a sequential dataset or a sensor data stream. The LboroAV2 data from multiple modalities contains a useful sequential relationship between consecutive scans. Thus, using RNN model for odometer application is necessary to exploit those temporal relationships between successive scans. However, it's not pertinent to learn temporal relationships using the raw image and LiDAR point cloud data as these data are very high dimensional. Thus, in our architecture, we use the concatenated compressed representation of the LiDAR and camera data as input to the RNN.

Although the classic RNN can learn temporal information, it suffers from a vanishing gradient when the gradient passes over a long timestep. The Long Short-term Memory (LSTM) has been introduced to overcome this problem [42]. There are three gate systems in LSTM, an input gate, an output gate and a forget gate, represented as sigmoids (σ) or hyperbolic tangents (tanh) functions in Figure 6. These gates are used to decide which past information to be maintained and discarded when updating the current state. In the proposed method, we use a bi-directional LSTM, a construct of two LSTM. There is 1000 hidden state for each LSTM layer.

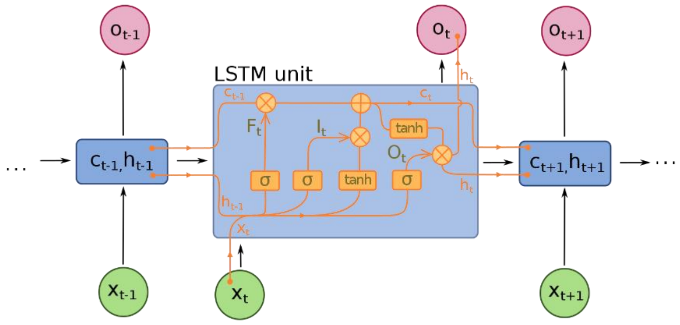


Fig. 6 The hidden state interactions in an LSTM model

3.3.7 Loss function

The proposed architecture computes the loss by finding the conditional probability of the pose, the translation Γ_k and rotation $\text{\d{O}}_k$, given a consecutive LiDAR L_d and camera C_a scan at time t, as shown in (02,03):

$$P\big(\,\Gamma_k \,\big|\, (L_d \wedge C_a)\big) = p(\,\Gamma_{k_{1,}} \;...\; \Gamma_{k_t} \,/\, (L_{d_{1,}} \;...\; L_{d_t}\,) \wedge\, (C_{1,} \;...\; C_{a_t})), \tag{02}$$

$$P\big(\,\text{\d{O}}_k \,\big|\, (L_d \wedge C_a)\big) = p(\,\text{\d{O}}_{k_{1,}} \;...\; \text{\d{O}}_{k_t} \,/\, (L_{d_{1,}} \;...\; L_{d_t}\,) \wedge\, (C_{1,} \;...\; C_{a_t})), \tag{03}$$

The optimal parameter (θ) of the network is derived from:

$$\theta^{\Gamma} = \Big(\, \underset{\theta}{argmax}\; p\,(\,\Gamma_k\,)_t \,\Big|\, (L_d \wedge C_a)_t\,\Big)\,, \tag{04}$$

$$\theta^{\text{\d{O}}} = \Big(\, \underset{\theta}{argmax}\; p\,(\,\text{\d{O}}_k\,)_t \,\Big|\, (L_d \wedge C_a)_t\,\Big)\,, \tag{05}$$

where θ^{Γ} and $\theta^{\text{\d{O}}}$ are the optimal parameters for translation and orientation respectively.

The loss function is calculated using Mean Square Error (MSE) by minimising the Euclidean distance between the ground truth pose (translation Γ_k and rotation $\text{\d{O}}_k$) and the predicted pose $(\acute{\Gamma}_k\,,\ \hat{\text{\d{O}}}_k)$, at a given time t, as shown in (06):

$$\theta^{*} = \underset{\theta}{argmin}\ \tfrac{1}{N}\ \textstyle\sum_{i=1}^{N} ||\, \acute{\Gamma}_k - \ \Gamma_k||_2^2 + \varkappa\, ||\, \hat{\text{\d{O}}}_k - \ \text{\d{O}}_k||_2^2\ , \tag{06}$$

where ‖·‖ is 2-norm, $\varkappa$ is a scale factor to balance the weight of rotation, and N is the number of samples. We use similar scale factor as [21].

4. Results

This section provides a detailed description of the proposed method of training and evaluation. As described in the methodology section, we implement three different models and evaluate their performance using publicly available KITTI dataset and our LboroAV2 dataset collected for SLAM-related research. We compare our results against single sensor-based and fusion of LiDAR and camera-based recent work.

4.1 Training

The LboroAV2 dataset and the KITTI VO/SLAM benchmark [35] are used to evaluate the proposed method. The KITTI VO/SLAM benchmark, which contains 22 images and LiDAR data sequences. Only 11 of the KITTI sequences (00-10) have ground truth. The other sequences (11-21) only contain raw sensory data. This benchmark is very challenging for VO algorithms because of its urban scenery with maximum speeds of up to 90km/h with numerous dynamic objects within each frame, all while being recorded at a relatively low frame rate (10 fps).

Because only the ground truth for sequences 00-10 are provided in the KITTI benchmark, the relatively long sequences 00, 01, 02, 06, 07, 08, and 10 from KITTI and sequences 03 and 04 from LboroAV2 are used for training, and the KITTI sequences 03, 04, 05, 09 and LboroAV2 sequences 01 and 02 are used for testing. To generate more samples for both raw images and LiDAR data, each trajectory is segmented into a random sequence length between the range of [5, 7] for training and length of 6 for testing. However, due to this randomness, duplicated segments are inevitable and are therefore removed if a generated segment is already presented within the augmented dataset. This transformation is applied on both datasets used.

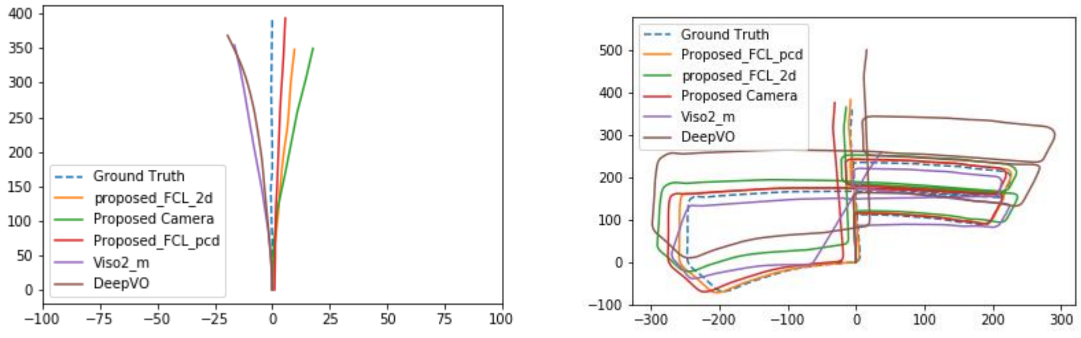
4.2 Experimental set up

The network is implemented in PyTorch and was trained using an NVIDIA GeForce 2080 T GPU. The optimiser Adagrad is used to train the network for up to 50 epochs with a learning rate of 0.0005. The encoder designed is inspired by the FlowNet model [17]. However, in contrast to using a pre-trained model, the model parameters in our experiment are trained from scratch. To prevent the model from overfitting, the Batch Normalization and Dropout layers are employed at every 2D convolutional layer. To reduce the size of the high-dimensional point-cloud data, the Average Pooling is employed at every 1D convolutional layer.

4.3 Experimental results and discussion

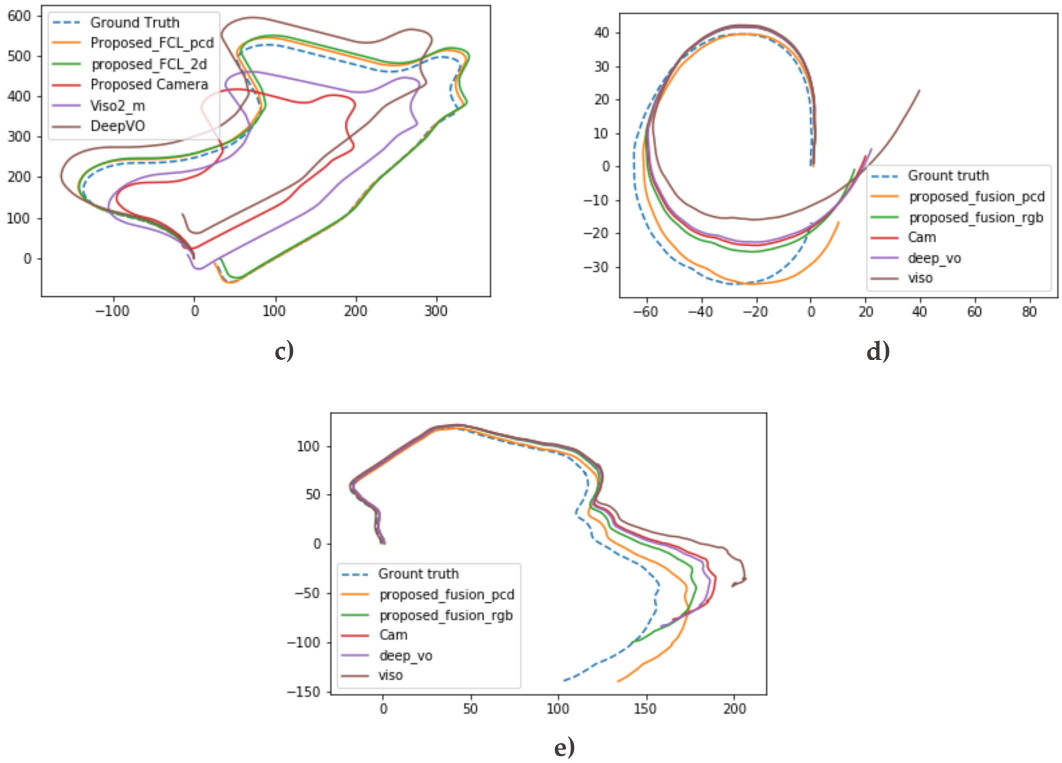
We experimented our approach in three different methods:

- RGB camera Image (denoted as Proposed Camera)
- Fusion of RGB camera image and LiDAR point cloud data (denoted as Proposed_FCL_pcd)
- Fusion of RGB camera image and projection of RGB LiDAR panoramic projection image (denoted as Proposed_FCL_2d).



a)

b)



c)

d)

e)

Fig. 7 Results of the proposed algorithms trajectory (measured in meter) on the testing sequence. a), b), and c) Results with KITTI dataset sequence 04,05 and 09, respectively; d) and e) Results with sequence 01 and 02 of LboroAV2 dataset.

Table 1. presents the Root Mean Square Error (RMSE) of the translation and rotation for all subsequent of length between 100 and 800 meters with a varying speed among each sequence. The proposed method's quantitative result is based on the KITTI SLAM/VO evaluation procedure.

The training and testing dataset contains a mixture of moving and static objects captured at 10fps with the car moving at different speeds. Our average rotation and translation error on the testing sequences while using the KITTI dataset shows that all the three proposed methods outperform Deep VO and Viso2_m algorithms.

The result shows that the rotation and translation in the LboroAV2 dataset is higher than that of KITTI. This is because the LboroAV2 dataset is collected in a lower speed less than 5kmh. It's also been evident in the results, as presented in Fig 7, that the fusion of the LiDAR and camera outperforms the single modality techniques in all sequences. This proves that the proposed network can effectively learn the ultimate way to fuse the extracted RGB camera and LiDAR feature at the compressed representation node. The result also confirms that the fused features contain relevant temporal information, which helps the RNN network to learn the progressive relationship between consecutive scans; hence the errors caused by drift decrease.

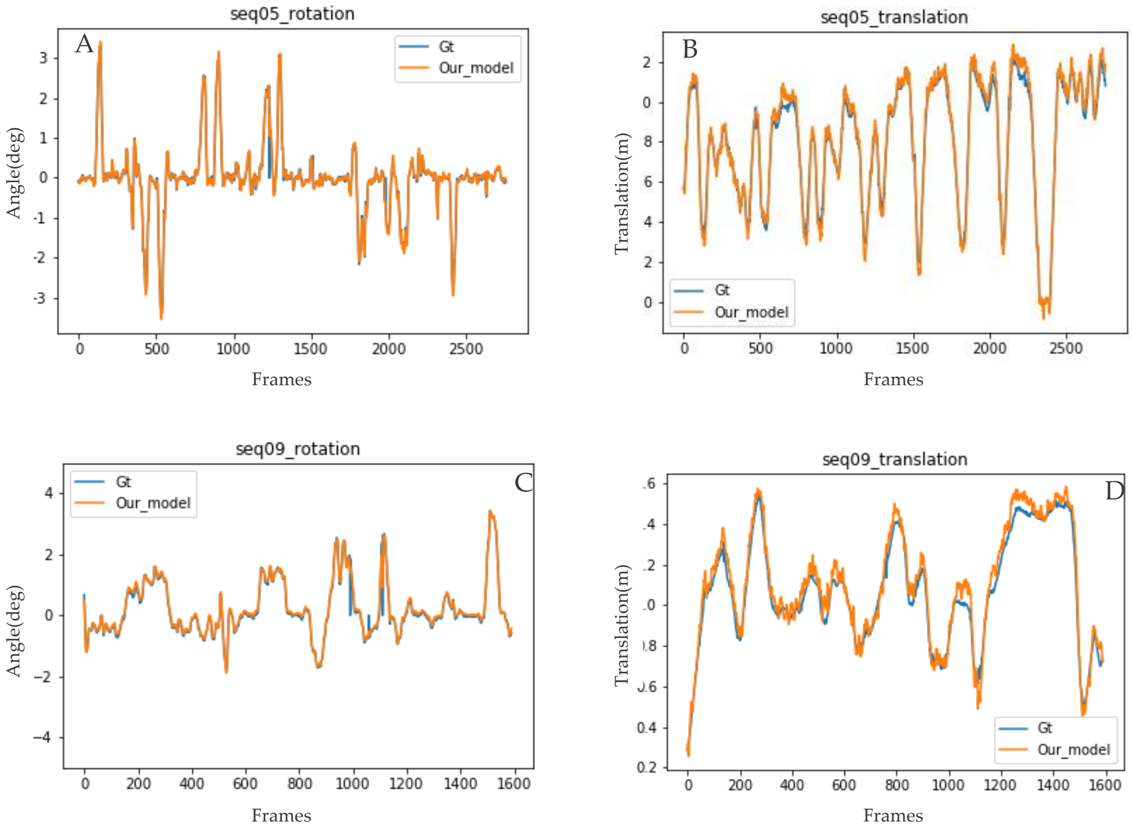


Fig. 8 Results of our proposed methods estimated translation and rotation plotted against the ground truth (Results from KITTI 05 (A & B) and 09 (C & D) is used as a sample).

It's also evident in Fig 8 that estimated trajectory errors are not evenly distributed throughout each sequence. Result shows that most translation and rotation occurred around turns. When the car approach turns, it reduces speed or comes to a stop. The amount of training data on KITTI while the car is driving lower than 20km/h and greater than 50km/h is minimal, errors are observed higher in this region. Major part of the translation error originates when the car is driven at a speed of more than 50km/h or when the translation between two consultive frames is more than 1 meter as shown in fig 8 (B & D).

4.4 Results evaluations

Table 1. Comparison of RMSE between the proposed method Vs the DeepVO and Viso2_M.

Dataset	*Sequence*	*DeepVO*		*VISO2_M*		*Our camera only*		*Our fusion using LiDAR 2D image & camera*		*Our Proposed (Fusion using LiDAR point cloud & camera)*	
		Γ_k	$Ọ_k$	Γ_k	$Ọ_k$	Γ_k	$Ọ_k$	Γ_k	$Ọ_k$	Γ_k	$Ọ_k$
KITTI	*04*	*7.19*	*6.97*	*4.69*	*4.49*	*2.33*	*3.25*	*2.23*	*2.33*	*2.01*	*2.11*
KITTI	*05*	*2.62*	*3.61*	*19.22*	*17.58*	*2.45*	*2.56*	*1.89*	*1.90*	*1.75*	*1.45*
KITTI	*09*	*8.11*	*8.83*	*41.56*	*32.99*	*9.34*	*10.70*	*1.77*	*1.83*	*1.70*	*1.78*
Mean KITTI		*5.97*	*6.47*	*21.82*	*18.35*	*4.70*	*5.50*	*1.96*	*2.02*	*1.82*	*1.178*
LboroAV2	*01*	*9.56*	*10.2*	*44.63*	*35.8*	*8.24*	*9.11*	*7.94*	*7.29*	*3.30*	*3.14*
LboroAV2	*02*	*9.86*	*10.41*	*38.21*	*42.11*	*10.53*	*10.61*	*3.69*	*3.27*	*3.43*	*3.62*
Mean_lboroAV2		*9.71*	*10.30*	*41.42*	*38.95*	*9.38*	*9.86*	*5.81*	*5.28*	*3.36*	*3.38*

Γ_k Average translation drift RMSE (%) on length of 100-800m, $Ọ_k$ average rotation RMSE drift (^{0}/m) on length of 100-800m

Furthermore, a significant amount of KITTI and LboroAV2 datasets is collected while the car is driven on a straight road; hence the number of corners and turns in training is limited. One of the contributions of the LboroAV2 it is richness in the slow-moving training dataset. While this dataset helps improve the model by providing a slow-moving dataset, the LboroAV2 has few corners. An extensive training dataset which represents all driving scenarios in the environments is required to increase the robustness of deep learning-based odometry algorithms. The images and LiDAR scans are captured at 10Hzin both datasets. Hence, there is a higher motion displacement between consecutive frames, especially when the car travels at a higher speed. This is an additional challenge for the model as there will be no features to extract, especially driving in open areas like highways.

Table 2. Comparison the proposed method and Deep_VO training and testing time on the KITTI dataset

Method	*Avg. Training time /epoch (minutes)*	*Avg. Testing time /epoch (minutes)*
*Deep_VO**	*4.25*	*0.52*
Proposed_camera	*1.51*	*0.28*
proposed FCL_2d	*2.18*	*0.34*
proposed FCL_pcd	*2.25*	*0.35*

* Since no official DeepVO publicly available code for benchmarking exists, we test our result by implementing the architecture as proposed in the [21] paper.

As presented in table 2, the proposed method yields a training and testing time reduction compared to the DeepVo architecture. This is due to the proposed method's robustness in extracting useful geometrical and temporal features to a size of 250-dimension compressed representation node. Hence, the trainable parameter passed to the RNN is smaller than similar methods.

4.5 Explainable representation

Hence our feature extractions are accompanied by a compressed representation node. One of the contributions of the proposed architecture is that it opens the door for odometer researchers to explore the compressed representation node. One of the major drawbacks of deep neural architecture is that the hidden neurons' learning process is unknown [43]–[45]. Visualising the compressed node during training time gives access to gain insight into the neural network's black box, which makes it possible to observe and analyse the learning process of the model.

Even though exploring the black box of deep learning architecture is not the focus of this paper, the authors are interested in expressing the proposed method's contribution to attaining explainable AI. We re-train the model using a 16-dimensional compressed representation to make the visualisation of the compressed space more expressive. Fig .9 displays the learning process of the proposed architecture through the training period. We took a sample of six images between the range of the first iteration to the last. The result shows that each of the 16 dimensions is learning different features. The model is learning how to differentiate feature spaces between each dimension whenever the learning time is increased.

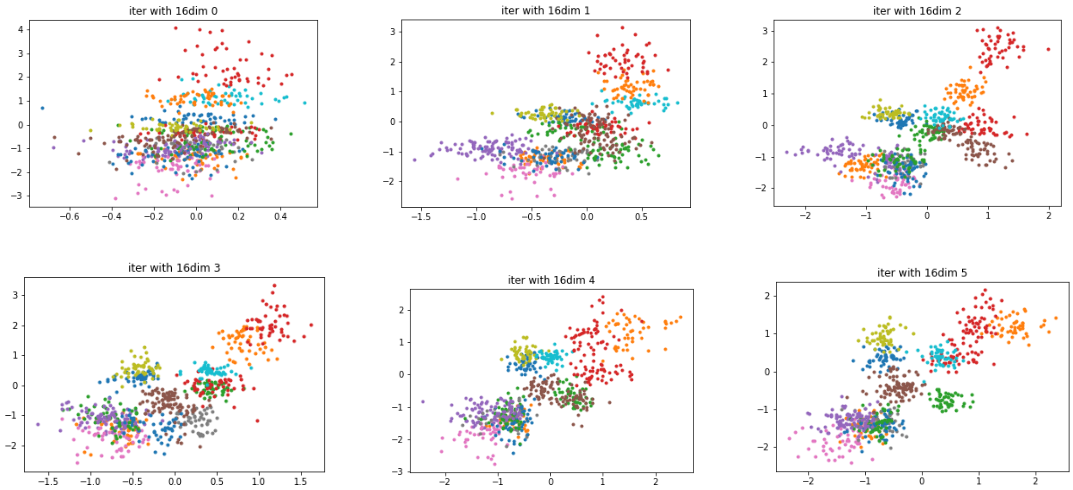


Fig.9 Visualisation of the compressed representation through the training cycle. This result was obtained by training the model with a 16-dimensional compressed representation to make the visualisation more expressive.

This paper aims to highlight the proposed architecture's potential to shed light on the hidden box of deep learning. Moreover, investigation of the compressed representation node will help to fine-tune the model by omitting redundant dimensions. Further research is also required to understand individual compressed representation node and their contribution towards increasing or decreasing translation rotation accuracy. A full investigation of the interpretation, explainability, and usage of compressed dimensions is the future research direction of the authors.

5. Conclusion

This study presents a deep learning-based fusion of LiDAR and camera architecture for odometry application. Our model consists of an encoder designed to extract optical flow between two frames, a compressed representation node of the input and the RNN model. Besides effectively downsampling the input features, the proposed approaches extracted useful geometrical information whiles it also learns to extract suitable features for the RNN to learn temporal information. Besides the model. We also contribute a uniquely reach dataset which can be used for a range of localisation research. We benchmark the proposed method on KITTI and LboroAV2 dataset, and our fusion approach significantly decreases translation and rotation error compared to both single modality and fusion-based algorithms. Our approach decreases substantially training time whilst using high-dimensional data from two different sources.

Besides better accuracy, the compressed representation node of the proposed methods opens the door for future research to explore the black box of deep learning architecture. Further examination is also essential to understand specific compressed representation node and their role in improving translation rotation accuracy.

Acknowledgments: This research was funded by the UK EPSRC grant number EP/T000783/1: MIMIc: Multimodal Imitation Learning in MultI-Agent Environments and grant number EP/S515140/1: Intuitive Learning: new paradigm in AI for decision making in intelligent mobility.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] H. Tibebu, J. Roche, V. De Silva, and A. Kondoz, "LiDAR-based glass detection for improved occupancy grid mapping," *Sensors*, 2021.

[2] G. Zhai, L. Liu, L. Zhang, Y. Liu, and Y. Jiang, "PoseConvGRU: A Monocular Approach for Visual Ego-motion Estimation by Learning," *Pattern Recognit.*, 2020.

[3] A. Briod, J. C. Zufferey, and D. Floreano, "A method for ego-motion estimation in micro-hovering platforms flying in very cluttered environments," *Auton. Robots*, 2016.

[4] P. Vicente, L. Jamone, and A. Bernardino, "Robotic Hand Pose Estimation Based on Stereo Vision and GPU-enabled Internal Graphical Simulation," *J. Intell. Robot. Syst. Theory Appl.*, 2016.

[5] J. Liu *et al.*, "Feature Boosting Network for 3D Pose Estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

[6] T. Pandey, D. Pena, J. Byrne, and D. Moloney, "Leveraging deep learning for visual odometry using optical flow," *Sensors (Switzerland)*, 2021.

[7] B. Teixeira, H. Silva, A. Matos, and E. Silva, "Deep Learning for Underwater Visual Odometry Estimation," *IEEE Access*, 2020.

[8] Q. Liu, H. Zhang, Y. Xu, and L. Wang, "Unsupervised deep learning-based RGB-D visual odometry," *Appl. Sci.*, 2020.

[9] Q. Liu, R. Li, H. Hu, and D. Gu, "Using Unsupervised Deep Learning Technique for Monocular Visual Odometry," *IEEE Access*, 2019.

[10] B. Li, M. Hu, S. Wang, L. Wang, and X. Gong, "Self-supervised Visual-LiDAR Odometry with Flip Consistency," 2021.

[11] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*, 2007.

[12] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Trans. Robot.*, 2017.

[13] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011.

[14] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct monocular SLAM," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014.

[15] C. Duan, S. Junginger, J. Huang, K. Jin, and K. Thurow, "Deep Learning for Visual SLAM in Transportation Robotics: A review," *Transp. Saf. Environ.*, 2019.

[16] X. Yang, X. Li, Y. Guan, J. Song, and R. Wang, "Overfitting reduction of pose estimation for deep learning visual odometry," in *China Communications*, 2020.

[17] A. Dosovitskiy *et al.*, "FlowNet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[18] E. Parisotto, D. S. Chaplot, J. Zhang, and R. Salakhutdinov, "Global pose estimation with an attention-based recurrent network," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2018.

doi:10.20944/preprints202209.0276.v1

[19] P. Muller and A. Savakis, "Flowdometry: An optical flow and deep learning based approach to visual odometry," in *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, 2017.

[20] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia, "Exploring Representation Learning With CNNs for Frame-to-Frame Ego-Motion Estimation," *IEEE Robot. Autom. Lett.*, 2016.

[21] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2017.

[22] G. Costante and T. A. Ciarfuglia, "LS-VO: Learning Dense Optical Subspace for Robust Visual Odometry Estimation," *IEEE Robot. Autom. Lett.*, 2018.

[23] J. Zhang and S. Singh, "LOAM: LiDAR Odometry and Mapping in Real-time," 2015.

[24] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and Ground-Optimized LiDAR Odometry and Mapping on Variable Terrain," *IEEE Int. Conf. Intell. Robot. Syst.*, no. September 2019, pp. 4758–4765, 2018.

[25] M. Velas, M. Spanel, and A. Herout, "Collar Line Segments for fast odometry estimation from Velodyne point clouds," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2016.

[26] M. Velas, M. Spanel, M. Hradis, and A. Herout, "CNN for IMU assisted odometry estimation using velodyne LiDAR," in *18th IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC 2018*, 2018.

[27] Q. Li *et al.*, "Lo-net: Deep real-time LiDAR odometry," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019.

[28] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song, "L3-net: Towards learning based LiDAR localisation for autonomous driving," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019.

[29] Y. Cho, G. Kim, and A. Kim, "Unsupervised Geometry-Aware Deep LiDAR Odometry," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2020.

[30] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan, and S. Song, "DeepVCP: An end-to-end deep neural network for point cloud registration," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[31] J. Zhang and S. Singh, "Visual-LiDAR odometry and mapping: Low-drift, robust, and fast," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2015.

[32] J. Graeter, A. Wilczynski, and M. Lauer, "LIMO: LiDAR-Monocular Visual Odometry," in *IEEE International Conference on Intelligent Robots and Systems*, 2018.

[33] J. Yue, W. Wen, J. Han, and L.-T. Hsu, "LiDAR Data Enrichment Using Deep Learning Based on High-Resolution Image: An Approach to Achieve High-Performance LiDAR SLAM Using Low-cost LiDAR," 2020.

[34] V. Guizilini, J. Li, R. Ambrus, S. Pillai, and A. Gaidon, "Robust Semi-Supervised Monocular Depth Estimation with Reprojected Distances," 2019.

[35] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Rob. Res.*, 2013.

[36] J. Roche, V. De-Silva, and A. Kondoz, "A Multimodal Perception-Driven Self Evolving Autonomous Ground Vehicle," *IEEE Trans. Cybern.*, 2021.

[37] M. Valente, C. Joly, and A. De La Fortelle, "An LSTM network for real-time odometry estimation," *IEEE Intell. Veh. Symp. Proc.*, vol. 2019-June, no. Iv, pp. 1434–1440, 2019.

[38] R. Restrepo, “LiDAR Data to 2D.” [Online]. Available: http://ronny.rest/blog/post_2017_03_25_LiDAR_to_2d/.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural

doi:10.20944/preprints202209.0276.v1

networks," *Commun. ACM*, 2017.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[41] M. Valente, C. Joly, and A. D. La Fortelle, "Deep Sensor Fusion for Real-Time Odometry Estimation," in *IEEE International Conference on Intelligent Robots and Systems*, 2019.

[42] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A Search Space Odyssey,” *IEEE Trans. Neural Networks Learn. Syst.*, 2017.

[43] V. Buhrmester, D. Münch, and M. Arens, "Analysis of Explainers of Black Box Deep Neural Networks for Computer Vision: A Survey," *Mach. Learn. Knowl. Extr.*, 2021.

[44] Y. Shao, Y. Cheng, R. U. Shah, C. R. Weir, B. E. Bray, and Q. Zeng-Treitler, "Shedding Light on the Black Box: Explaining Deep Neural Network Prediction of Clinical Outcomes," *J. Med. Syst.*, 2021.

[45] Y. Liang, S. Li, C. Yan, M. Li, and C. Jiang, "Explaining the black-box model: A survey of local interpretation methods for deep neural networks," *Neurocomputing*, 2021.